Project: Enterprise Architecture in an Agile world

DAVID PEREIRA and PEDRO SOUSA*, Instituto Superior Técnico, Portugal

Enterprise Architecture and Agile methodologies are two increasingly important notions in the enterprise world. The first is crucial to ensure business success through documentation of the current practices of the enterprise, and enable innovation through the safety of well-documented processes and functions [14]. The latter helps reduce operation costs, and in turn, maximize profits, through different software development processes [1]. This study utilizes the already defined workflows of an Agile development project to develop automated processes that help in creating and maintaining architectural representations of the project. The Enterprise Architecture follows the principles stated in TOGAF's ADM [12] and it uses the Atlas Enterprise Cartography Tool [11] to support the creation of dynamic architectural assets, providing a mapping between Agile development and Enterprise Architecture.

Additional Key Words and Phrases: Enterprise Architecture, Agile methodologies, TOGAF ADM, Atlas Enterprise Cartography Tool

1 INTRODUCTION

The notion of Enterprise Architecture is ever-present within organizations. With the demanding requirements of the technological landscape, it is becoming harder to process information and control change. Enterprise Architecture is therefore gaining importance, as a method to control and document the organization's current practices and as an enabler of change and evolution [16].

On the other hand, software development methodologies are evolving to prioritize a fast product delivery, focused on the client's needs. This is achieved by bridging the gap between the business and the development aspects, and prioritizing in-person meetings and discussion. These are the principles of Agile development, as stated in the Agile Manifesto [3].

There are some incompatibilities between Enterprise Architecture and Agile development. The main incompatibility is the Agile mindset of focusing on delivering complete software as fast as possible and sacrificing the documentation to achieve that.

There is evidence that Enterprise Architecture can be deployed in the context of Agile development methodologies. Studies show that doing so may lead to several benefits to both the Agile development team, and the governing body of the organization [4, 15]. Additionally, models that map Agile development and Enterprise Architecture are also available [5, 7]. These models focus on attributing different roles to the team members, regarding the creation of the architecture, and propose a mapping with the TOGAF ADM framework.

To create dynamic documentation, the Atlas Cartography Tool [11] is used alongside TOGAF's ADM [12], guiding the development of the Enterprise Architecture assets.

The main objectives are the easy maintainability of the architecture, without requiring excessive manual effort and minimizing the impact that the solution has on the already existing workflow, supporting the ever-changing needs and requirements of the development team and client.

Authors' address: David Pereira, david.apolinario.pereira@tecnico.ulisboa.pt; Pedro Sousa, pedro.sousa@tecnico.ulisboa.pt, Instituto Superior Técnico, Av. Rovisco Pais 1, Lisbon, Portugal, 1049-001.

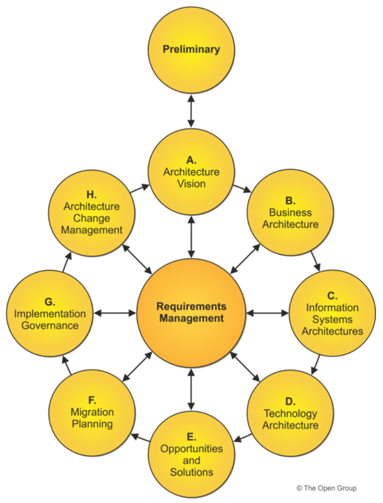

Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management
© The Open Group

Fig. 1. Phases of TOGAF's Architecture Development Method

The following Sections provide an overview of some important concepts, utilized on this study.

1.1 Enterprise Architecture

Enterprise Architecture is a widely studied topic. Zachman defines Enterprise Architecture as "that set of descriptive representations (i.e. 'models') that are relevant for describing an Enterprise such that it can be produced to management's requirements (quality) and maintained over the period of its useful life (change)" [16]. These representations are created by joining domains that may have been unrelated in the past. This poses a challenge, as different domains are certain to differ in definitions and practices used [9].

With the gradual increase in the number of enterprises and the increasingly competitive market, innovation is now more than ever a requirement for success. Having good Enterprise Architecture practices is crucial. Providing insight on the business processes of the company, as well as keeping records of customers and partners are two advantages of Enterprise Architecture [8].

1.2 TOGAF ADM

To guide the creation of the project's Enterprise Architecture, TOGAF's Architecture Development Method [12] was the chosen framework. This framework supports the creation of the architecture through different phases that can be adapted to the needs of each Enterprise, with a focus on managing requirements during the whole cycle. Additionally, several documents and assets are recommended to support each phase. An overview of the Architecture Development Method can be seen in Figure 1.

