Inference of IT Architecture based on Project Plans

Francisca Faria de Sales Parente Cambra
francisca.cambra@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2019

Abstract

Nowadays, few are the organizations that are willing to spend a lot of effort and time on keeping up-to-date architectural representations. This proved to be a very complex thing to do mainly because of the rapid pace of change that we are living today. Organizations need representations of their IT and business architecture to make several decisions and perform their daily work tasks. To reach this goal, we defined a set of rules to allow us to exchange relevant information between projects and enterprise architectures, by doing the integration of two technological components directly related to this dissertation, being these Microsoft-Project and Atlas. The results were compared with the manual solution previously created by Link Consulting, the company with which this thesis was made and which was responsible for the development of the Atlas tool. Several metrics have been evaluated, such as time, number of fields, effort, etc., so it can be concluded that the fact that this transmission of information is almost full automatic has many benefits for the company, namely regarding the time and quality efficiency of the planning and execution of the project portfolio itself.
Keywords: Project Plans, Project Management, Enterprise Architecture, EA Artefacts

1. INTRODUCTION

1.1. The Problem

The Information Technology (IT) of organizations grows based on the result of IT projects, so there are no IT changes that do not result from a project, planned and approved long time before [9, 11]. Despite the existence of such project plans, which consist of plans of temporary organizational structures that intend to create, or change, unique products or services so that benefits can be generated for the organizations [1], they are still unable to preserve up-to-date architectural representations that result from the reconciliation of all changes made within projects. *But why is it so hard to do that?*

The main cause is that it is very hard to link the Project Management (PM) and the Enterprise Architecture (EA) domains, meaning that it is difficult to recognize and detail the relationships between projects and the existing artefacts from the enterprise architecture, which is a topic that has been widely covered in recent times.

Additionally, the current practices implemented by organizations to try to solve the problem of not having architectural representations updated consists on having EA modelling tools with which the enterprise architects manually design the architectural models of the organizations. Drawing models by hand might seem a good approach, but when (and every time) there are changes, architects have to re-draw the models again, by hand, and this has currently been a well-known problem of organizations because it isn't a long-term solution for this matter.

Another issue that arises when it comes to preserve accurate models from the architecture is that the enterprise evolution is typically an asynchronous and distributed process [13]. This means that when planning and designing the enterprise evolution of a certain organization, there is not a formal way of communication between its stakeholders, since tasks are done by different actors from different departments. Therefore, when a certain department has the intention of doing some changes, the director of that department must meet with the directors of the other departments, so they can check the dependencies and impacts created by that change in each of the other departments. Despite the existence of such meetings, there is no way of sharing that combined knowledge across the enterprise. This lack of knowledge management impairs the flow of updated (and consistent) information between everyone involved in the organization and, subsequently, unable the update of architectural representations.

Hereupon, one can foresee that organizations need an enormous effort to update the models in the enterprise architecture tools used, and they are not predisposed to allocate that much effort

on maintaining architectural views updated. This effort is proportional with the speed with which the enterprise changes, i.e., the faster the enterprise changes, the more effort is essential to keep up-to-date representations. These enterprise changes are caused by transformation initiatives within projects, since projects are transformation elements of the organization [1] and their achievements drive the progress of the enterprise architecture.

Thus, we can realize that the enterprise cartographer, which is the person responsible for the study and elaboration of the enterprise maps (blueprints) to support the understanding of a dynamically changing organization, will need information from project plans to maintain those maps, i.e., the architectural models, properly updated throughout the evolution of the organization. This is a very complicated task since projects are always suffering variations, particularly when it comes to the planning phase of the project life cycle. The flow of information between the project management and the enterprise architecture domains is very hard to be achieved because of the three following issues:

- What information needs to be transferred between the project manager and the enterprise cartographer, so the architectural models can be updated based on project plans, and the project plans updated based on the architectural models?
- How should the architectural models and views be designed so the project's dependencies can be analysed and evaluated both by the project manager and the enterprise cartographer?
- How should project management and enterprise architecture tools be integrated and implemented?

