Inference of IT Architecture based on Project Plans

Francisca Faria de Sales Parente Cambra

Thesis to obtain the Master of Science Degree in
Information Systems and Computer Engineering

Supervisor: Prof. Pedro Manuel Moreira Vaz Antunes de Sousa

Examination Committee

Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Pedro Manuel Moreira Vaz Antunes de Sousa
Member of the Committee: Prof. André Ferreira Ferrão Couto e Vasconcelos

November 2019
Acknowledgments

To my parents, for having supported me financially and emotionally during all these years, which made possible my education and my non-cessation of the course, which often seemed the right and easiest way to take. Their encouragement, caring and friendship were vital for this enormous achievement, as well as their advices.

To the rest of my family, who despite having given me a headache for not going to all the family events that took place, continued to give me support and value when I actually attended some of them, since they realized that I was doing an effort to do so.

To my friends, that helped me grow as a person and were always there for me during good and bad times in my life.

To all the community of the Instituto Superior Tecnico (IST) - teachers, colleagues and friends - that enabled me this great learning and that without them it would not have been possible to get here. In particular to João Zeferino, who was my companion at Link, and with whom I had productive conversations related to the thesis since the domains of our theses is similar.

To some of the employees of Link Consulting, who welcomed me and provided me with good moments during working hours. To David Pinto Moreira, for helping me to familiarize with the tool used by the company, Atlas, and for having the patience to explain things that were simple for him, and were huge, and sometimes even seemed impossible, challenges for me.

To my supervisors, professor Pedro Sousa and professor Maria do Rosário, that helped me to overcome all the doubts that arose to me, and that consequently allowed this work to be developed, due to their availability and predisposition.

To each and every one of you who have had directly and indirectly supported me in this long, difficult and challenging journey, thank you.
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.
Resumo

Atualmente, poucas são as organizações que estão dispostas a gastar muito tempo e esforço para manter as representações da sua arquitetura atualizadas. Isso provou ser uma tarefa bastante complexa maioritariamente devido ao rápido ritmo de mudança que estamos a viver nos dias de hoje. As organizações necessitam de representações da sua arquitetura de IT e de negócio para efetuar várias decisões e executar as suas tarefas de trabalho diárias. De forma a atingir este objetivo, nós definimos um conjunto de regras que nos permitiram trocar informações relevantes entre os planos de projeto e a arquitetura empresarial, através da integração das duas componentes tecnológicas diretamente relacionadas com esta dissertação, sendo estas o Microsoft-Project e o Atlas. Os resultados foram comparados com a solução manual previamente criada pela Link Consulting, empresa com a qual foi realizada esta tese e que foi responsável pelo desenvolvimento da ferramenta Atlas. Várias métricas foram avaliadas, tais como o tempo, número de campos, esforço, etc., pelo que se pode concluir que o facto desta transmissão de informação ser quase totalmente automática traz muitos benefícios para a empresa na medida em que o tempo e a qualidade do planeamento e da execução do portfólio de projetos se tornam muito mais eficientes.

Palavras Chave

Planos de Projeto; Gestão de Projetos; Arquitetura Empresarial; Artefatos da Arquitetura.
## Contents

List of Figures ix  
List of Tables xi  

1 Introduction  
  1.1 The Problem ................................................. 2  
  1.2 Thesis Objectives ......................................... 4  
  1.3 Thesis Outline ............................................ 5  

2 Fundamental Concepts  
  2.1 Project Management ...................................... 8  
    2.1.1 Project Management Tools .............................. 12  
  2.2 Enterprise Architecture .................................. 13  
    2.2.1 Enterprise Architecture Tools .......................... 15  
  2.3 Link between Project Management (PM) and Enterprise Architecture (EA) ...................... 16  

3 Solution Proposal  
  3.1 Language ................................................. 20  
    3.1.1 What types of artefacts? .............................. 21  
    3.1.2 What types of impacts? ............................... 21  
    3.1.3 Limitations of impacts ............................... 23  
    3.1.4 Language rules ...................................... 24  
  3.2 Impact visualization ..................................... 25  
  3.3 Alerts ................................................... 30  
    3.3.1 How will it be generated? ............................. 36  
  3.4 Macros .................................................. 37  
    3.4.1 What is a Macro? ..................................... 37  
    3.4.2 How is it going to work? ............................. 37  
    3.4.3 Microsoft-Project (MS-Project) Macros ............... 41  
    3.4.4 Microsoft Excel Macros ............................... 44  
    3.4.5 Macros Support ...................................... 47
# List of Figures

2.1 The PM Methodology Guide. ............................................................... 9  
2.2 The PM project life cycle. ............................................................... 10  
2.3 ArchiMate EA Artefacts. ................................................................. 14  

3.1 Solution deployment steps. ................................................................. 20  
3.2 ArchiMate relations between artefacts. .............................................. 22  
3.3 Project selection in Atlas. ................................................................. 26  
3.4 Blueprint showing the project impact detailed. .................................... 27  
3.5 Desired arrangement of elements in the container. ............................... 30  
3.6 Expectation of error container. ......................................................... 31  
3.7 Reality of error container in Atlas. ................................................... 31  
3.8 File template of the Application Component class, exported from Atlas. .......... 39  
3.9 How to export artefact names: step 1. ............................................... 39  
3.10 How to export artefact names: step 2. ............................................ 40  
3.11 Example of a lookup table in MS-Project. ......................................... 41  
3.12 Artefact class names exported from Atlas. .......................................... 42  
3.13 Excel file exported from MS-Project after running the 'Export' macro. .......... 43  
3.14 How to import artefact templates: step 1. ....................................... 46  
3.15 How to import artefact templates: step 2. ....................................... 47  
3.16 How to import artefact templates: step 3. ....................................... 47  
3.17 Indexation file template. ............................................................... 48  
3.18 File template of the System Software class, exported from Atlas. ............ 51  
3.19 Solution from the user perspective. ................................................ 53  
3.20 Use Case of the solution. ............................................................. 54  

4.1 Project properties in Atlas form. ................................................... 58  
4.2 Impacted artefacts in Atlas form. ................................................... 58
4.3 Continuation of impacted artefacts in Atlas form. ........................................ 59
4.4 Data analysis. ............................................................................................... 61
1 Blueprint created for the analysis of project dependencies ................................ 71
List of Tables

3.1 Summary of the language concepts. .................................................. 25
3.2 Possible combinations for logical artefacts. ................................. 35
3.3 Possible combinations for physical artefacts. ............................... 35
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APM</td>
<td>Application Portfolio Management</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model and Notation</td>
</tr>
<tr>
<td>EA</td>
<td>Enterprise Architecture</td>
</tr>
<tr>
<td>EC</td>
<td>Enterprise Cartography</td>
</tr>
<tr>
<td>IST</td>
<td>Instituto Superior Tecnico</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>LC</td>
<td>Link Consulting</td>
</tr>
<tr>
<td>MS-Project</td>
<td>Microsoft-Project</td>
</tr>
<tr>
<td>PM</td>
<td>Project Management</td>
</tr>
<tr>
<td>PMO</td>
<td>Project Management Office</td>
</tr>
<tr>
<td>PPM</td>
<td>Project Portfolio Management</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software Development Life Cycle</td>
</tr>
<tr>
<td>TOGAF</td>
<td>The Open Group Architecture Framework</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>VBA</td>
<td>Visual Basic for Applications</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
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 [1, 2]. 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 [3], 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 [4]. 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 [3] 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 dependencies can be analyzed 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 Thesis Objectives

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 [1]. 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. We will exhibit approaches to produce and maintain IT architectural landscape up-to-date, which will be triggered by the existing project deliverables, allowing IT organizations to align foreseen architectural representations with their IT portfolio management planning. Additionally, organizations use enterprise architecture discipline to help them improving business and IT alignment, since it conducts and describes an enterprise from an integrated business and IT perspective, so it will also be presented a method to merge both domains (PM and EA), by extracting project plan information from project management tools, and by using that same information to update enterprise architecture models, since that is the source of information and transformation.

Therefore, 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 the project manager and the enterprise cartographer, who are the people who have responsibilities in these two domains, by defining a new language/taxonomy. Afterwards we will propound the best way to show that same information, after doing the integration between Microsoft-Project (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 (LC). 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.
1.3 Thesis Outline

This thesis is divided in six chapters, each comprising one or more sections.

Chapter 1:

The first chapter addresses the general scope and context of the work, by covering the problem statement, the thesis main objectives and underlying motivation.

Chapter 2:

The second chapter is an overview of project management and enterprise architecture domains. Fundamental definitions and concepts regarding each one are clarified in order to achieve the full knowledge of the problem and solution inherent in this paper. Solutions that exist nowadays and are already implemented to try to solve ongoing problems are also described, and consequently the tools that support it.

Chapter 3:

The third chapter concerns fundamental research aspects necessary to define a set of rules (a new language) to be followed before implementing the solution. Once having the language set, it will be connected with the other component of this thesis, which is focused in representing and designing architectural models with the information descendant from project plans. Additionally, this chapter also describes the solution implementation, that is, how the propagation between MS-Project and Atlas information was reached. It is given a contextualization with the feature used - macro - and the different approaches are discussed particularly.

Chapter 4:

The fourth chapter refers to the metrics used, and respective results, so the solution could be evaluated. The different methodologies supported by this thesis are also described in here.

Chapter 5:

The fifth chapter comprises the conclusion of this work, as well some limitations and future work proposals with some ideas and respective foundations exposed.
Fundamental Concepts
2.1 Project Management

Before starting to explore the project management concept itself, 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 [3].

- *Temporary* because a project has a well-defined start and end.
- Has a *unique output* since the project’s product or service has not been created before, and even it may be identical to an existent one, there will always be a certain level of uniqueness in the new product or service, due to the introduction of changes.
- The project is defined, planned and executed under certain *constraints* of *time, cost, quality* and other constraints associated to the environment of the project’s organization, capabilities, etc..

According to PM2GUIDE [3], projects have the following goals:

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.

Once understood the project concept and its functions within an organization, the focus of this dissertation will be on the fourth point (4), which is mainly concerned with project plans.

A project plan is a written projection of project activities and resources needed to execute a process, that is executed by the project manager role. An important competency of a project manager is to recognize how to manage those project plans, which leads us to the following question “*What is project management and what does it comply?*”.

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 [3]. As already said, this discipline is performed by a project role named project
manager, who manages the project’s daily progress to deliver the outputs within the agreed constraints. In order to achieve it, the project manager should, among many other things, understand the project management methodology used in its organization. The PM Methodology described in PM2GUIDE [3] joins all the best practices in this domain, and it is composed by four main activities represented by the pillars of Fig. 2.1:

![PM Methodology Guide](image)

**Figure 2.1:** The PM Methodology Guide.

1. A project governance model (i.e. roles and responsibilities);
2. A project life cycle (i.e. project phases);
3. A set of processes (i.e. project management activities);
4. A set of project artefacts (i.e. templates and guidelines).

From the four pillars pronounced in the PM Methodology, our emphasis will be on the second point – **project life cycle** – which consists on describing the diverse phases of a project, identifying in what they consist and what are their objectives.

**The PM project life cycle**

Similar to everything with a life cycle, also projects have a beginning and an end. The beginning of a project corresponds to the date where it is approved, and the end corresponds to the date where it is completed. The project life cycle contains all the project activities between these two dates.
The PM project life cycle comprehends four phases, and each of them represents a different period of time in the life of a project. The figure below (Fig. 2.2) shows us over time the various project phases present in the PM project life cycle.

![Diagram of PM project life cycle with phases: Initiating, Planning, Executing, Closing, Monitor & Control.]

**Figure 2.2**: The PM project life cycle.

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

In the **planning phase**, 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 identifies and organizes the project into activities, tasks and work packages needed to accomplish the project goals. It also defines a base to estimate the duration of the project, determine the required resources and schedule the work. The project (work) 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.