The most relevant phases that support the created architecture are five phases. Phase 0 (Preliminary Phase) contains the groundwork of the cycle. In this phase, details such as the context and the scope of the architecture are detailed. Phase A (Architecture Vision) defines the value that should be provided by the architecture, as well as the compilation of information regarding vision, strategy, and goals. Phase B (Business Architecture) supports the creation of rules and guidelines that detail how the architecture will achieve the required business goals. Phase C (Information Systems Architecture) creates a mapping between the Business Architecture artifacts and the Data and Application artifacts. Phase D (Technology Architecture) creates documentation regarding the decisions behind technological choices.

1.3 Atlas Enterprise Cartography Tool

The Atlas Enterprise Cartography Tool is an instrument that can be used to control transformations within an organization, through the use of Enterprise Architecture principles. To achieve this, Atlas provides a repository where data can be gathered both manually and automatically and several representations of the imported data, that can be configured to suit the needs of the enterprise.

The data is structured in classes and objects, and several representations can be created on top of it, most notably, tables that contain objects and their properties, matrices that relate objects of two classes on a specific relation, and blueprints. These are explored to create the solution.

1.4 Agile Development Methodologies

Agile Software Development methodologies are defined in the Agile Manifesto [3] through a list of principles. Agile values the continuous delivery of quality software to the customer, while embracing change in the requirements "even late in development". Another important aspect of Agile is having the business side and the developers working closely together on a daily basis, with the goal of reducing development time and improving agility and adaptability.

Agile Development is increasing in popularity among enterprises. Scrum [10] is currently the most widely used Agile development methodology, representing 66% of overall Agile use, according to Digital.ai's 15th State of Agile Report [6]. Additionally, 52% of respondents claim that their enterprise has adopted Agile as the go-to development methodology, and only 3% claim that Agile development is not used on any scenario.

2 SOLUTION IMPLEMENTATION

2.1 Solution Architecture

Figure 2 contains an overview of the architecture of the created solution. The architecture representation follows the ArchiMate Enterprise Architecture Modeling Language [13]. The created Atlas Agile Integration is integrated into the Atlas solution.

The Atlas Agile Integration application includes three components:

(1) **Azure DevOps Batch Job**: This component is responsible for collecting information from the DevOps environment utilized by the Agile team, and importing it into Atlas. This program can be run periodically to ensure that the latest changes are reflected in the Atlas repository, as well as automatically, ensuring that the project team does not need to manually import the data.

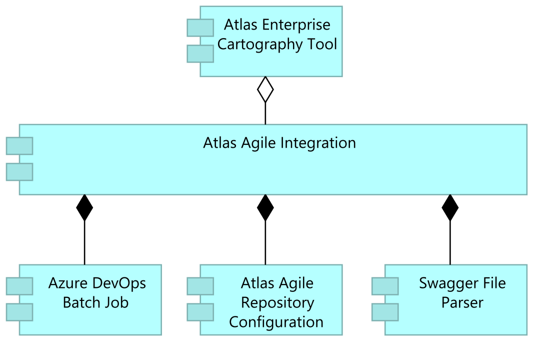

Atlas Enterprise Cartography Tool
Atlas Agile Integration
Azure DevOps Batch Job
Atlas Agile Repository Configuration
Swagger File Parser

Fig. 2. Solution Architecture

Table 1. Mapping between Azure DevOps Work Items and Atlas Architectural Assets

Work Item	Architectural Mapping
Epic	Application
Feature	Goal
Product Backlog Item	Requirement

(2) **Swagger File Parser**: This component includes an integration with Swagger. Its main objective is to parse internal documents to gather additional information that can be utilized by the Atlas configuration to better represent the architecture of the Project. Mainly, the information that can be gathered from these files includes dependencies between project modules, and external service utilization.

(3) **Atlas Agile Repository Configuration**: This component includes all of the classes, objects, representations, and documents that are generated by Atlas, following TOGAF ADM recommendations. Blueprints, matrices, and charts allow for the visualization of the architecture during a specific timestamp.

2.2 Supporting Project Details

The solution is based on a digital transformation project.

The project team chose to utilize Microsoft's Azure DevOps [2] environment to track the project's progress. The DevOps environment includes information regarding backlog management, sprint planning, and client acceptance. This information is used by Atlas to represent the architecture of the project.

Azure DevOps' work items are used to represent the details of the project. A mapping between these Agile concepts and the Atlas architectural ones is provided in Table 1. The development team establishes each Epic as a different Application required to complete the project. Under each Application, the team defines and aggregates the Objectives and Requirements needed to successfully implement each one.