1.2. Objectives and Motivation

This thesis aims at the definition of inference mechanisms of the enterprise architecture resulting from the analysis of the project portfolio, more concretely, the analysis of the plan of each project. The IT of organizations grows based on the result of IT projects, so there are no IT changes that do not result from a project, planned and approved long before [11]. Despite the planning of each project, organizations are unable to preserve architectural representations and models updated, that result from the reconciliation of all changes made to the different projects. To prevent this from happening, we are going to generate architectural views automatically because it is the only way to keep them updated.

First, we will explore, and consequently define, what is the information about the changes both in the architecture and in the project management levels required to transmit between project manager and enterprise cartographer. Thereafter, we will propound the best way to show that same information, after doing the integration between MS-Project, which will be used as a concept proof in the field of project management, and Atlas, as a solution of the enterprise architecture used by Link Consulting. The configurations and development of the solution will be guided to meet the requirements set by each organization, thus adjusting the representation of the enterprise architecture to each organization's needs.

2. FUNDAMENTAL CONCEPTS

Since the scope of this thesis lies in the areas of project management and enterprise architecture, we will focus on these two domains in this section, presenting all the fundamental concepts and explanations for the full knowledge of the problem and the solution.

2.1. Project Management

Before starting to explore the Project Management concept, we should first understand what a project is. PM2GUIDE defines a project as *a temporary organizational structure* which is setup to create a *unique product or service*, so-called the output, within certain constraints such as time, cost and quality.

Projects have the following goals [1]:

1. Define the project scope and deliverables (products or services);
2. Create a business justification for the investment (project's value for the organization, business context, list of alternative solutions, *etc.*);
3. Identify project stakeholders and define project core team;
4. Create the project plans to help guide and manage the project;
5. Assign and coordinate project work to teams;
6. Monitor and control of the project (progress, changes, risks, etc.);
7. Handover the deliverables and administratively close the project.

The focus of this thesis will be on the fourth point (4), which is mainly concerned with the elaboration of project plans. This task is performed by a role named project manager, and an important competency of him is to recognize how to manage

those project plans, which leads us to the following question *"What is project management and what does it complies?"*. Project management can be described as the activity of planning, organizing, securing, monitoring and managing the necessary resources and work to deliver specific project goals and objectives in an effective and efficient way [1]. Projects have a life-cycle that comprehends four phases, each of them representing a different period of time.

The **initiating phase** is the first phase and it defines what the project will do, i.e., defines the desired outcomes, assures that the project is properly aligned to the organization's strategic objectives and offers the essential information to get the project approval.

The next phase is the **planning phase**, where the objectives of the project are verified and developed into a specific and workable plan, named the project plan, that is going to be carried until the project and all its activities (work packages) are completed. The project plan can be updated many times during this phase, since a function inherent of the project manager and its team is to try to achieve the optimal balance between the project objectives, the available resources and the existing constraints, which are constantly changing.

The **executing phase** is the following phase, which comprises a restrict coordination of the execution of the project plans. It is where the project activities defined previously are performed and the project deliverables are produced.

After the production of the project deliverables, the project may enter the **closing phase**, which is the final phase of this life-cycle. During this phase, project activities are completed and documented, the finished deliverables are handed out to the client, the project resources are formally released, and the project is administratively closed.

In the meantime, another phase is being executed when the other four are as well, that is the **monitor** and **control phase**. This phase is subdivided in two, being these monitoring and controlling. Monitoring is about measuring the correct execution of the project, i.e., the ongoing project activities and the project variables (cost, time, effort) against project plans. Controlling is about taking corrective and preventive actions if the project execution varies from plans.

The focus of this work will be on the planning and executing phase. The former is important since the project plan identifies the activities, tasks and work packages needed to accomplish the project goals and defines approximations of the project activities' duration. Additionally, it is in the project plans that the source of information that changes in an organization is found, and that is needed to maintain the architectural representations constantly updated and valid, taking into account all changes that occur within a project portfolio. Those changes are performed when the project activities are executed, which corresponds to the executing phase of a project. That is where all the activities previously defined in the project plans are performed, that may or may not happen as described in the plan, since there may be discrepancies in terms of time, costs, resources, among others.