During this phase, the project manager prepares the deliverables acceptance plan and gives it to a business manager (e.g. enterprise architect), so he can review and validate the deliverables acceptance requirements, activities and associated metrics. The deliverables acceptance plan documents the deliverables acceptance approach, responsibilities, activities and acceptance criteria so that project’s deliverables can be formally accepted based on objective criteria and predefined timelines. Once these deliverables are approved by the business manager, the project goes to the next phase, the **executing phase**. This phase comprises a restrict coordination of the execution of the project plans, i.e., it’s where the project activities defined previously in the project plans are performed, and, consequently, 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
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 at any level of business change 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. It is very important to understand the differences among projects and portfolios, as well as their specific management requirements. So, what is a project portfolio?

According to the Portfolio Management Guide [5], 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. The projects of the portfolio may not be necessarily dependent nor related to each other, so what are the key benefits of a Project Portfolio Management (PPM)?

1) Have a clear picture of projects
   Allow us to see and understand the overall landscape of change in the organization due to collaborative work between departments, in order to see all the projects and their timeline and dependencies.

2) Understand which projects impact you
   Allow us to identify how other projects may have an impact on your department in terms of process changes.
3) Understand dependencies between projects

Allow us to visualize where dependencies are within the portfolio, which is helpful when planning your project to better understand the existing risks.

2.1.1 Project Management Tools

There are several tools to support an efficient project management, such as Jira by Atlassian, Zoho Projects by Zoho Corporation, Teamwork Projects by Teamwork, Microsoft-Project by Microsoft, Accelo by Accelo and Kantask by Kantask. They all support agile methodologies, collaboration tools and milestone tracking. Despite their specific advantages and disadvantages, the one picked for this dissertation was MS-Project. This tool was elected just as concept proof because I have already worked with it in the course of Information Systems Project Management in Instituto Superior Tecnico, so I was already familiar with it.

MS-Project is a project management software developed by Microsoft to assist project managers on their daily work, for instance when they are developing a plan, assigning resources to tasks, tracking progress, managing the budget and analyzing workloads. The two capabilities provided by this tool that we are mostly interested in are the possibility to assign dates for each task, as well as the information if they are milestones or not.

In this tool, every task has an actual start and end date, which correspond respectively to the date the task has started to be performed, and the date it ended. These dates are being filled as the project progresses. Additionally, tasks have also planned start and end dates. The first one corresponds to the date the task is supposed to start, and the second one is the date the task is supposed to end, which are filled throughout the project planning. These two planned dates are just assumptions and a prevision given by project managers that is stated in project plans, since those plans are just a promise and a guide to follow. They rarely describe what and when tasks happen with precision.

Once talked about the dates feature of MS-Project, the other crucial point for this work to be possible is the presence of milestones. They are used to mark a specific point along a project timeline, which identifies the deliverables successfully achieved by the organization, after the tasks have been reviewed. They provide us the necessary elements that might have impact on the enterprise architecture, and that will allow to preserve up-to-date architectural models. The deliverables achievement correspond to the changes made within an organization, since they are the projects output, and IT projects implement changes, so those deliverables (specified in milestones) are going to impact the enterprise architecture, and will be vital to keep architectural models updated.
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 [6]. 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 [7, 8], meaning that an architecture provides an integrated view of the system being studied or designed [6].

When this concept is applied in an organization, the 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 [6]. An enterprise architecture basically describes an organization from a perspective that combines business and IT viewpoints, in order to decrease the communication gap between business and IT people. By using enterprise architecture, organizations are more easily able to improve their business and IT alignment [9], which contributes to their success.

There are several enterprise architecture frameworks 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 often used by organizations is The Open Group Architecture Framework (TOGAF), which was developed in 1995 and its currently in version 9.2, released recently on April 16th, 2018.

Furthermore, TOGAF is a widely used framework that combines a detailed method and a set of tools for supporting the development of enterprise architectures [10]. It does not provide elements of transformation that support the evolution of an organization, which is why we are not going to use it entirely. Those elements are what compose an enterprise architecture, and they are called enterprise architecture artefacts (EA artefacts). They 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 [11, 12]. 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, which correspond to the metamodels adopted by them [13].

Since each organization implements a different metamodel, because they are set up to sustain and adopt the needs of each organization, this dissertation will sustain the EA artefacts offered by the ArchiMate metamodel because it is very broad and allows organizations to easily adjust to it.

Archimate is a modelling language that is an open and independent enterprise architecture standard that sustains the description, analysis and visualisation of architecture within and across business domains. Additionally, it is one of the open standards hosted by The Open Group and is fully aligned with TOGAF. All the architectural elements it offers are depicted in the image below, and are both supported...
by Atlas and this dissertation.

Figure 2.3: ArchiMate EA Artefacts.

Subscribing what was said in the previous section, the deliverables resultant from the project plans will have impact on the enterprise architecture, and the opposite too. According to TOGAF 9.1, deliverables are architectural work products that represent the output of projects, so they will be caused by transformation initiatives, and those deliverables will typically be archived after the project completion [10].

Transformation initiatives are a collection of planned and purposeful activities that might yield changes in the enterprise artefacts (the project deliverables) [13]. Those activities create, change and/or remove artefacts and their dependencies, which subsequently might result in changes in the architecture [4]. 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). This is essential to generate architectural representations of organizations from the information gathered in projects plans and other information sources.

EC 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 EA because it focuses on producing representations based on observations, so it does not include the purposeful design, as one expects in EA [4]. 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.

2.2.1 Enterprise Architecture Tools

There are several tools in the enterprise architecture domain, such as Enterprise Architect by Sparx Systems, Innoslate by SPEC Innovations, IRIS Business Architect by Benchmark Consulting, Abacus by Avolution, BiZZdesign Enterprise Studio by BiZZdesign and HOPEX by MEGA. The common features that the tools referenced above have are the creation of diagrams, modeling and simulation, transformation plan and project management. Although one of their resources is project management, they do not have the project management as the source of changes in the organization, that is, they do not use information from project plans to model or change the existing architectural models. For example, with the HOPEX tool one can get a clear view of applications that become obsolete over the years so one can better plan the transformation of the organization’s IT landscape, but it is not possible to use the deliverables achieved by projects to keep the architectural representations updated. Most of the tools offer integration with TOGAF, ArchiMate notation and Unified Modeling Language (UML) and Business Process Model and Notation (BPMN) standards, but not with a tool from the project management domain. Despite their advantages and disadvantages, this thesis was done in collaboration with the company Link Consulting, so the tool used in the enterprise architecture domain will be the one adopted by this company, which is Atlas.

Atlas is an enterprise architecture tool that was developed to support an efficient management in enterprise architectures, where any (client) organization can configure its contents and analyze it through different types of customizable visualizations, named architectural views or models. An architectural view is a graphical representation of the organization architecture from a given point of view and at a given moment in time [14].

This tool was also conceived to minimize the effort needed to keep architectural models updated in enterprises where changes are constant, and design occurs in multiple tools [15]. In Atlas, architectural views are generated automatically based on the queries elaborated and executed, and have a time bar allowing an easy navigation in time, since an important and complex problem regarding architectural
models is that they can represent different points in time. There are models to represent the past state of the enterprise (AS-WAS models), models to represent the current enterprise (AS-IS models) and models to represent the future state of the enterprise (TO-BE models). Atlas was developed to support those different states by providing visualizations in which the information about changes is gathered by the observation of the plans of ongoing change initiatives. Change initiatives correspond to the concept of projects, since they are also defined as temporary, unique and a purposeful activity [15].

The most common views present in Atlas are blueprints, which are ways of communicating between people, specifically to express an architectural description of an object, a system, a model or an enterprise for example. Blueprint is the set of all architectural views from a given point of view over all possible moments in time (past, present and future), making architecture artefacts having a life cycle [1]. In this tool, all blueprints have a time slider that allows the visualization in any point in time, and in each position of the time slider the blueprint shows the architecture artefacts in the state corresponding to that point in time. It is also possible to see the relations between the artefacts. According to the metamodel chosen by the organization it’s possible to visualize the relations (for example: realizes, uses, composes, etc.) among the different enterprise artefacts.

Blueprint’s visualization is possible through the configuration and execution of queries and batches. They are made in a language based on tags, so-called Extensible Markup Language (XML). A query is a set of hierarchical clauses that produces a result set of data elements that can be stored using assignments, and later can be used as variables. A query can be used independently, or can be embedded in either blueprints or chart specifications. They are going to be used to develop the architectural representations so that the enterprise cartographer can observe and evaluate the impacts between a certain project and its artefacts, after the models have been automatically drawn.

Despite the obvious benefits of enterprise architecture in the planning and management of the organization’s transformation initiatives, the enterprise architecture domain is still a remote reality for several organizations. [16] In fact, they still do not have current and updated representations of their architecture, but they all need them to plan the next transformations initiatives, which can be achieved with the help of this enterprise architecture tool - Atlas.

2.3 Link between PM and EA

Despite the fact that integration of enterprise architecture and project management tools is still scarce, there are already several articles and information that link these two domains theoretically, presenting the similarities and benefits that would be achieved if an organizations did so.

There is already a master’s thesis [17] that explains the relationship that exists between Application Portfolio Management (APM) and EA, and how APM is used in EA and vice versa. Application portfolio
management is considered as the discipline to assess individual and application portfolios to justify their value, in order to make better decisions regarding the future state. This discipline is closely related to EA, since EA can be used to unveil the relations between business processes, applications and underlying infrastructure. Moreover, EA can be used to effectively detect misalignments and redundancies in the application landscape. In this thesis [17] it is noted that EA is a key skill and support area for IT portfolio management, being the application portfolio management a sub-part of the previous one. In the end, it is concluded that APM is strongly influenced by EA, and becomes increasingly visible in EA and vice versa [17].

More directly with project management, we have articles like [2], where it is said that the prime goal of PPM is to manage projects and programs to improve performance. While Project Management Office (PMO) teams (which are the people responsible for setting and maintaining project management standards within the company to optimize the control and execution of proposals within the organization as a whole or within a specific area) may manage a portfolio of projects and the resources that are used to deliver them, EA teams have a different focus when it comes to portfolio management. They are mostly concerned with APM because it is seen as a clean problem that the business understands and that lends itself to potential cost savings, what resembles to [17]. It is explained that EA provides PMO with strategic alignment, and PMO provides EA with effective project execution, being these two essential for the success of an organization.

We also have two other articles that link functions and objectives between IT PMO and EA. They clarify the differences between roles, their responsibilities, the artefacts used, as well as the similarities and benefits of using both in a complementary way [18,19].

What we can draw from this is that there are already several organizations dedicated on trying to keep PM and EA domains close when planning and executing their daily work tasks. There is already a concern about communicating information about IT projects to the people responsible for maintaining the company blueprints updated [20], but there is no automatic way to update these blueprints directly through the information coming from IT projects. There is still no tool or method for spreading information that has been proven to have dependencies (projects and enterprise architecture) in a direct and automatic way, which is what we intend to develop in this thesis. It will be possible to assess the dependencies and impacts that exist within a project portfolio, by detecting the incongruities that occur in certain projects, as it is virtually impossible to have a global view of everything that happens within an organization with numerous transformation initiatives (projects) occurring at the same time, and managed by different people and departments.
3

Solution Proposal
The goal of this chapter is to present a solution for the identified problems, what will lead to answer the three questions presented in chapter 1. The first question (“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?”) will be explained in section 3.1. The second question (“How should the architectural models and views be designed so the dependencies of the project can be analysed and evaluated both by the project manager and the enterprise cartographer?”) will be clarified in both sections 3.2 and 3.3, and finally the third question (“How should project management and enterprise architecture tools be integrated and implemented?”) will be answered in section 3.4. In a general way the process will be as follows. It specifies the activities needed to be done in order to reach the solution, and for an easy understanding it was made with a BPMN tool named Bizagi Modeler, that is focused on process modeling.

3.1 Language

As already mentioned, projects are transformation elements of the organization, and their achievements drive the progress of the enterprise architecture [3]. Thus, enterprise changes are caused by transformation initiatives within projects, so it will be necessary to understand what information from Microsoft Project, and more specifically from project plans, will impact the architecture of the organization.