3 RESULTS

This Section contains an overview of the created Enterprise Architecture. Figure 3 contains the layers and components utilized

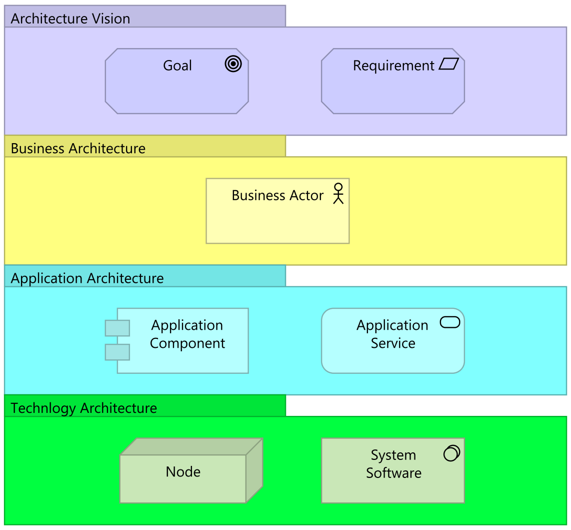


Fig. 3. Layers and components of the created Enterprise Architecture

Table 2. Description of the main architectural Representation Types

Representation Type	Description
Catalog	Tabular view showing every object of a certain class and the most relevant properties
Map	Matrix view showing a relation between the column class and the row class
Blueprint	Visual representation showing objects and relevant relations, according to the scope

to represent the architecture of the project. The representations are divided into several categories following the TOGAF standard, namely Architecture Vision, Business Architecture, Application Architecture, Data Architecture, and Technology Architecture.

After establishing the architecture layers, the representations were created. The types of representations used are detailed in Table 2. Catalogs are mainly used as a listing of the architectural objects. Maps are used to relate two classes based on a relevant property, and to also allow for quick updating of information. Finally, Blueprints are used to overview important objects, and relevant properties and relations, in specific contexts.

3.1 Architecture Vision Layer

The first layer of the proposed architecture is the Architecture Vision. Similarly to the TOGAF standard, this layer includes the motivation for the architecture, mainly focusing on the goals and requirements that are established for the project. To support this layer, representations such as the Goal and Requirement Catalogs, representing

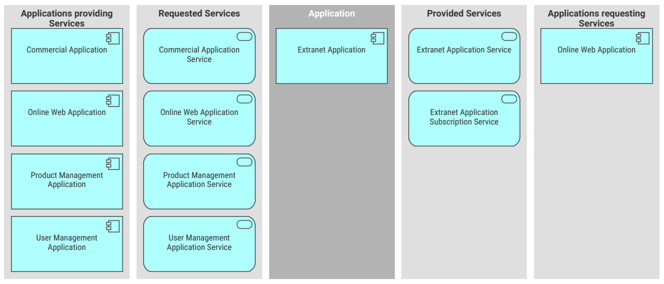
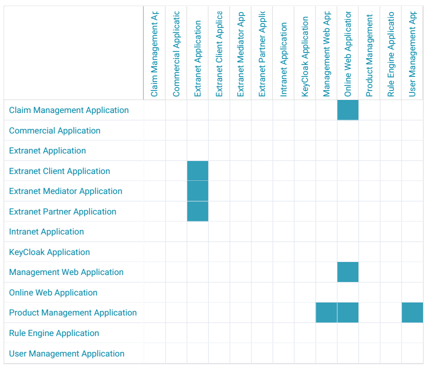

Applications providing Services
Commercial Application
Online Web Application
Product Management Application
User Management Application
Requested Services
Commercial Application Service
Online Web Application Service
Product Management Application Service
User Management Application Service
Application
Extranet Application
Provided Services
Extranet Application Service
Extranet Application Subscription Service
Applications requesting Services
Online Web Application

Fig. 4. The Application Integration Blueprint



	Claim Management Ap	Commercial Applicatic	Extranet Application	Extranet Client Applica	Extranet Mediator App	Extranet Partner Applic	Intranet Application	KeyCloak Application	Management Web App	Online Web Applicatior	Product Management	Rule Engine Applicatio	User Management App
Claim Management Application										■			
Commercial Application													
Extranet Application													
Extranet Client Application			■										
Extranet Mediator Application			■										
Extranet Partner Application			■										
Intranet Application													
KeyCloak Application													
Management Web Application										■			
Online Web Application													
Product Management Application									■	■			■
Rule Engine Application													
User Management Application													

Fig. 5. The Application Interaction Map Representation

tabular representation of every Goal and Requirement that is currently loaded into the Atlas repository (Table 3) are offered. Representations listing every Goal and Requirement under its Domain, offer a global view of the Architecture Vision.