A critical part of any plan is dependencies. Within a simple project this might be moderately easy but when it comes to a project portfolio environment it is particularly challenging to identify, track and manage dependencies. According to the Portfolio Management Guide [6], a project portfolio is a collection of projects and other activities which are grouped together for a better control over financial and other resources, and to facilitate their effective management in terms of meeting strategic objectives. Even though projects of the portfolio might not be necessarily dependent nor related to each other, the project portfolio management allow us to identify how other projects may have an impact on your department in terms of process changes, and allow us to visualize where dependencies are within the portfolio, which is helpful when planning your project to better understand the existing risks.

2.2. Enterprise Architecture

The evolution and expansion of organizations are constantly increasing their complexity, and an architecture is considered an important instrument for managing and controlling that complexity [8]. Architecture is defined as the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principle guiding its design and evolution [3, 5], meaning that an architecture provides an integrated view of the system being studied or designed [8].

When this concept is applied in an organization, the Enterprise Architecture (EA) concept arises, which refers to a coherent whole of principles, methods and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems and infrastructure [8]. An enterprise architecture basically describes an organization from a perspective that combines business and IT viewpoints, in order to decrease the communication gap between that people. By using enterprise architecture, organizations are more easily able to improve their business and IT alignment [7], which contributes to their success.

There are several enterprise architecture frame-

works that define and help in how to create and use properly an enterprise architecture. An EA framework provides principles and practices for creating and using the architecture description of a system, and the most popular one used by organizations is The Open Group Architecture Framework (TOGAF). TOGAF is a widely used framework that combines a detailed method and a set of tools for supporting the development of enterprise architectures [4].

An enterprise architecture is composed by its artefacts (EA artefacts), which are separate documents that provide descriptions of an organization from different perspectives important for the various actors involved. EA artefacts can be considered as key elements of an EA practice, since they enable decision-making and IT planning in organizations [15, 2]. They are the observable elements of an organization, so it is accurate to claim that different organizations might consider different sets of artefacts as being part of their architecture, what will later correspond to their metamodels [12].

Transformation initiatives are a collection of planned and purposeful activities that might yield changes in the enterprise artefacts (the project deliverables) [12]. Those activities create, change and/or remove artefacts and their dependencies, which subsequently might result in changes in the architecture [13]. Currently the existing EA paradigms are not able to support the organization's pace of change, neither for designing nor for keeping the architectural models updated. It was this need, to keep apart design from representation, that made emerge the term enterprise cartography (EC). It is the process of representing an enterprise, i.e., the architectural artefacts and their relations, observed directly from reality, as for example from projects. It differs from enterprise architecture (EA) because it focuses on producing representations based on observations, so it does not include the purposeful design, as one expects in EA [13]. Since reality changes very quickly, it is essential to be able to propagate those changes, caused by transformation initiatives, into the architecture in an automatic way. That will be possible by generating the models automatically based on the information descendant from project plans.

3. SOLUTION DESCRIPTION

This thesis proposes a solution that will allow the collaboration between the project management and the enterprise architecture domains, by exporting project plans information from a project management tool named MS-Project. Afterwards, that information will be imported to Atlas, which is an enterprise architecture solution developed by Link Consulting, and the architecture models will be automatically updated, because this is the only way to keep them up to date.

3.1. Language

This subsection will help answering the first question presented in 'The Problem' subsection of the introduction, which is concerned in understanding what information from project plans will impact the organization's architecture. Not all project information will have impact on the architecture, so it is important to create a language that depicts only the information required to transmit.