The deliverables successfully achieved by an organization, and that consequently implement changes on its architecture, are present in the milestones of a project. Milestones are used to mark a specific point along a project timeline, which identifies the deliverables successfully achieved by the organization, after the tasks have been reviewed. Therefore, the information that is present in the milestones of project plans provide to the architecture of organizations the elements that will influence and change them, and consequently, that will be vital to preserve architectural representations updated.

For instance, in an IT project, milestones such as “Oracle installed” and “CRM implemented” will be ordinary, but to what architectural artefacts do those elements correspond? It is important to define a set of rules to be followed when integrating project management and enterprise architecture domains, so the information transmitted can be coherent. Those rules will be evolved throughout this chapter.

From the point of view of the project management domain, there is a large amount of data we con-
sider relevant to extract and integrate with the enterprise architecture domain. That data corresponds to the artefact name (element depicted in the milestone), type of artefact (class of the element), action (impact suffered by the element) and temporal milestones (dates when the impact occurs). These elements are essential since project completions are the drivers for maintaining the architectural representations updated. Projects are transformation elements of an organization, so their achievements drive the progress of the architecture, and consequently of its models.

3.1.1 What types of artefacts?

Especially during the early phases of a transformation project, it is often found that one source of misunderstand and confusion is that every stakeholder uses different notions to refer to similar concepts. Enterprise architecture metamodels are used to overcome this problem, in order to reach an agreement about which concepts are relevant to perform those tasks, and how to name them.

Metamodels are a core concept of EA, which describe the fundamental artefacts of business and IT (meta-types), the rules among them (associations or relations), and the attributes that describe and distinguish them. They are used by end users to express models, which means that they provide the framework within which the model can be constructed, so that all models can be produced consistently and coherently.

In this way, metamodels enforce semantic rigor among the models subsequently created in their shadow. Such rigor is a precondition for successful communication and documentation. Without a metamodel, the whole management of an enterprise architecture becomes extremely arduous. A metamodel provide not only a common language to talk about an enterprise at several levels between all the stakeholders of an organization, but also a clear view of the structure and dependencies between relevant parts (EA artefacts) of the organization.

It is also relevant to notice that just as different companies have different business objectives, also their concepts and relationships vary from each other. This circumstance makes the implementation domain broad, since there is no universal metamodel to be based on. This dissertation will support the ArchiMate metamodel since it was designed to be as compact as possible, but still usable for most corporate architecture modeling tasks. Even though some organizations do some adjustments to the ArchiMate metamodel, it will be the one tolerated in this thesis (see Fig. 2.3).

3.1.2 What types of impacts?

After knowing what elements from project plans change and have impact on the architecture of an enterprise, which corresponds to the EA Artefacts from the ArchiMate metamodel depicted in the project’s milestones, we must explore how they will be impacted by projects. It is important to understand what
will be the associations and dependencies between projects and the corresponding impacted artefacts.

The first terms to arise were the 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 those deleted artefacts become as if invisible to the organization. In spite of being no longer part of the company’s operations, a representation of them continues to exist 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 it does not 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 getting the information without actually changing it. But what if an organization wants to change some artefact?

For this reason, another term was created. It specifies the modifications that occur to an artefact, and the keyword for this type of impact is CHANGE. After a bit of research, important facts for this development were found, which led to its disappearance. According to ArchiMate, an artefact does not change itself since it doesn’t have inner properties; what does change are the relations that it presents with its constituents, i.e., with its components. In the architecture domain, its said that nodes have other nodes, being nodes the existing artefacts [21]. A node cannot change, only the relation between parent node and child node, as we can observe in Fig. 3.2.

![ArchiMate relations between artefacts.](image)

**Figure 3.2:** ArchiMate relations between artefacts.

The left side of Fig. 3.2 provides a standard visualization used when designing a system in the field of software engineering, for example in UML. Objects are represented in boxes, which may contain data (attributes), and additionally, they can be linked to other objects through arrows, that typify the existing relations.
On the other hand, we have the right side of Fig. 3.2, which reflects artefacts in ArchiMate. Similarly to the other side of the figure, artefacts are also represented in boxes, but they do not have their own properties. If we want to state that a given artefact, such as an ‘Application Component’, has a derived relationship with two components, then two arrows must leave from the application component artefact to two new artefacts (‘Component 1’ and ‘Component 2’). Each artefact represents a different component. Hereupon, CHANGE impact serves as an encapsulation, since whenever someone wants to modify from ‘Component 1’ to ‘Component 2’, it is not necessary to change the ‘Application Component’ artefact itself. It is only necessary to eliminate the relation between this artefact and ‘Component 1’, and afterwards create the relation with ‘Component 2’. Thus, it is easily realized that the CHANGE impact is the combination of CREATE and DELETE impacts, according to the information descendant from ArchiMate, which led to its disappearance.

The prior types of impacts were based on the four basic functions of persistent storage, being those the CRUD permissions, also known as Create, Read, Update and Delete. There are other variations of CRUD [22], such as BREAD (meaning Browse, Read, Edit, Add and Delete), DAVE (meaning Delete, Add, View and Edit) and CRAP (meaning Create, Replicate, Append and Process), but they do not apply so well to our requirements, so they have been left out. For instance, BREAD and DAVE are often used to describe user interface conventions that facilitate viewing, searching and changing information, which is not suitable for this work.

3.1.3 Limitations of impacts

To conclude the types of action that will be possible, another important point to consider is the difference between the domains of the EA artefacts, 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, it can be assigned to the logical domain any artefact of construction, like application components or business objects, 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, which is respectively the date where the artefact becomes available to be used by the internal operations of the organization (and starts to exist a representation of it in the
architecture), and the date where the artefact should no longer be used by the internal organization’s operations, but continues to exist a representation of it in the architecture.

On the other hand, the previous types of impacts (CREATE and DELETE) won’t make sense 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 given physical system, that should exist primarily in the architecture. Therefore, the impact types that fit these physical artefacts are the DEPLOY and UNDEPLOY (which correspond respectively to the logical impacts CREATE and DELETE). The former corresponds to the moment when the respective installation version will become available for the company to use, and the latter 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), the previous/old version is also uninstalled, 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 a 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 any other physical system.

For instance, this type of impact will enable actions like upgrade from Java 7 (in node x) to Java 8 (in node y). The unnecessary repetition of impacts is mitigated because it is not necessary to represent four milestones that reflect the same action in a simplified way: the deployment of Java 8, the deployment of node y, the undeployment of Java 7 and the undeployment of node x. With the UPGRADE impact, there would only be necessary one milestone describing the following action: upgrade from Java 7 in node x to Java 8 in node y, which is a lot simpler.

Briefly, logical artefacts might be created, which results in a representation of that object starting to exist, used by a certain project, that is, analyzed in order to extract the necessary information from them without being actually modified, 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, which were already described and explained above.

### 3.1.4 Language rules

In order to be able to integrate the two technological components directly related to this thesis, Atlas and MS-Project, it is essential to create a set of semantic rules to have a defined language that we can always apply. 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 section, 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 has to be extracted from the milestones of the project plan, since these are the points in time that have the architectural transformations that will occur in the organization. The changes that a project promises to make, because the plan is just a promise and isn’t effective when it comes to its completions, will be loaded into Atlas. Afterwards, Atlas will update the TO-BE (or emerging AS-IS) state of the organization’s architecture, keeping this way the architectural views properly updated, based on the following imported information:

<table>
<thead>
<tr>
<th>TYPE OF IMPACT</th>
<th>TYPE OF ARTEFACT</th>
<th>ARTEFACT</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE</td>
<td>class_a</td>
<td>a</td>
<td>d_a</td>
</tr>
<tr>
<td>DELETE</td>
<td>class_b</td>
<td>b!={}</td>
<td>d_b</td>
</tr>
<tr>
<td>USE</td>
<td>class_c</td>
<td>c!={}</td>
<td>d_c</td>
</tr>
<tr>
<td>DEPLOY</td>
<td>class_d</td>
<td>d</td>
<td>d_d</td>
</tr>
<tr>
<td>UNDEPLOY</td>
<td>class_e</td>
<td>e!={}</td>
<td>d_e</td>
</tr>
<tr>
<td>UPGRADE</td>
<td>class_f</td>
<td>f!={}</td>
<td>d_f</td>
</tr>
</tbody>
</table>

Table 3.1: Summary of the language concepts.

From the table above we can observe the elements by which our language will be composed. We will have the name of the artefact that is undergoing changes to a particular project in the organization’s portfolio, and this field is called ‘ARTEFACT’, the date when that change occurs in the field ‘DATE’, the type of artefact that is being impacted by the project, which corresponds to the class of the metamodel of ArchiMate that artefact falls into, and finally the kind of impact the project is having on the corresponding artefact, which can be CREATE, DELETE, and USE for the artefacts of the logical domain, and USE, DEPLOY, UNDEPLOY and UPGRADE for the artefacts of the physical domain.

Regarding the ‘ARTEFACT’ field, it is important to notice a symbol denotation that demonstrates that a given logical artefact cannot be deleted or used without having been previously created, and the same is true for physical artefacts. These cannot be used, uninstalled or upgraded without having been previously installed.

### 3.2 Impact visualization

In the Language section, it was 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 answer the second question of the Problem section, being that “How should the architectural models and views be designed so the project’s dependencies can be analyzed and evaluated both by the project manager and the enterprise cartographer?”. For this effect we will use blueprints, since they exist to show architecture elements in the state corresponding to that point in time, as defined before in chapter 1. Although we take it into account, the
focus is not on the aesthetics and framing of the different elements on the screen, but on the ease of interpreting the blueprint and on the amount of knowledge that is transmitted through the architectural models demonstrated.

Hereupon, the essential elements that need to be displayed so the enterprise cartographer and the project manager can easily identify and analyze the dependencies between the elements from the project management and the enterprise architecture domains consist in the actual project and the impacted artefacts. The actual project corresponds to the project that was extracted from the project management tool used as proof of concept, and has to be chosen from the project portfolio of the organization (Fig. 3.3), so the blueprint can be drawn with the details of that specific project. This actual project corresponds to the project that will undergo evaluation and analysis by the responsible persons within the organization, so the corresponding impacts can be identified and mitigated after such analysis.

![Figure 3.3: Project selection in Atlas.](image-url)
Additionally, the impacted artefacts are the ones used to represent the project deliverables that yield changes in the architecture, and they depict the EA changes that occur in our project, that corresponds to the actual project. They also denote the artefacts with which our project has any type of dependency, that is, artefacts that suffered impacts caused by this project, such as having been created, undeployed or used by it, for example. As already addressed, those artefacts can be split into logical and physical artefacts, since the types of impacts vary between them, so the blueprint resulted in the following Fig. 3.4.

![Blueprint showing the project impact detailed.](image)

**Figure 3.4:** Blueprint showing the project impact detailed.

1. **Project and Work-packages**

   This segment (1) exists not only to allow the visualization of the project that will be analysed but also to visualize all the projects it aggregates, that is, its sub-projects. Hereafter they will be called as ‘our project’, and they will only be shown on the blueprint if they exist, because otherwise that container won’t appear on the screen. 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 coincides with the project definition itself. [21]

2. **Artefacts**

   The EA artefacts that undergo any kind of action/impact caused by our project are present in this segment (2). 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 (logical or physical) suffer from our project. For instance, it will exist a container to display the logical artefacts created by segment 1, another for the ones deleted by it, another for the physical artefacts deployed by segment 1, and so on, as we can see in Fig. 3.4. Additionally, it can be observed that not all types
of impacts are represented in the blueprint, such as the UPGRADE. This impact was suppressed since it does not provide any valuable information in a visual way. We don’t need to know in which project occurred the upgrade of an artefact, we just need to know which version was deployed and undeployed. It is important to get concrete and objective information in the blueprint, so it can be feasible and easy to interpret. Therefore, in this specific case, we are going to use the extended language (the DEPLOY and UNDEPLOY impacts separately) and not the simplified one (UPGRADE), because they provide us information that is not only more valuable but also more easily analyzed visually.