As already mentioned, projects are transformation elements of the organization, and their achievements drive the progress of the enterprise architecture. Thus, enterprise changes are caused by transformation initiatives within projects, which can be observed in project milestones. They are used to mark a specific point along a project timeline, which identify the deliverables successfully achieved by the organization. Therefore, they provide us the necessary elements that will influence the organization's architecture, and consequently, that will be vital to preserve architectural representations updated. Besides the artefact name (element extracted from project milestones), it will be also needed its type, that is, the metamodel class to which it belongs to. This class will correspond to an ArchiMate metamodel class, since Atlas is based on it and supports this modelling language (its artefacts and relations).

After having the basic artefacts information, we have to delimit which are the associations between projects and the corresponding impacted artefacts. The first terms to arise were CREATE and DELETE, which represent the moments when EA artefacts are created and deleted by some project. The former means that a representation of the artefact was effectively created in the enterprise architecture, so it starts to become available for use by the company's operations, and the latter means that the deleted artefact becomes sort of invisible to the organization. In spite of being no longer part of the company's operations, continue to exist a representation of it in the architecture.

Thereafter, the term USE was raised. It represents the moment when a project uses data from a certain artefact, that must exist previously in the architecture. Its purpose is basically to ensure that there is a way of relating an artefact to the outside, because otherwise there would be no interactions. Logically doesn't make much sense to create an artefact, for instance an application component like 'ERP', that is not going to be used later on by any other artefact; only existed in the architecture, without having any external dependencies. Therefore, this type of impact represents the getting of infor-

mation without being actually changing it. But what if an organization wants to change some artefact?

According to ArchiMate [10], an artefact does not change itself since it doesn't have inner properties; what changes are the relations that it presents with its constituents, with its components. In the architecture domain, its said that nodes have other nodes, being nodes the existing artefacts. A node cannot change, only the relation between parent node and child node, so this fact prevented the creation of the impact CHANGE. Instead of having that project *x* changes artefact *a*, we will have that project *x* deletes artefact *a1* and creates artefact *a2*.

Another important point to consider is the difference between the EA artefacts domains, which implies getting to know them a little better. According to The Open Group, EA artefacts are split into logical and physical elements. The former represents implementation or product-independent encapsulations of data or functionality, whereas the latter represents tangible devices, software components, etc.. Briefly, logical artefacts correspond to more abstract elements, such as application components, and physical artefacts to more tangible ones, such as installation versions. To be classified as a physical artefact, one should be able to store it in a digital device or other physical system, i.e., one must be composed of bits and bytes. Therefore, can be assigned to the logical domain any artefact of construction, like application components, and to the physical domain any artefact of installation that produces executables ('.exe'), like nodes and system softwares.

The difference between these two artefact domains influences the type of impact they can have on the enterprise architecture. For instance, for logical elements it would make sense to say they can be created and deleted by a project, but it won't for physical elements, since they correspond to installation versions and they cannot be created or deleted by a project. Instead, they can be installed or uninstalled in a physical system, that should exist primarily in the architecture.

Thus, the impact types that fit physical artefacts are DEPLOY and UNDEPLOY, which correspond respectively to the logical impacts CREATE and DELETE. The first impact corresponds to the moment when the respective installation version will become available for the company to use, and the second the moment when the version will stop being available for use. Typically these two impacts are related; when an organization does the deployment of a certain installation version (physical artefact), it is also uninstalled the previous/old version, so we created another impact, the UPGRADE. This impact simplifies cases as the one described above, allowing to detail which version will be uninstalled, and located in what digital device(s), to give place to the (new) version, also specifying its location or locations. Those locations will be vital since they are a precondition of physical artefacts, that according to ArchiMate must be stored in a digital device or other physical system, like the node artefact.

Briefly, logical artefacts might be created, used and deleted by a project. Both USE and DELETE actions imply that the artefact being referenced must exist formerly in the architecture. Regarding physical artefacts, the types of impact that are appropriate are the DEPLOY, UNDEPLOY, UPGRADE and USE.