3. Related Projects

The related projects segment (3) represents all the projects associated, i.e., that have any type of impact (CREATE, USE, DEPLOY, DELETE and/or UNDEPLOY) with at least one artefact from the Artefacts segment (2), that is, have any type of relation with an artefact that is created, used, deployed, deleted and/or undeployed by segment 1, more specifically, by our project.
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 (segments 1 and 3) of an organization, i.e., its project portfolio. 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, after applying some rules that will be described in the next subsection.
3.3 Alerts

For the analysis of the dependencies between projects and the EA artefacts they have an impact on, we developed a feature of alerts that is divided into two main groups of alerts: the orange and the red one. The former represents situations where the dependency between project and artefact are merely warnings, and where the probability of failure exists (it is not mandatory). The latter represents all the critical cases, i.e., the cases where the relation between the project and its artefact(s) cannot exist because it is problematic and incongruous. In this situation there is no chance of success.

The first attempt to implement this feature of alerts consisted in creating sets of three elements: the name of the project being analyzed (the one extracted from MS-Project), the name of the artefact being impacted by several projects, and the name of the second project causing the impact on that artefact (the related project). This would enable a visual ease when realizing which projects were having conflicts with a particular artefact, because the three elements would be designed in the same lane of the container, as observed in Fig. 3.5.

![Figure 3.5: Desired arrangement of elements in the container.](image)

However, the blueprints generation supported by Atlas does not allow the design of a certain artefact more than once in the same container, that is, in the same blueprint box or segment. Therefore, whenever a related project created two different artefacts, that related project would only be designed once in the container, which would have a negative impact on the structure of the container. The sets of three elements would be unbalanced whenever one or more elements was repeated, because they wouldn’t be drawn and their visual perception would be destroyed, becoming much more difficult to visualize, and consequently useless. In the two figures below, we can notice an example of the arrangement of elements in the container if there was no restriction imposed by Atlas, in the left figure (Fig. 3.6), and the same example taking into account the Atlas implementation rules, in the right figure (Fig. 3.7).

To counter this impasse, another solution was reached. It consisted in creating 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 relationship our project (segment 1) has with the artefact, and
the second impact refers to the relationship a related project (segment 3) has with that same artefact. Those relations will be drawn only if they represent warnings (orange alerts) or critical situations (red alerts).

As we can see in the blueprint of Fig. 3.4, the project for which its dependencies are being evaluated appears in segment 1 and it’s always the same, since it is picked up before the generation of the blueprint (Fig. 3.3), so we can suppress it from the error containers. For example, if an error situation is produced because our project (A) is deleting an artefact (x) that is being created by another project (related project B), the container would be called ‘deletion-creation’. Both artefact x and related project B will be displayed in that container, with the representation of an arrow from B to x, to identify that B is creating artefact x (see an example in appendix 1). The relation between B and x will be known to be of creation since the container’s name will be ‘deletion-creation’, and the second impact corresponds to that specific relation. Furthermore, our project A won’t be displayed, as already explained, but the relation between that project and artefact x is easily perceivable. It corresponds to the first impact of the container’s name (‘deletion-creation’), which means that project A would be deleting artefact x.

The rules regarding the language to use and the representation of the different elements in the blueprint were already set, so which alerts will be classified as orange or red? Regarding the orange group, the following occurrences might happen:

**Creation-Deletion:**

This alert is produced whenever our project, which corresponds to the one extracted from MS-Project (segment 1 of Fig. 3.4), creates an artefact that is deleted by another existing project in the organization (segment 3 Fig. 3.4). An error can emerge when a project deletes an artefact that hasn’t been previously created. The artefact production and deprecation dates must be evaluated, since this situation is problematic only when the deprecation date is earlier than the production date of the artefact. These dates are part of the artefact life cycle, and they are properties inherent of objects in Atlas. An artefact cannot
be created after it has been deleted because its representation is maintained in the architecture, and consequently in Atlas repository. Despite that it cannot be used by any organization’s operations, there is still a reference for that object, and two artefacts with the same name cannot exist, and cannot be created twice. When we refer to the creation and deletion of an artefact, we are referring to both logical and physical artefacts, so the impacts referenced here in the creation are the CREATE and DEPLOY impacts, and in the deletion the DELETE and UNDEPLOY impacts, which will be applied from now on in every case.

**Deletion-Creation:**

This alert is similar to the one above, but varies on the project that creates and deletes. Contrary to the above, in which the creator of the artefact is our project (segment 1 of Fig. 3.4) and the related project the one that deletes the artefact (segment 3), in this type of alert the opposite occurs. The one that creates the artefact is a related project (segment 3), and the one that deletes the artefact is our project (segment 1). An error can emerge when a project deletes an artefact that hasn’t been previously created in the organization, so in this alert it is also necessary to evaluate the production and deprecation dates of the artefact, because this error will only be produced whenever the artefact is deleted before it has been created.

**Creation-Using:**

This alert is produced whenever our project (segment 1) creates an artefact that is being used by another existing project in the organization (segment 3). It is noticeable that an error will only be generated in the situation of an artefact being used by a related project and that has not been previously created by our project. When an artefact doesn’t exist in the organization, it cannot be accessed by any organization’s operations, so the using and the creation dates of the artefact must be checked. Whenever the using date is earlier than the creation date, this type of alert will be generated and displayed in this specific container of the blueprint.

**Using-Creation:**

This alert is very identical to the one described above, but in this one is our project that uses the artefact and a related project that creates it. Here, also the using and creation dates of the impacted artefact have to be evaluated, to check if a warning will be produced or not. This will happen every time the related project creates the artefact later than our project uses it.

**Deletion-Using:**

This alert is generated when our project (segment 1) deletes an artefact that is being used by an-
other existing project in the organization (segment 3). An error will be reproduced in the situation of an artefact being used by a related project, being that this artefact is no longer available in the operations of the organization, since it was formerly deleted by our project. Therefore, both deletion and using dates must be analyzed in order to verify if those restrictions apply.

Using-Deletion:

Similarly to the alert explained previously, the using date and the deletion date of the artefact being impacted by several projects should be evaluated to check if they are invalid. If so, it means our project is using an artefact after it has been deleted by a related project, which will trigger the alert.

As we can observe, this orange type of alerts has a great dependency with dates. Since warnings represent situations that might be problematic, it is fundamental to verify the variables of it. If those variables satisfy the requirements to be considered valid, the elements (projects and artefacts) will not be drawn in the error containers. Otherwise, they will, because the relations between projects and artefacts are invalid, and accordingly impossible in a business context.

On the other hand, there is the red group, that have the following situations that might occur wrongly:

Creation:

This alert is generated when different projects create the same artefact. This cannot happen because when a project creates a certain artefact, even if another project creates an artefact with the same name, the object is not the same. It is necessarily a different one. The rule is that whenever the representation of an artefact exists in the architecture, it is not possible to create it ever again.

Having in consideration implementation aspects, the artefact creation date is updated with the value of this date obtained during the project plan, and it corresponds to the productive date, which is the date when the artefact becomes part of the organization operations. Since in an organization it is more than usual that several projects occur at the same time, which is why the concept of project portfolio emerged, the stakeholders involved still don’t have a structured and organized way of communicating with each other. This results in the elaboration of project plans that have contradictions and nonsense dependencies between the impacted EA artefacts and the corresponding projects. Hereupon, numerous projects may be creating the same artefact, unnecessarily spending time, money and other organization resources. Situations like these should be prevented from happening in real projects, that is, during the execution of project plans, by applying a proper control and monitoring.

In the simultaneous creation of an artefact by multiple projects, the final creation date will correspond to the date that is in the last project plan propagated/imported from Atlas. The data kept in the repository will be the last one being imported according to the destructive update mode of importation practiced by
Atlas, that says the common data between the repository and the file to be imported remains the same as the one in the file. This is not correct from a structural and organizational point of view, since the last project plan does not necessarily contain the best date for the creation of the artefact. However, such constraint doesn’t affect the efficiency of the generation of blueprints nor the generation of errors related to the dependencies between projects and its artefacts. The creation dates are not relevant for this type of alerts since the crucial thing to know is that an artefact is created by more than one project, which is effectively propagated from MS-Project to Atlas. The error exists and will be generated regardless of the value associated to those dates.

**Deletion:**

This alert is generated when different projects delete the same artefact. This is an impossible situation since when a project deletes a certain artefact, it stops being available in the organization’s operations, although there is still a representation of it in the architecture. So, it cannot be deleted by another project after that, which brings us to the rule that refers that whenever an artefact is deleted by a project, it cannot be deleted anymore.

Similarly to what was said in the creation type of alert, the artefact deprecation date is also updated with the date value obtained during the project plan, which corresponds to the deprecation date. It is the date when the artefact can no longer be used by the organization’s operations. As several projects can delete the same artefact due to the lack of communication among the organization’s various stakeholders, with regard to implementation details we have that the deprecation date that will be propagated from project plans to the architecture will correspond to the date that is in the last project plan imported from Atlas. However, this does not imply that the saved date is the best deprecation date for that artefact. Nevertheless, the efficiency of the generation of blueprints, and consequently of errors, won’t be affected. The crux of this process is to produce alerts for the existing errors between different projects of an organization and their impacted artefacts, so that they can be identified, communicated and resolved by the stakeholders before the execution of the project itself, so the setbacks of the organization can be minimized. To be able to identify this type of alert, it is crucial to know that an artefact is deleted by more than one project, information that is effectively propagated from MS-Project to Atlas, but it is not essential to know those specific deprecation dates. The error exists and will be generated regardless of the value associated to those dates.

The previous error situations are related with the CREATE/DEPLOY, DELETE/UNDEPLOY and USE impacts, but what about the UPGRADE? This type of impact is a simplification of the deployment and undeployment impacts, as mentioned hitherto, since an EA artefact cannot be changed itself, it can only stop having relation(s) with some component(s) to start having relation(s) with other(s). Therefore,
the artefacts that suffer an UPGRADE by some project are contained indirectly in the DEPLOY and UNDEPLOY impacts, because it means that an artefact was undeployed to give rise to another one, that was consequently deployed.

Additionally, it doesn’t make sense to display the UPGRADE impact because it is not self explained. We need to have the concrete information about what is being deployed and undeployed in the organization, to be able to identify properly the existing dependencies. So for this factual purpose, the simplification made for the DEPLOY and UNDEPLOY, i.e., the UPGRADE impact, is not advantageous and won’t be used when designing and implementing the error containers of the blueprint. It will be used to fill the project plan and to be exported from MS-Project.

In summary, we can have the following possible combinations of pairs of impacts between projects and artefacts, and the respective situations where they actually cause error:

<table>
<thead>
<tr>
<th>ACTUAL PROJECT</th>
<th>RELATED PROJECT</th>
<th>CONDITION</th>
<th>TYPE OF ALERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>creates</td>
<td>creates</td>
<td>Situation always invalid</td>
<td>Critic</td>
</tr>
<tr>
<td>creates</td>
<td>uses</td>
<td>Error when <em>c.date</em> postdates <em>us.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>creates</td>
<td>deletes</td>
<td>Error when <em>c.date</em> postdates <em>d.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>uses</td>
<td>creates</td>
<td>Error when <em>u.date</em> predates <em>c.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>uses</td>
<td>uses</td>
<td>Situation always valid</td>
<td>NA</td>
</tr>
<tr>
<td>uses</td>
<td>deletes</td>
<td>Error when <em>us.date</em> postdates <em>d.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>deletes</td>
<td>creates</td>
<td>Error when <em>d.date</em> predates <em>c.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>deletes</td>
<td>uses</td>
<td>Error when <em>d.date</em> predates <em>us.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>deletes</td>
<td>deletes</td>
<td>Situation always invalid</td>
<td>Critic</td>
</tr>
</tbody>
</table>

Table 3.2: Possible combinations for logical artefacts.