Based on the information that needs to be transferred between project plans and architectural models, and giving continuity to what was referred to in this entire subsection, our language will be composed by the artefact name (subject), its metamodel class according to ArchiMate (class), the impacts they suffer from the project itself (action) and the planned start date and planned finish date when that action occurs (dates). All this information have to be extracted from the milestones of the project plan, since these are the points in time that have the transformations that will occur in the organization and that will impact the organization's architecture.

3.2. Impact visualization

It was already decided which information needs to be transmitted from MS-Project to Atlas, so the next step consists in finding a simple and effective way to represent that same information, which will help answering the second question of 'The Problem' subsection of the introduction.

For this effect we will use blueprints, since they exist to show architecture elements in the state corresponding to that point in time, and their generation are one of the main features supported by Atlas. After exporting the project milestones with the language defined above from MS-Project, Atlas will import them and a blueprint will be automatically generated after running some queries implemented for this purpose. The blueprint created is divided in four main segments:

1. Project and Work-packages

   This segment exists not only to allow the visualization of the project that will be analysed, hereafter called as 'our project', but also to visualize all the projects it aggregates, that is, its sub-projects. Both projects and its sub-projects will be designed in this thesis as work-packages, since ArchiMate refers that these EA elements might be used to model a series of actions identified and designed to achieve

specific results within specified time and resource constraints, which is concordant with the project definition itself. [10]

2. Artefacts

   The EA artefacts that undergo any kind of action/impact caused by 'our project' and/or its sub-projects are present in this segment. It is composed by two distinct sections, one that refers to the physical artefacts and another that refers to the logical artefacts. Each of these sub-sections is divided again, being every division representing the specific type of impact that those EA artefacts suffer, being that for logical artefacts we'll have the CREATE, USE and DELETE containers, and for physical artefacts the DEPLOY, USE and UNDEPLOY.

3. Related Projects

   The related projects segment represents all the projects that impact at least one artefact from the Artefacts segment, that is, have any type of relation with an artefact that is created, used, deployed, deleted and/or undeployed by 'our project' and/or by one of its sub-projects.

4. Error

   This segment exists in order to facilitate the effort made by the enterprise cartographer and the project manager in the identification and analysis of the dependencies between several concurrent projects of an organization. It is quite recurrent that numerous projects have dependencies with the same artefact, and sometimes those dependencies become invalid. Obviously those connections must be congruent and valid so that there is an effective execution of the projects, and so that organization losses can be minimized. Every time there is a conflict between concurrent projects, which means they have conflicting impacts in the same enterprise architecture element, that situation will be visualized in this segment of the blueprint, following the techniques explained below.

3.3. Alerts

For the analysis of the dependencies between projects and the EA artefacts they have an impact on, was developed a feature of alerts which is the support of the blueprint error segment. It consisted on the creation of different containers, where each one was made of a pair of impacts that could generate errors between projects and artefacts. The first impact of the pair refers to the relation 'our project' or one of its sub-projects has with the artefact, and the second impact refers to the relation a related project has with that same artefact. For them to be designed, the relationships they present must be invalid at a business level. For example, an error can emerge when a project deletes an artefact that hasn't been previously created by a related project. This is problematic since the date when the related project creates the artefact must occur before the project deletes it, and not after it, because the artefact isn't available yet for the company's operations. In this situation, the container would be called '**deletion-creation**', since the first impact corresponds to the relation the artefact has with the project we are analysing, and the second corresponds to the relation the artefact has with a related project. The several types of alerts that might occur are the following: creation (-creation), deletion (-deletion), creation-deletion, creation-using, deletion-creation, deletion-using, using-creation and using-deletion.

3.4. Integration

The information that needs to be exported from project plans was already settled after defining a language to apply, and the blueprint that will be displayed after importing that same information from Atlas was as well. The next step consists in understanding how the information exported from MS-Project correlates with the information needed to import from Atlas.

For the blueprints generation, a mandatory source of information is the artefact templates. Atlas stores the artefact information for each class in separate files, so there are as many files as the number of different classes in the metamodel adopted by the organization in question. Those files are called artefact templates, and their file format is the Excel format, since it is supported both when importing and exporting files from Atlas, so it will be the one used along this integration.