<table>
<thead>
<tr>
<th>ACTUAL PROJECT</th>
<th>RELATED PROJECT</th>
<th>CONDITION</th>
<th>TYPE OF ALERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploys</td>
<td>deploys</td>
<td>Situation always invalid</td>
<td>Critic</td>
</tr>
<tr>
<td>deploys</td>
<td>uses</td>
<td>Error when <em>de.date</em> postdates <em>u.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>deploys</td>
<td>upgrades</td>
<td>Error when <em>de.date</em> postdates <em>un.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>uses</td>
<td>deploys</td>
<td>Error when <em>u.date</em> predates <em>de.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>uses</td>
<td>uses</td>
<td>Situation always valid</td>
<td>NA</td>
</tr>
<tr>
<td>uses</td>
<td>upgrades</td>
<td>Situation non-sense</td>
<td>NA</td>
</tr>
<tr>
<td>uses</td>
<td>undeploys</td>
<td>Error when <em>u.date</em> predates <em>un.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>upgrades</td>
<td>deploys</td>
<td>Situation non-sense</td>
<td>NA</td>
</tr>
<tr>
<td>upgrades</td>
<td>uses</td>
<td>Situation non-sense</td>
<td>NA</td>
</tr>
<tr>
<td>upgrades</td>
<td>upgrades</td>
<td>Situation non-sense</td>
<td>NA</td>
</tr>
<tr>
<td>undeploys</td>
<td>deploys</td>
<td>Error when <em>un.date</em> predates <em>de.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>undeploys</td>
<td>uses</td>
<td>Error when <em>un.date</em> predates <em>u.date</em></td>
<td>Warning</td>
</tr>
<tr>
<td>undeploys</td>
<td>upgrades</td>
<td>Situation non-sense</td>
<td>NA</td>
</tr>
<tr>
<td>undeploys</td>
<td>undeploys</td>
<td>Situation always invalid</td>
<td>Critic</td>
</tr>
</tbody>
</table>

Table 3.3: Possible combinations for physical artefacts.
3.3.1 How will it be generated?

Every element of the blueprint is displayed after running a set of queries made for that purpose. Those queries have information regarding the different types of artefacts, and their particular properties, such as dates and relations with other artefacts. The following query demonstrates a simple example:

```xml
<IQD Name="WP" Description="Workpackages">
  <ARG Index="1" />
  <BLOCK>
    <PROPERTY>Aggregates</PROPERTY>
    <BLOCK>
      <WHERE>
        <PROPERTY>Name</PROPERTY>
        <OPERATOR>!=</OPERATOR>
        <VALUE><ARG Index="1"/></VALUE>
      </WHERE>
      <AS>WP</AS>
    </BLOCK>
  </BLOCK>
</IQD>
```

The previous query is responsible for getting the work packages container of segment 1 (Fig. 3.4), that is, all the sub-projects of the project we are analysing. The reference element of this query is the first argument (Arg 1), which is the project picked in Atlas to be displayed in the blueprint in order to observe more detailed information about its dependencies with other projects and artefacts (as seen in Fig. 3.3). From this element, we go through the property ‘aggregates’, which is a property present in Atlas inherent to this type of elements - projects -, for all the elements that have a name different from the one given as argument 1. Since the ‘aggregates’ property is restricted to the ‘work package’ type of artefacts, that represents transformation elements like projects according to ArchiMate, this query will give us all the projects (sub-projects) that are aggregated by the actual project given as argument 1 in the blueprint. This group of elements will be saved in a variable named ‘WP’ so we can get those elements later, for example, when printing the container elements of the blueprint. All the other queries were made following this logic, but most of them were more complex, and structurally larger, so they will not be shown in this document.
3.4 Macros

In order to be able to integrate the project management tool we are using as proof of concept – Microsoft Project – and the enterprise architecture solution adopted by Link Consulting – Atlas – to keep the architectural representations and the project plans properly updated with the information descendant from each other, we used a well-known feature of Microsoft Office named macro. This feature exists in several Microsoft tools, one of them being MS-Project, so we will use them to our advantage.

3.4.1 What is a Macro?

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 [23].

A very important conceptual difference is that the programming language normally used by the macros is Turing complete, whereas spreadsheets, such as Excel, are not. In computability theory, a system of data-manipulation rules, like a programming language, is said to be Turing complete if it can be used to simulate any Turing machine, i.e., if rules followed in sequence over arbitrary data can produce the result of any calculation [24]. This cannot be done in the normal mode. A practical example that proves it is that it is only possible to write loops with an indeterminate number of steps with macros, which is one of the reasons why we are going to use them.

Briefly, a macro is a series of commands and instructions that are grouped together as a single command to accomplish repetitive tasks in an automatic and efficient way. Although Microsoft Office provides numerous built-in spreadsheet formulas/functions such as text, logical, math and statistical functions, they are limited when it comes to their capabilities, because they are calculation specific and can only process data. This means they are formatted to receive an input value, the data is processed and another value (output) is returned, but what we’ll want to do is to transform the information descendant from MS-Project project plans, so that it can be later imported from Atlas. This import implies that the information is received in a certain template and format that must be supported by Atlas. Therefore, it is through the implementation of macros that we will enable the automation of this process, to allow the update of the architecture models as the project plans change, and vice-versa.

3.4.2 How is it going to work?

So the solution could be reached, different macros for the different defined objectives were created, some of which are located in the project management tool chosen, and others in a support auxiliary
tool. This auxiliary tool will be the connecting link to do the integration between the two technology components directly related to this dissertation, being those Atlas and MS-Project.

Macros will automate tasks that could be manually done with more effort and time, but instead, they will be performed automatically so the architectural representations from Atlas can be maintained updated based on the information from project plans, and vice-versa, with the least possible effort. This approach will also save time to the ones using the previous tools, namely enterprise cartographers and project managers, since whenever the architectural models change within an organization, that information can be spread to its corresponding project plan, and the opposite as well. Whenever information from project plans are updated, they will be automatically propagated to Atlas, so the architectural representations can be drawn correctly and kept updated.

Hereupon, were developed macros in MS-Project to enable the transfer of information from project plans to a file external to this tool, and that could be later exported to Atlas. When having in account the file formats that MS-Project export (and import), we also have to consider the other side of the pipe, which are the file formats that Atlas imports (and exports) respectively. There were two possible solutions for this problem:

- Generate and manipulate files in XML markup language

  The main purpose of XML is the ease of sharing information through the internet (web pages). It is used to exchange structured data between different applications/tools, so that is why this markup language can be verbose, complex and redundant. The large amount of repeated information impairs the speed of information transfer, which is a negative point.

- Generate and manipulate files in Excel format

  This file format supports easy and effective comparisons because of the strong analytical tools included on it. Contrary to XML, Microsoft Excel enhances the ability to analyze large amounts of data by allowing powerful analysis techniques, which is a plus.

XML and Excel are the only file formats that Atlas supports both when importing and exporting data. This restriction limited the several existing choices we had to perform the transfer of information, but based on the previous explanations, we chose the Excel format. XML is a markup language that might be a bit more difficult to read due to its tags, and Excel has the information structured in a more visual way, being that its lines and columns are well identified in the file, which made the implementation effort smoother.

Atlas keeps the artefacts information divided in classes. Each class from the metamodel adopted by the organization has a distinct file, that contains a list with every artefact (identified by its name), in the different rows of the first column, and the several property names of that class in the first row of the
following columns, as we can see in the example of Fig. 3.8.

| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| CRM | Aggregate | Application Component | Application Layer | Begin Date | Build Cost | Business Source | Created by | Created by | Description | End Date | External Link | Intern Version | Level | OPA | |
| DRF | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |

**Figure 3.8:** File template of the Application Component class, exported from Atlas.

For instance, the value from the cell (2,10) of the application component file template represents the value that an application component named ‘CRM’ has for the property ‘Created by’. That cell depicts the name of the project or projects that create the ‘CRM’ artefact, which in this specific case is just one, the ‘ProjectPlan’ project.

The first aspect that must be taken into account in this integration is the consistency between the names of the artefacts used by the project manager and the cartographer. Since one of elements belonging to our language is the artefact name, we must assure they are concordant between the ones used in Atlas and MS-Project, to enable the correct propagation of information. Therefore, to prevent business actors from using different names to specify the same elements, a list of all artefacts within the organization’s architecture will have to be exported from Atlas, in the Excel file format. The steps that have to be performed are shown in the following figures.

**Figure 3.9:** How to export artefact names: step 1.

First, one must go to the ‘Data Explorer’ menu and click on the icon symbolizing the download. It is in the upper right corner highlighted in orange since it is being hovered, as we can see in the figure above (Fig. 3.9).
Then, the 'All classes' option must be selected (with a check), as shown in Fig. 3.10, and the 'Export mode' has to be in the Excel format, as it has already been explained. After following these procedures, a zipped folder is downloaded into the computer where this download was done. That folder contains the classes of all artefacts present in the metamodel chosen by the organization in question, which is going to be important not only to prevent wrong names but also for the rest of the process.

Having a list with the artefact names that can be used by the project manager when planning its projects is convenient and appropriate when that list exists in the project management tool used. Those artefacts must be copied in the beginning of each (new) project plan to the lookup table of the 'Customizable Field' created for this purpose in MS-Project, that will be named 'Artefact1'.

Every time a task (milestone) is originated, the project manager will be able to select in the 'Artefact1' column of MS-Project the deliverable (EA artefact) from the preexisting list that will suffer an impact from that project. Instead of writing the name of the element being impacted, project managers will be able to select the element from an existing list (Fig. 3.11), which is much more simpler and avoids mismatches between the names of the elements of project plans and the architecture.

The same will happen for the other terms of the language. A list for each term will be created as 'Customizable Field', which is a feature inherent of Microsoft Project that supports the customization of fields and the possibility of having a larger diversity of them during the project life cycle. It also allows to restrict the input that is given to the cells (with the use of a lookup table), such as preventing project managers from entering a value that is not contained in the list of displayed values. With this feature we are able to define our own fields, i.e., the fields we want to assign to the project tasks. This will be required since fields that already exist in this tool will not be enough to achieve the integration with Atlas (hence the need to create a new language), and many of them are not even necessary to achieve our
goal, like the 'Duration' and 'Predecessors' fields, but they are still important for the execution of the project itself.

### 3.4.3 MS-Project Macros

MS-Project macros will allow to export (and import) information to (and from) external sources. The changes that occur in the project plans are the source of changes in the architecture, so these macros will enable to export the valuable information from project plans to the architecture, and vice-versa. This way, the enterprise architecture solution can easily keep the architectural models constantly updated.

As defined in chapter 3, the information that needs to be transferred between project plans and the enterprise architecture consists in action (type of impact), class of the artefact (type of artefact), artefact 1 name (Artefact1), artefact 2 name (Artefact2), planned start date and planned finish date. Of the previously mentioned fields, only the planned start date and the planned finish date are fields provided by MS-Project as default in every project, reason why all the other fields had to be created as 'Customizable Fields'.

The 'Action' field corresponds to the possible types of impact that an artefact can suffer from a project, and which will match with some properties of the different artefacts in Atlas. It is essential that the types of impact exist both in MS-Project and Atlas, so they can be interconnected, and later their information merged. This field ('Type_of_Impact') consists of a predefined list of well-known terms defined earlier by us, which is composed by CREATE, DELETE, USE, DEPLOY, UNDEPLOY and UPGRADE.
Moving on to the next field, ‘Type of Artefact’, it corresponds to the name of the classes present in the metamodel of the organization, such as, for example, Application Component, System Software or Node. These classes are known from the beginning, thus the procedure implemented will be the same; create this field and introduce its possible values to the lookup table. The values correspond to the artefact class names extracted from Atlas after applying the steps from Fig. 3.9 and Fig. 3.10). A zipped folder is downloaded, which contains the template files of all metamodel classes, as seen in Fig. 3.12.

![Figure 3.12: Artefact class names exported from Atlas.](image)

Regarding the ‘Artefact1’ and ‘Artefact2’ fields, each one represents the name (not class) of the artefact being referenced by the ‘Action’ that anticipates it. The list of names of each artefact class can be obtained from the first column of each file, as shown in Fig. 3.8, and shall be used whenever a certain project is creating (CREATE), deleting (DELETE), using (USE), deploying (DEPLOY) or undeploying (UNDEPLOY) an artefact. The ‘Artefact2’ field must only be used when an artefact undergoes an upgrade (UPGRADE). We tried to implement this restriction in the lookup table so that when choosing the USE impact, for instance, only ‘Artefact1’ could be chosen and not ‘Artefact2’, but the limitations of Microsoft Project did not allow it. However, in the propagation process developed, even if a value is chosen for ‘Artefact2’, it is not taken into account since it is assumed it was inserted by mistake. Finally for ‘Node1’ and ‘Node2’ fields, their values correspond to the ones from the metamodel class ‘Nodes’, that was already exported from Atlas (Fig. 3.12).

The customizable fields described above will only need to be added once to each project manager’s Microsoft Project, assuming they use always the same computer, which is the most usual situation at a professional level. Those values can be stored locally in an organizer named ‘Global.MPT’, so there is no need to create those fields and lists of values (lookup tables) whenever there is a new project, which would be quite overwhelming.

After this inclusion of customizable fields and lookup tables in MS-Project, the project manager can
start planning its projects, filling the necessary information in each field, depending on the values in the available lists. Having the project plan complete, the next step is to export the fields that matter (the ones that compose the defined language) from the project plan, so they can be later imported from Atlas. The file format chosen to do this propagation was the Excel format, since Atlas supports both importation and exportation of information in this file format.

For this purpose, we made a Microsoft macro called ‘Export’ that uses a feature named ‘Map’, that has the function to export only the specified fields, and not the whole project plan, to an external file, which in this case is an Excel file. This feature (map) is responsible for assigning fields from MS-Project to the respective fields of the Excel file. The template of the file exported after running the ‘Export’ macro will be similar to the one in Fig. 3.13, which will only have the filled fields when they represent project milestones. The other fields of the project plan exist in Microsoft Project, such as the costs and resources of each task, but they are not useful from an architecture perspective, so they weren’t exported. This file hereafter is going to be called as taxonomy file.

![Figure 3.13: Excel file exported from MS-Project after running the ‘Export’ macro.](image)

Following the propagation and unification of information, the respective updated files will be imported from Atlas, so the architectural representations can be designed, as explained in chapter 3. Both the project manager and the enterprise cartographer will be able to identify, analyse and evaluate the dependencies between projects and EA artefacts, with the help of the alerts feature implemented. Whenever some invalid dependencies occur, the enterprise cartographer will be able to correct those situations so they can become correct in a business context. Therefore, those modifications will have to be propagated back to the respective project plan, in order to have coherent and consistent information within the different departments and sectors of an organization.

To support this coherence, another Microsoft macro was developed, this one called ‘Import’. It allows
updating the information of project plans, based on the information that has been modified and improved in the architecture models. Whence, it would not make sense to update the information in only one side of the pipe since they are connected and dependent of each other, and so the effort that is required by the project manager and the enterprise cartographer is much smaller this way. This macro also uses the feature ‘map’ to import external information and to match fields from Excel to the corresponding ones in MS-project.

A restriction that is imposed by this macro is that whenever an artefact is created in the architecture of the organization, it has to be added to the lookup table of that field (‘Artefact1’, ‘Artefact2’, ‘Node1’ or ‘Node2’) before using the ‘Import’ macro. By creating a new artefact in a particular project plan, it may be new to the architecture as well, although it is not very ordinary. Most commonly, the artefacts have already been previously referred to in the organization, whether in another state of their life cycle, such as in their projection for example. Microsoft Project does not allow to import values that do not exist on the lookup table of a certain customizable field, so this information has to be communicated between project manager and enterprise cartographer before importation takes place. If this communication does not happen and the name of an artefact that does not exist in the architecture tries to be imported, the macro will import all values except the recently created, so the insertion of the new value into the corresponding lookup table has to occur for the system work one hundred percent correctly.

3.4.4 Microsoft Excel Macros

After having the information exported from project plans to an Excel file, the next step is to manipulate that same data through the use of Excel macros, so the information in the template exported from MS-Project can be uniformed to be compliant with the templates that can be imported from Atlas. To be able to reach that unification, one needs to know what are the templates supported by Atlas, that is, the templates that contain all the information that Atlas needs to import so that the architectural representations can be reproduced.

As we already described, they consist in distinct files that correspond to each of the classes present in the metamodel adopted by the organization. An example of this file is in Fig. 3.8, where the class and the properties shown match with the application component type of artefact.

Hereupon, we will have to transfer information from the taxonomy file (Fig. 3.13) to the artefact templates (Fig. 3.8), in order to integrate the two technological components used in this dissertation. The taxonomy file will be exported through the ‘Export’ macro (see section 3.4.3) and the artefact templates through the method already explained (see Fig. 3.9 and Fig. 3.10). Once we have all the useful information, the process of propagation can begin. This process will occur in a separate Excel file originated for this purpose, from now on called as ‘workbook’, which will be where all the necessary macros will be located, and where the data will be transmitted and uniformed to comply with the right templates. To
accomplish the data spread six macros were created:

- **GetTaxonomy** - Copies the taxonomy file into the first sheet of the workbook (sheet1).

- **GetSheets** - Copies all the files located in the same folder as the workbook into the following sheets of the workbook (sheet != 1), by alphabetical order. Those files must correspond to the artefact templates extracted from Atlas in the beginning of the process (Fig. 3.8).

- **UpdateData** - Propagates all the data (dates and impacts) from sheet 1, 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 sheet 1, 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, in the source files from where they were exported.

- **SaveTaxonomy** - Saves all the modifications made to sheet 1, that is, to the taxonomy file, in the source file from where it was exported.

In order to complement the previous macros, and to facilitate the process of updating the information, the **RunMacros** macro was also created. It is composed by two sub-macros that are responsible for the integration between MS-Project and Atlas, in both sides, which means that one is concerned with keeping the information from the architectural models updated based on the information descendant from project plans (ProjectAtlas1), and the other is concerned with the other way around, that is, in keeping the project plans information updated based on the information descendant from the architectural representations (AtlasProject2). Those two sub-macros take advantage of the other six macros to be able to correctly update the information, as we can see below:

- **ProjectAtlas1** - GetSheets + GetTaxonomy + UpdateData + SaveSheets, in this specific order.

- **AtlasProject2** - GetSheets + GetTaxonomy + ReverseData + SaveTaxonomy, in this specific order.

The first sub-macro presented allows to update the information in the direction MS-Project → Atlas, that is, updating the architectural models based on the information coming from the project plans. It runs the macros GetSheets and GetTaxonomy to get both artefact templates and the important part of the project plan (taxonomy file) into the workbook, after they have been respectively exported from Atlas and MS-Project. Then, this sub-macro propagates the necessary information from the taxonomy file into
the respective artefact templates (sheets different than the first one of the workbook), with the help of
UpdateData macro, and finally saves the latter, since the information was propagated in this direction
and the taxonomy file has not changed. Having the artefact templates properly updated, the next step
is to transfer this information to Atlas, so the blueprint queries can be executed and, consequently,
the blueprint drawn. For this matter, a similar approach to the one for the exportation of the artefact
templates must be done. Each artefact class has to be uploaded, so they must be sent to a compressed
folder in order to be able to upload all the classes at once.

First, one must go to the ‘Data Explorer’ menu and click on the icon symbolizing the upload. It is in
the upper right corner highlighted in orange since it is being hovered, as we can see in Fig. 3.14. Then,
the import mode must be chosen from a predefined list, and it must be equal to ‘destructive update’, as
shown in Fig. 3.15. This mode refers that the full content (properties and references) of data instances
existing both in the repository of Atlas and in the file is set as in the file. Data instances existing only
in the file are created with the contents as in the file, and data instances existing only in the repository
remain unchanged. Finally, the ‘Select files’ button must be chosen (Fig. 3.16) so the project manager
can pick the zipped folder with all the artefact templates updated, in order to upload them to Atlas. Based
on this data, the enterprise architecture can be kept up-to-date with minimal effort.

![Figure 3.14: How to import artefact templates: step 1.](image)

Additionally, and in contrast, we have the second sub-macro that allows to update the information in
the opposite direction, from Atlas to MS-Project. This means that project plans will be updated based
on the information descendant from the architectural representations, because these may be modified
and improved by the cartographer after consulting and analysing the blueprint. In a similar way to the
example above, GetSheets and GetTaxonomy will be run in order to get the taxonomy file and the
artefact templates into the workbook. After that, the ReverseData will be responsible for propagating
the information changed in the architectural models and that have impact in the project management
domain, that is, in project plans, by updating the respective fields in the taxonomy file. Finally, this
taxonomy file is saved into its original location, since this will be the only file being updated when running
this sub-macro. After having this last file updated and saved, it is only needed to use the ‘Import’ macro
in Microsoft Project, in order to transmit the updated information to the project management tool. This
way, the project manager will not have much effort in keeping project plans up-to-date on the basis of
the models from the architecture.

3.4.5 Macros Support

Considering that the integration between MS-Project and Atlas will be used by people possibly with dif-
dferent display languages, it is important that different idioms are supported. Fields, classes and property
names may vary depending on the organization and the people that are using the tools.
This obstacle was overcome by creating an indexation table, which due to the fact that the transmis-

sion of information between the two technological components is done through Excel files, this table will

be also done in this file format, as it can be seen in Fig. 3.17.

Every field from MS-Project and Atlas that belong to the defined language and that are vital to reach

the solution are represented in this table, since they might change depending on the display language

chosen by the person who will use it, or simply because one came to the conclusion that those names

no longer made sense, and had to be replaced by others. In order to support and maintain the efficiency

of this integration despite the changes made over time, we had to make the source code of the macros
dynamic. This way, one does not need to edit the macros source code whenever a change is made to

the names assigned to the different fields.

The indexation table is divided in three columns. The first one refers the category in which the fields

and properties are inserted, that is, if they typify words from Atlas (property names), Taxonomy (field

names from MS-Project), Classes (names of artefact classes) or even Macros (file paths). The latter

makes it easy and more flexible to change both folders and computers where the files are located. It also

allows the files to be transferred between different people, since this action would influence the paths of

those same files.

The following column (second) is composed by the names that will be read by Excel macros, which

means that they are immutable. Wherefore, those cells are written in English since it is one of the largest

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Macros</td>
<td>C:\Users\francisca\cambra\Dropbox\Faculdade\Thesis\MS-Project\Macros\Indexation\</td>
</tr>
<tr>
<td>2</td>
<td>Macros</td>
<td>C:\Users\francisca\cambra\Dropbox\Faculdade\Thesis\MS-Project\Macros\Indexation\Taxonomy\</td>
</tr>
<tr>
<td>3</td>
<td>Macros</td>
<td>Blank idm</td>
</tr>
<tr>
<td>4</td>
<td>Classes</td>
<td>Application Component</td>
</tr>
<tr>
<td>5</td>
<td>Classes</td>
<td>Application Service</td>
</tr>
<tr>
<td>6</td>
<td>Classes</td>
<td>Business Process</td>
</tr>
<tr>
<td>7</td>
<td>Classes</td>
<td>Business Actor</td>
</tr>
<tr>
<td>8</td>
<td>Classes</td>
<td>Node</td>
</tr>
<tr>
<td>9</td>
<td>Classes</td>
<td>System Software</td>
</tr>
<tr>
<td>10</td>
<td>Taxonomy</td>
<td>Name</td>
</tr>
<tr>
<td>11</td>
<td>Taxonomy</td>
<td>Type_of_Impact</td>
</tr>
<tr>
<td>12</td>
<td>Taxonomy</td>
<td>Type_of_Artefact</td>
</tr>
<tr>
<td>13</td>
<td>Taxonomy</td>
<td>Artefact1</td>
</tr>
<tr>
<td>14</td>
<td>Taxonomy</td>
<td>Node1</td>
</tr>
<tr>
<td>15</td>
<td>Taxonomy</td>
<td>Planned_Start_Date</td>
</tr>
<tr>
<td>16</td>
<td>Taxonomy</td>
<td>Planned_Finish_Date</td>
</tr>
<tr>
<td>17</td>
<td>Taxonomy</td>
<td>Production_Date</td>
</tr>
<tr>
<td>18</td>
<td>Taxonomy</td>
<td>Deprecated_Date</td>
</tr>
<tr>
<td>19</td>
<td>Atlas</td>
<td>BeginDate</td>
</tr>
<tr>
<td>20</td>
<td>Atlas</td>
<td>EndDate</td>
</tr>
<tr>
<td>21</td>
<td>Atlas</td>
<td>Create</td>
</tr>
<tr>
<td>22</td>
<td>Atlas</td>
<td>Delete</td>
</tr>
</tbody>
</table>

Figure 3.17: Indexation file template.
universal languages, so it is easier for everyone to understand them. They represent the information necessary to access for integration to be possible.