In order to propagate the information descendant from project plans to the architectural models, and vice-versa, we used a well-known feature of Microsoft Office named macro. This feature exists in several Microsoft tools, such as in MS-Project and Excel. Microsoft Office files may contain embedded code, known as a macro, written in the Visual Basic for Applications (VBA) programming language. A macro is a subroutine that performs a certain task. They are normally used to automate repetitive tasks, but can also be used to perform tasks that would be impossible to do in the normal mode, that is, without using macros.

So the solution could be reached, were created different macros for the different defined objectives, some of which are located in the project management tool chosen, and others in Excel. Regarding MS-Project, was created a macro to export only the

information needed to keep the architectural representations updated, which consists in the different elements defined in our language, to an external excel file, hereafter referred as taxonomy file. It was also created another macro that imports the information coming from the architecture (Atlas) to MS-Project, so project plans can be maintained properly actualized when changes are made in the architectural models. Regarding Excel, were created the following six macros:

- *GetTaxonomy* - Copies the taxonomy file into the first sheet of an auxiliary workbook.
- *GetSheets* - Copies all the artefact templates extracted from Atlas into the following sheets of the workbook (sheet!=1), by alphabetical order.
- *UpdateData* - Propagates all the data (dates and impacts) from the first sheet, which corresponds to the taxonomy file descendant from MS-Project (project plan), to the other sheets of the workbook, which correspond to the artefact templates supported by Atlas.
- *ReverseData* - Propagates all the data (dates and impacts) from every sheet of the workbook different than the first one, which corresponds to the artefact templates supported by Atlas, to the first sheet, that corresponds to the taxonomy file.
- *SaveSheets* - Saves all the changes made to the templates of the artefacts, i.e., to every sheet of workbook different than the first one, into the files from where they were copied.
- *SaveTaxonomy* - Saves all the modifications made to the first sheet, that is, to the taxonomy file, into the file from where it was exported.

The previous macros can only be run once we have all the information gathered in the same folder, which means we must have both the taxonomy file exported from MS-Project after having the project plan filled in, the artefact templates downloaded from Atlas, and the workbook where the macros are located, which will be the place where all the information spread occurs. This may be either to keep the architecture models updated based on the information from the project plans, or the other way around, that is, to keep the project plans updated based on the changes made in the architecture.

4. Results & discussion

4.1. Metrics

In order to be able to evaluate correctly this solution we have to compare it with the solution that Link provides to its customers today to solve the problem presented in this document. The method implemented by Link consisted in introducing manually the project information, the impacted artefacts and dates into a form directly in Atlas. This might seem the easiest solution for the problem explained in the introduction section, but it is not definitely the best for, as we will see along this subsection.

When evaluating the introduction of fields metric, we can observe that whenever the IT project does not imply many creations, updates, uninstalls or any other impacts on the artefacts, both previous solution and the one developed in this thesis apply well. In the former, there are not many artefacts to be manually introduced into Atlas forms, and in the latter because the whole process encompasses almost none field introductions.

When the project impacts many artefacts, the scenario is completely different. In the old solution you need to insert each of the artefacts into their corresponding fields, which implies a huge effort and time. Otherwise, with the solution presented in this document, artefacts will be updated almost automatically, only having to perform very simple tasks. In both solutions, and regardless of the number of artefacts, the field entries during project planning in MS-Project are not being considered as this would have to be done anyway by the project manager. Even so, the introduction of fields is smaller with the implementation of this automatic solution.

Regarding the creation of the project plan during its planning, it is quite obvious that the easiest method is the manual, since the number of entries to write down is smaller. When it comes to updating it, the fields are filled along the project planning in MS-Project (with the new solution), and then it is only needed to export that plan through a macro, download the artefact templates, open a file that makes the update needed to propagate project plan information to the artefact templates, and import those templates from Atlas. Even if the number of artefacts being impacted by a certain project was equal to one hundred, the effort needed would be the same as for the update of just one artefact. With the old solution we would have to introduce a different row for each impacted artefact from each class. If the number of impacted artefacts was one hundred, the project manager would have to fill one hundred names in the respective fields of the form.

Considering time and effort metrics, they are both proportional to the number of fields to insert. When this task is done manually using Atlas forms, if the number of artefacts is low, both time and effort are still warranted; if not, the number of artefacts is high, and time and effort will be also high

because of the enormous number of fields to introduce, which does not verify with this new solution. The time taken to transmit project plan information to architectural models, and vice versa, will be the same regardless of the number of artefacts, since the tasks to be performed will always be constant and unchanging. Obviously the execution time of those tasks, however simple and shortly they are, will be more sensible and more noticeable when compared to the need of entering dozens of fields in Atlas forms. When those times and effort are compared to the ones from the previous solution, only when a small number of artefacts is involved, the latter prevails at a performance level compared to the new solution. This happens since although the tasks to be performed for the automatic transmission of information are quite simple, they take time, and it turns out to be more time consuming than where the field input is minimal.

Finally, regarding the user friendly metric we have that the new solution fits in the positive value in all cases, i.e., creating or updating a small or large number of artefacts, since the process is performed almost automatically, only requiring little interactions from the project manager. One factor that weighs heavily in this appreciation is that the project manager does not have to recurrently use two tools in his day to day work. In contrast, we have that the old solution only shows good results when the number of artefacts is reduced for the reasons mentioned above. In all other cases, it does not support an efficient and fast method of propagating project plan information to the architectural representations, and requires the project manager to enter the same information twice into separate tool, which in addition to being exhaustive, is quite susceptible to misaligned and incoherent information.

In addition, one aspect that is quite negative with the old solution is that when enterprise architecture models are modified because there are incongruities and invalid relations between projects and artefacts, there is absolutely no way to pass this information to project managers, and more specifically, to project plans. Therefore, the respective milestones must be manually changed in MS-Project, which again proves to be a very inefficient and quite exhausting process.

The comments and reflections made along the previous paragraphs allow us to realize that although the configurations required for the new solution to be properly implemented, it turns out to be much more efficient, less susceptible to human error, faster and less effort consuming for project managers.

4.2. Methodologies

The solution implemented along this thesis supports popular development methodologies, such as the waterfall and the agile methodologies. The first one is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one, and corresponds to a specialisation of tasks. The waterfall development moves a project through various Software Development Life Cycle (SDLC) phases - analysis, design, development and testing, implementation, documentation, and evaluation. One phase is completed in its entirety before moving on to the next phase, which is why it tends to be among the less iterative and flexible approaches [14].

Instead, agile development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers. It advocates adaptive planning, evolutionary development, early delivery and continual improvement, and it encourages rapid and flexible response to change. Agile software development uses iterative development as a basis but defends a lighter and more people-centric viewpoint than traditional approaches. An iterative product development allows the software to evolve in response to changes in business environment or market requirements, having continuous feedback that it provides to successively refine and deliver a software system [14].

These are some software development approaches that are sustained by the solution presented in this document, since we offer the possibility to automatically update the enterprise architectural representations countless times based on the information gathered from the project plans, and vice versa. It is possible to continuously keep updating those same objects (enterprise architectural models and project plans) whenever they change, because both of them are not constant and evolve over time within organizations.

5. Conclusions

After describing the problem and respective solution in the earlier sections, we can understand that at any given time, in a medium or large organization, multiple transformation initiatives from different scopes and sizes may occur, making not only the opportunity for an organization to preserve an accurate representation of the enterprise reality impossible, but also future changes that might affect the architecture.

The integration between MS-Project and Atlas will make it possible for the organizations to do an accurate planning of their projects since this solu-

tion will maintain their architectural representations up-to-date, by observing the enterprise reality and by analysing the dependencies that would emerge if the changes implemented by certain projects were actually executed. This will only be achieved after defining a set of rules (language) that will reduce the misalignment between the concepts used both by the project manager and the enterprise cartographer, in order to extract valuable information from project plans, i.e., information that will impact the architecture of the organization.