The third column has the encoded names which represent the actual/real key words from Atlas, MS-Project and file paths. These encoded names must be filled by the person that is using the tool, so the fields from MS-Project, the classes and properties of Atlas’ repository and the file paths can be concordant with its information.

This indexation file is copied into the workbook after running the GetSheets macro, so it must be in the same folder as the workbook and the artefact templates. Independently of who is using the tools, the indexation file enables the flexibility and adaptability of the integration system to every country, organization and person. This makes it possible to have different names for each piece of information needed and used to keep architectural models and project plans updated based on the information descendant from each other.

3.4.6 Data propagation details

From the previous section, we already know which are going to be the existing macros and what are their functions. There are two sub-macros, one responsible for the propagation, and all that it implies, of data from the project plans to the architectural representations, and the other one on the other way around, which means it gets the changes made in the architecture and transfers them to the organization’s project plan present in MS-Project. This occurs because transformation initiatives, like projects, are responsible for achieving the goals set by its organization, which correspond to the deliverables accomplished within projects, that originate, transform, use and delete elements [21]. But how is the information from the defined taxonomy propagated through macros? How was it implemented?

There are two distinct and essential steps in this implementation process. The first one is concerned with the transmission of information from the taxonomy file, which is the file that contains the information that will have impact in the architecture and that was exported from MS-Project, to the artefact templates, which are the templates supported by Atlas that contain all the information about the artefacts of the different classes of the metamodel adopted by the organization in question, and that will allow the architectural models to be designed through this framework (Atlas), and through the execution of queries. The second step is concerned with the transfer of information from these artefact templates directly to the taxonomy file, which will be later imported from Microsoft Project, allowing the project plan to be updated and aligned with the organization’s strategy, since enterprise architecture aims guiding organizations through the business, information, process and technology changes, vital to execute their strategies and accomplish business success [10].

Regarding the first step, and in order to contextualize the situation, the whole process that must occur from the point of view of the project manager since the beginning of the project life cycle to the update
of the architecture will be as follows:

1. Several files, each one containing the list of artefacts of a given metamodel class (artefact templates), will have to be exported from Atlas.

2. Add those artefacts to the lookup table of each created customizable field in Microsoft Project.

3. Do the planning of the project, filling in each milestone the information regarding the created fields, such as the type of impact, type of artefact, artefact name, node name, if applicable, and planned start and finish dates.

4. After having the whole project planned, run the 'Export' macro in MS-Project. It will create an Excel file (named taxonomy) with all the necessary information to transmit to the architecture, in order to keep it updated in an automatic way.

5. Open the workbook, which is the Excel file where all the macros are located, and run ProjectAtlas1. This sub-macro will propagate the information from the taxonomy file into the respective artefact templates.

6. Import the templates of all artefact classes into Atlas, by uploading the folder where all the modified artefacts templates are, and which will be already aligned with the transformation initiatives that take place within the organization, that is, with projects.

7. Run the batch that verifies if the transformation dates of all EA artefacts are valid in the 'Batch Explorer' menu of Atlas.

8. Select in the 'Blueprint Explorer' menu of Atlas the name of the project that needs to be analyzed (the one planned and referenced in the previous bullets).

9. Identify and evaluate the dependencies between the projects of the organization’s project portfolio and the EA artefacts, through the observation of the blueprint and error container.

10. Modify the incoherent relations between projects and artefacts directly in the blueprint, improving the organization’s alignment, since transformations made within different departments are difficult to control and maintain valid.

Almost every bullet either is self explained or has been described in detail in other sections, except for the sixth. This is one of the most important steps so it has to be carefully explained.

Although the files from MS-Project (Fig. 3.13) and Atlas (Fig. 3.8) have different templates and characteristics, we can still manage to cross-reference information that is common to both files, the taxonomy file and the artefact template. The UpdateData macro is responsible for the propagation and unification of that data, which will have to be imported from Atlas so the architectural models are kept updated.
according to changes that occur within the organization, through the deliverables of the projects. Therefore, it makes sense that the transmission of information occurs from the project plan to the templates of the artefacts that were extracted from Atlas, so these can be later imported back in the right format, and with the information properly updated.

Foremost, we must analyse what is the information that crosses, and from Fig. 3.13 and Fig. 3.8 we can conclude this related information is composed by artefact names, artefact classes, project name, types of impact and dates, which corresponds mostly to the taxonomy defined in chapter 3. The artefact names can be obtained through the fields 'Artefact1', 'Artefact2', 'Node1' and 'Node2' of the taxonomy file (Fig. 3.13), and through the first column of each artefact template (Fig. 3.8), that contains all the names of the artefacts from that class. The artefact classes can be obtained through the name of the file itself, since when those classes are exported from Atlas, their name is already saved with the respective name of the metamodel class. The project name can be gotten from the second row of the taxonomy file, and from the cross between the name of the artefact and the type of impact of the artefacts template. For example, if it is known that a project named 'ProjectPlan' creates an application component named 'CRM', as we can see in Fig. 3.13, then the name of that project must be in the property 'Created by' from the artefact template of the application component, obviously in the row that corresponds to that artefact ('CRM'), as we can verify in Fig. 3.8. It should be noted that the file from Fig. 3.8 is already the output obtained after the transfer and unification of data from the file of Fig. 3.13. Secondarily, we have the different types of impact, which are described in the field 'Type of Impact' of the taxonomy file, and in the respective properties of the artefact templates, such as the 'Created by', 'Deleted by', 'Used by', 'Deployed by' and 'Undeployed by'. Finally, concerning dates, they can be seen in the 'Planned Start Date' and 'Planned Finish Date' fields from the taxonomy file, and in the respective properties of the artefact templates, like the 'Production Date' and 'Deprecation Date', as can be seen in the example below (Fig. 3.18).

Knowing how the fields from the taxonomy file fields and the artefact templates relate to each other, the next step was the implementation of the **UpdateData** macro, that briefly is transmitted in:

1. Consult the indexation table, and save the names of all the information that will be needed to propagate into variables.

2. Go through every cell of all files in the same folder as the workbook (except for the workbook itself, and the taxonomy and indexation files), and whenever the name of the project (the one referenced in the second row of the project plan extracted from MS-Project, Fig. 3.13) is found in any impact
cells, remove it. This will allow the process to be performed indefinitely, since the stored information will always be the last one to be run, and the previous information will be overwritten.

3. Go through every cell of the 'Type of Artefact' field of sheet 1 (taxonomy file), and whenever a word is found, go to the workbook spreadsheet with that same name to propagate the information, which will correspond to the artefact template.

(a) Evaluate the 'Type of Impact' field. If it is equal to 'UPGRADE', then save both 'Artefact1' and 'Artefact2' into variables. Otherwise, save just 'Artefact1'.

(b) Find the row in the respective sheet of the workbook where the artefact name is equal to the value saved in the variable that contains the information regarding the name of the 'Artefact1' (and of 'Artefact2' if it the case).

(c) Check what is the type of impact in the taxonomy file of the row being evaluated.

(d) Go to the respective property column in the artefact template, and insert the name of the project in that cell. If another name is already there, append it to the previous string. Otherwise, paste the name of the project directly into that cell.

(e) Copy the values of the dates from the taxonomy file to the corresponding fields in the row of the artefact being analysed ('Propagation Date' for instance, whenever the 'Type of Impact' is equal to 'CREATE').

4. Propagate the project dates from the taxonomy file into the respective fields from the corresponding artefact template, which will always be the 'Work Package' sheet.

5. Move to the next row of the taxonomy file, and repeat every point described since the third one.

6. When an empty cell is found, the macro ends.

After the information descendant from project plans has been transmitted to the artefact templates, it will be imported from Atlas, and the architectural representations will be consequently displayed. The enterprise cartographer and the project manager will be able to analyse the dependencies caused by different projects when they impact the same EA artefacts. Those dependencies are easily observable through the error container, which was implemented in the blueprint of Fig. 3.4. The existing errors can be eliminated by the cartographer in the enterprise architecture tool itself, but what about project plans? Will they keep containing wrong information?

Whenever the execution of tasks that affect and depend on other projects in the portfolio is being planned, due to the artefacts they influence in a common way, this information cannot be registered and enhanced only in the enterprise architecture tool. If incoherent situations are observed in the planning of
a project, because when a project plan is carried out one has no idea of the existing dependencies, they must be suppressed and resolved as soon as possible, in order to avoid that a greater number of constraints occur in the execution of the project itself. Thus, the improvements made in the blueprint created in Atlas must also be propagated to the corresponding project plan. It was this need that triggered the creation of the ReverseData macro, that must be run only after exporting the artefact templates, since they might have suffered modifications from the last time they were exported, when the cartographer solved and fixed incongruous relations between projects and EA artefacts. This macro works much like the one described above because the common elements between the taxonomy file and the artefact templates are the same. From an user point of view, the steps that the project manager has to follow consist broadly in the ones presented below in Fig. 3.19.

The activities typified above will contribute to the success of the implemented solution, whose main affected will be the actors represented in the following use case (Fig. 3.20), as well as its actions or event steps that define the interactions between them and the system, which is our solution.
Figure 3.20: Use Case of the solution.
4 Evaluation
4.1 Metrics

In order to evaluate the performance of this integration, the metrics that will be used are related with **time** and **effort**, and they are going to be evaluated when performing activities required by the roles concerned. Does this more automatic solution help project managers doing their job, preventing them from having to create and update project forms in Atlas, which is the (old) solution already implemented by Link Consulting to try to solve this problem of not having methods for the inference of IT architecture based on project plans? Have there been improvements on times and effort required by stakeholders with this new solution?

To achieve results, the approach that will be followed is to create hypothetical scenarios that are likely and common in IT organizations and in an IT domain overall, to examine whether the chosen language (taxonomy) works when applied. Some situations that might exist and are very ordinary when planning and executing a project are the following: create artefacts (1), delete artefacts (2), read artefacts (3), change artefacts (4), deploy artefacts (5), undeploy artefacts (6), upgrade artefacts (7), replicate artefacts (8), aggregate artefacts, that is, take three artefacts and replace them by one for example (9), and disaggregate artefacts, that is, take one artefact and split it into two for example (10).

Situations 1, 2, 3, 5, 6, and 7 can be achieved by applying directly the language created in this dissertation. The following milestones would have to be created, respectively, in order to accomplish the integration: CREATE artefact\(_x\), DELETE artefact\(_y\), USE artefact\(_c\), DEPLOY artefact\(_d\), UNDEPLOY artefact\(_e\) and UPGRADE from artefact\(_f1\) to artefact\(_f2\). The fourth (4) can be reached by deleting an artefact and by creating a new one with the required properties. That implies adding two milestones, one with DELETE artefact\(_g\) and another with CREATE artefact\(_h\). The eighth (8) can be reached by creating a new artefact with the same properties as the one we are replicating, but with a new id (name), because it must be unique. This implies that the following milestones are created: DELETE artefact\(_{to\_replicate}\) and CREATE artefact\(_{replicated}\). Regarding the ninth (9), it would be needed to delete the three artefacts and create a single one, and about the tenth (10) it would be the opposite, which means we would have to delete an artefact and create two others. In every example explained above, a milestone for each impacted artefact had to be inserted in the project plan in MS-Project. Increasing the language concepts could imply that fewer milestones would have to be created to achieve the same results, but it would be much more complicated and less intuitive to tabulate each artefact impacted by some project in MS-Project, and even more complicated when doing the integration with Atlas.

After having the situations validated and the metrics to evaluate settled, we have to apply the latest in two different scenarios, namely with the previous solution Atlas had to keep architecture models up-to-date based on the information descendant from the project management domain, and with the solution presented and developed in this thesis.