Without a good architecture it is very difficult for organizations to achieve business success, so this approach will allow enterprise cartographers and project managers to identify possible conflicts that might happen with their working projects and with other coexisting projects within an organization, after analysing the enterprise reality. For instance, they will get enterprise information to recognize and assess the existing impacts and dependencies between projects from different departments, and project managers will get a better understanding about the future state of the organization's architecture, so they can plan ongoing projects and the next transformation initiatives more accurately.

Through the metrics selected and after their analysis, we can understand that we have successfully implemented a solution that generates architectural blueprints automatically on a weekly basis based on information retrieved from IT project plans. The need to link project management and enterprise architecture domains is increasingly becoming a business reality, but there aren't efficient methods that make the automatic generation of architectural representations possible. The definition of mechanisms for this inference eases the effort made by organizations, because they have constantly multiple transformation initiatives ongoing and it is impossible for an enterprise architecture to maintain an accurate representation, not only of the enterprise reality but also any future changes that might occur.

This solution will facilitate the maintenance of architectural models updated, allowing to project managers a better planning of projects and a better understanding of the future state of the organization's architecture, and to enterprise cartographers the facilitation that occurs since it is not necessary to draw up a new map whenever an enterprise's architecture changes. It will also facilitate and improve the analysis of dependencies and impacts that occur within an organization, whether within the same department or between different departments, as the generated blueprints present the inconsistencies found in the organization. This is only possible since project managers need enterprise information to plan their projects, and to identify and assess project risks and impacts so they can be communicated to their project teams. Similarly, enterprise cartographers need project data to maintain the information under their responsibility accurate (architectural models).

References

[1] *The PM² Project Management Methodology Guide – Open Edition*. Luxembourg, v.0.9 edition, 2016.

[2] R. Abraham. Enterprise architecture artifacts as boundary objects - a framework of properties. In v. H. E. C. R. van Hillegersberg, J., editor, *Proceedings of the 21st European Conference on Information Systems*, pages 1–12, Netherlands, 2013. Association for Information Systems.

[3] I.-S. S. Board. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.*, 2000.

[4] T. O. Group. The open group architectural framework (togaf), April 2018.

[5] International Organization for Standardization, Switzerland. *ISO/IEC/IEEE 42010:2011(E) - Systems and software engineering Architecture Description*, 1 edition, 2011.

[6] C. Kilford. *Portfolio Management Guide*. OGC, IIIIIMMMMPPPOORTTTAANNNTTEEEE.

[7] S. M. S. I. Kotusev, S. Investigating the usage of enterprise architecture artifacts. In *Twenty-Third European Conference on Information Systems (ECIS)*, Germany, May 2015.

[8] e. a. Lankhorst, M. *Enterprise Architecture at Work: Modeling, Communication and Analysis*. Springer-Verlag, Germany, 2 edition, 2005.

[9] M. McGregor. Align your ea and pmo teams. Technical report, Changepoint, 2015.

[10] The Open Group. *ArchiMate © 3.0.1 Specification*.

[11] L. J. S. A. e. a. Sousa, P. An approach for creating and managing enterprise blueprints: A case for it blueprints. In B. J. D. G. Albani, A., editor, *The 21st International Conference on Advanced Information Systems*, volume 34, pages 70–84, Germany, 2009. Springer-Verlag.

[12] S. P. C. A. Tribolet, J. The role of enterprise governance and cartography enterprise engineering. *Enterprise Modelling and Information Systems Architectures (EMISA)*, 9(1):12, June 2014.

[13] S. P. G. S. Tribolet, J. Enterprise cartography: From theory to practice.

[14] Wikipedia. Software development process, August 2019.

[15] F. R. Winter, R. Essential layers, artifacts, and dependencies of enterprise architecture. In A. Vallecillo, editor, *2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 30–37, China, 2006. IEEE.