The enterprise cartographer has the same effort and functions with both solutions. He uses Atlas as
an enterprise architecture solution to support an efficient management of enterprise architectures, but since we are evaluating the differences between the effort and time required to maintain architectural representations updated based on project plans, we have to analyze the necessary effort from the point of view of project managers. They are the ones responsible for inserting project data into Atlas, according to the solution previously supported by Atlas, and they are the ones that perform all activities that need to enter values, according to the new solution. The only task that is done by the enterprise cartographer is the modification and update of enterprise information after its analysis, whenever invalid dependencies exist between several projects that impact the same artefacts incorrectly.

As we can note with the old solution provided by Link Consulting, project managers have to use two tools efficiently and not entirely effortless, being those MS-Project and Atlas. With this proposed solution, they will need to work effectively with just one, that is, the project management tool, which is the regular situation already. Thus, is the use of the project management tool sufficient and better than the solution that requires project managers to use both of the tools? In order to reach some conclusions, it is necessary to analyze the different scenarios described in the following subsections.

### 4.1.1 Scenario 1

This scenario focuses on the responsibilities that project managers had with the previous solution supported by Atlas, which consisted in manually introducing the impacted artefacts in templates made for that purpose. One form had to be filled when creating the project plan, that is, for its insertion for the first time in the system. This had to be done by the project manager because he is the one with expertise in project portfolio management, and more specifically in project management. Additionally, another form had to be filled, whenever the project plan of an existing project was updated.

To create project plans with the available forms, the next steps must be accomplished:

- Create and fill the project plan in MS-Project;
- Open 'Project' form in Atlas;
- Write project properties, such as its name, scope, planned start date, etc. (Fig. 4.1);
- Fill every impacted artefact in the form (Fig. 4.2 and Fig. 4.3);
- Open 'Project Dependencies' blueprint in Atlas.
To update an existing project plan with the available templates, the next steps must be accomplished:

- Update the project plan in MS-Project;
- Open 'Project - WP Impact' form in Atlas;
- Choose the project to update;
- Fill every impacted artefact in the form;
- Open 'Project Dependencies' blueprint in Atlas.
Whenever the enterprise cartographer changes the architectural models in Atlas, that information has to be transmitted back to MS-Project. Since there is no integration between MS-Project and Atlas, all the artefacts that have been identified and modified, because they had invalid relationships with certain projects, must be manually inserted into their respective project plan in MS-Project. Aside from this time consuming process, it is quite laborious, tiring and not user friendly to insert the same information twice, in both the tools used.

### 4.1.2 Scenario 2

Contrary to the scenario described above, this one focuses on the responsibilities project managers have with the solution developed in this dissertation, which consist in propagating project plans information in a more automatic way, so the architectural models can be kept up-to-date. This is only possible because the integration between a project management tool and an enterprise architecture tool was made, which implies that there are only a small number of interactions and procedures to be performed by the project manager. This scenario is also composed by two situations that might happen; the creation and configuration of the project plan and all that it involves, and the consequent updating over its execution and over time.

When project managers need to create a project plan for the first time, they have to make some configurations so it can be accordant with the specifications supported by MS-Project, by Excel macros and by Atlas. These settings only have to be made **once**, which is when the machine (computer) of the person using the integration system varies. In order to achieve it, the steps below should be followed:

- Copy **Import** and **Export** macros into MS-Project and fill the respective file paths;
• Insert 'customizable fields' into MS-Project;

• Export artefact templates from Atlas repository and unzip the folder;

• Copy and paste values from artefact templates into the respective lookup tables in MS-Project;

• Fill file paths in indexation file, and in GetSheets and ThisWorkbook source code.

The steps identified in the configuration phase presented above, will not be counted to measure the system performance against outlined metrics (time and effort), since any system needs configurations to be performed. It will not be that single moment that affects and decreases the capabilities and efficiency of the solution developed. After the setup is done, project managers can then create the plan of their projects. For such, the steps below must be done:

• Create and fill project plan in MS-Project;

• Run Export macro in MS-Project;

• Open 'workbook' to propagate automatically dates and impacts;

• Zip artefact templates and import them from Atlas repository;

• Run 'Validate Dates' batch in Atlas;

• Open 'Project Dependencies' blueprint in Atlas.

When project managers need to update an existing project plan, after all the configuration and creation phases are done, the required steps they need to follow are:

• Update project plan in MS-Project;

• Run Export macro in MS-Project;

• Open 'workbook' to propagate automatically dates and impacts;

• Zip artefact templates and import them from Atlas repository.

Whenever the enterprise cartographer changes an architectural model in Atlas, it has to be transmitted back to MS-Project by following the next steps, both when creating and updating the project plan:

• Export artefact templates from Atlas repository and unzip the folder;

• Open 'workbook' to propagate automatically dates and impacts;

• Run Import macro in MS-Project.
4.1.3 Analysis and conclusions

The solution previously implemented by Link has two aspects that we consider to have a very negative effect on the smooth functioning of the non-automatic propagation of project changes. The first one is that it is mandatory to plan the project and fill its deliverables (artefacts created, updated, deleted, etc.) twice, both in MS-Project and in the forms offered by Atlas. When the universe of artefacts that is impacted by a given project is large, which typically occurs in medium and long term projects, the introduction of each artefact into its respective field becomes an exhausting and tedious task, so elaborating it manually is not the best approach to do so. The second one concerns the fact that whenever architectural models are improved and validated in Atlas, there is no way to communicate that same information to the project management domain. This implies that each modified impact must be corrected in the respective project, which can be either related to the project being analyzed or with the projects that the latter has relationships with. Therefore a constant communication between the different stakeholders of the organization must occur, and even between different departments, so that the information is coherent and valid in all the project portfolio, which is a highly fallible method.

According to the information obtained from the previous scenarios, a table (4.4) was prepared with the important metrics to consider and their results, comparing the new solution with the previous one.

<table>
<thead>
<tr>
<th></th>
<th>Old Solution</th>
<th>New Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MS to Atlas</td>
<td>Atlas to MS</td>
</tr>
<tr>
<td>Number of Steps</td>
<td>Create</td>
<td>Update</td>
</tr>
<tr>
<td>Few artefacts</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Many artefacts</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Number of Fields</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Few artefacts</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Many artefacts</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Effort</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Few artefacts</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Many artefacts</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Time</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Few artefacts</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Many artefacts</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>User Friendly</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Few artefacts</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Many artefacts</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 4.4: Data analysis.

Looking at the first metric used, none or few conclusions can be drawn, since the number of steps to perform is very similar in both solutions. Despite that the new solution has a larger number of steps globally, they often boil down to mere clicks, such as opening a file or clicking on the run button of a
batch, which implies minimal effort from the project manager.

Another metric used was the introduction of fields during the process of transferring information between project plans and architectural models. When the IT project does not imply many creations, updates, uninstalls or 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 very few 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 boxes, 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, field entries in 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 phase, 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 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 pick the artefact class from the form, and introduce a different row for each impacted artefact from that class (Fig 4.2). 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.

Time and effort metrics will be analyzed at the same time since they are complementary, and 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. This situation does not verify with this new solution. The time taken to transmit project plan information to the architectural models, and vice versa, will be the same regardless of the number of artefacts, since the tasks to be performed (zipping a folder, opening a file, clicking in buttons, etc.) will always be constant and unchanging. Obviously the execution time of those tasks, however simple and short 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., when the IT project has impact on 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 the other cases, it does not support an efficient and fast method of propagation of the information from the project plan to the architectural representations, and requires the project manager to enter the same information twice into separate tools, which in addition to being exhaustive, is quite susceptible to misaligned and incoherent information.

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

The comments and reflections made along the previous paragraphs allow us to realize that although the initiation process seems exhaustive with regard to 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. Therefore, the migration of the system from a development to a production environment gathers all the conditions and evidences to be successfully implemented and to produce good outcomes.

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 [25].

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 successsively refine and deliver a software system [25].

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.
Conclusion
5.1 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 and execution of their projects since this solution 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).

### 5.2 System Limitations and Future Work

To allow the collaboration between two distinct tools, MS-Project and Atlas, an integration method had to be developed. It consisted in the elaboration of Excel macros in the VBA programming language that had as main function the transformation of the file templates used so they could be compliant with each other. Therefore, the system limitations were imposed by the limitations of the tools used, i.e., MS-Project, Excel and Atlas. These limitations are divided by tools, starting with the ones imposed by Atlas:

- Artefact templates cannot be imported and exported trough the execution of a batch, which would be much simpler, with just a click, than having to do it manually, by clicking on the corresponding icon (import or export);

- In the artefact templates, the cells that cross information regarding what projects have impact on a certain artefact are composed by the names of the projects compressed, as can be observed in Fig. 3.9. This makes the situation where there are two projects with similar and matching names, such as 'ProjectPlan' and 'ProjectPlan2', vulnerable. As the solution is implemented, the impacts won’t be propagated correctly. In the impact cells where 'ProjectPlan2' is referenced, assuming we are analysing 'ProjectPlan', the macro will check whether the word 'ProjectPlan' is contained in the cells to delete it before transmitting the impacts. The word 'ProjectPlan' will be deleted and the word '2' will be left in every cell that contains 'ProjectPlan2'. To overcome this impasse, delimiters (';') were created to divide project names in order to avoid misinterpretations. When importing the artefact templates from Atlas, errors were generated. This happened because Atlas does not support that type of delimiter since it has its own delimiter ('Alt + Enter').

The second attempt was to modify the code of the macros so that they could identify the line change as the delimiter between different project names. Again, after applying the macros that propagated the information from the project’s plan to the architectural models, errors arose due to the fact that the delimiter used by Atlas differs between properties. Among project names no delimiter was accepted when trying to import the templates from the artefact classes, so the names had to be aggregated, which decreases the efficiency of the solution.
The restrictions limited by MS-Project and Excel are the following:

- Macros have to be enabled before their execution, which may reduce the system security (tool: MS-Project and Excel);

- To add values to a lookup table we have to copy and paste them, which would be more efficient if one could import them directly from an external file, like an Excel (tool: MS-Project);

- Whenever a lookup table is assigned to a column and the 'Import' macro is run to update the value from a certain cell, if that value is not part of the lookup table, the value will not be imported. This is a limitation of using lookup tables, and it must be taken in account since it cannot be changed. The new values should be added to the corresponding lookup table before running the macro so they can be properly updated (tool: MS-Project).

Future research activities in this area follow two different paths, one more theoretical and one more practical. Regarding the first one, an exhaustive investigation in order to understand specifically which enterprise artefacts can be impacted by projects might be done, since some of them that are present and supported by the organization’s metamodel and don’t have a direct relation with transformation initiatives may exist. The EA artefacts which can be referenced in the project management domain may be explored, and vice-versa, since these depend on each other as we already saw.

On a more practical domain, the integration solution could allow in a more efficient way the situation in which more than one artefact is impacted by an IT project. Whenever this situation occurs, a milestone has to be created for each EA artefact that suffers any kind of impact from a project.

Another practical method that can be improved is the ReverseData macro, since it is a bit archaic. It goes through every cell of the 'Artefact' field in the taxonomy file, gets an artefact name, checks what is the 'Type_of_Artefact' of that artefact, and goes to its corresponding artefact template. Whenever the artefact being analysed in the template matches with the one evaluated in the taxonomy file, the column that refers to the project impacts is assessed. When an impact is identified, from left to right, the 'Type_of_Impact' in the taxonomy file is evaluated to verify if they agree. If not, the impact is changed directly in the taxonomy file so the project plan can later be imported from MS-Project with the modified changes. This means that whenever two milestones that have two distinct impacts for the same artefact exist, those fields in the taxonomy file will be updated with the impact found in the rightmost column in the artefact template, which may not always be correct. A proposed solution to solve this problem would be the implementation of specific IDs for each combination of artefact and impact. IDs just for the artefacts wouldn’t be enough since the artefact to be referenced could be the same as in the previous example.

The concepts described in this thesis as well as the proposed solution are the result of research and work developed for a master thesis at Instituto Superior Tecnico (IST)/Link Consulting.
Bibliography


Figure 1: Blueprint created for the analysis of project dependencies