3.2 Business Architecture Layer

To support the Business level of the architecture, the Requirements and Goals utilized in the Architecture Vision phase were combined with the information collected from the DevOps environment regarding the project team members that are assigned to each PBI, and consequently, assigned to each Goal, Requirement, and Application. To support this layer, representations such as the Business Actor Context Blueprint are offered, showing the assets that are impacted by a specific actor in some way.

3.3 Application Architecture Layer

Regarding the Application Architecture Layer, the main components of this layer are the Application Component and the Application Service classes. They represent the required software to be developed and the communication interfaces established by them, respectively. Similarly to the Architecture Vision Layer, Catalog views of the Applications are offered. Additionally, blueprints and matrices showing dependencies between Applications are supported (Figures 4 and 5).

Table 3. The Goal Catalog Representation

Name	Strategic Domain	Owner	Description	Productive Date	Decommission Date
Claim Data Integration	Claim	Alice Clark	The Claims received from the current platform must be completely integrated.	07-02-2022	31-12-2022
Client Area	Client	Mark Evans	A new Client Area that shows the past and current Claims must be created.	27-06-2022	31-12-2022
Customer Support	Client	Mark Evans	A Customer Support feature must be integrated into the new solution, with a Chat Bot.	18-04-2022	01-06-2023
Document Access Control	Document	Bob Patel	The Documents received from the different platforms must be accessible.	04-07-2022	28-02-2023
Secure Sign-In	Security	John Irwin	Sign-in must be performed through secure protocols.	21-02-2022	19-05-2023

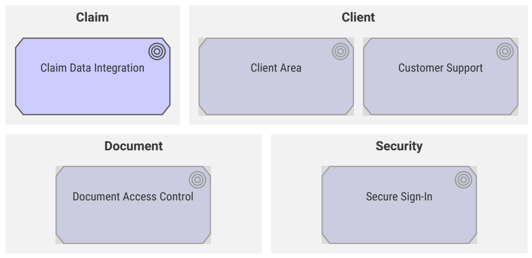


Fig. 6. The Goal Organic Blueprint on 07/02/2022

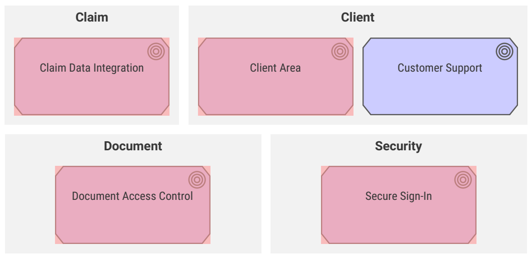


Fig. 7. The Goal Organic Blueprint on 01/06/2023

3.4 Technology Architecture Layer

The Technology Architecture Layer features the infrastructure that supports the Application, Business, and Vision landscapes. System Software represents software that supports the use of the Applications and that belong to a third-party vendor, supplied through licensing. The Nodes represent the physical machines that host the Applications and all of their functionality. It supports Catalogs of both these notions, as well as representations such as the Node and System Software Context Blueprints, depicting the Applications that are deployed at a certain Node and the usage of System Software per Application.

3.5 Evolution of the Architecture

The highlighted components support a layered Enterprise Architecture. However, the Architecture of an Agile development project is constantly changing and evolving, with new requirements from the client and unforeseen changes during development.

To combat this, a time bar is available in every Atlas representation, which allows for quick browsing through dates where state changes occur. This highlights the evolution of the Architecture in a visual manner, as colors are assigned to each state. For instance, Figures 6 and 7 show an example of how the time bar affects the representations in Atlas. As time passes, Goals shift from under development (grey fill), productive (no fill), and decommissioned (red fill).

This information can change according to the imported data. For instance, if a Goal's decommission date property were to be changed in some manner, the result would be immediately visible in the representations that showed that Goal.

4 DISCUSSION

4.1 Resulting Architecture

The Architecture detailed in Section 3 shows a collection of representations that can be used during Agile software development to represent the current and planned state of the assets. The solution makes use of the identified data suppliers and current workflows to generate documentation that evolves as new requirements are created and as new modules are developed. This creates a base that can be extended upon as the needs of the enterprise and the development team shift.

On the other hand, depending on the priorities of the project team, data can be manually inserted by any team member when there are no viable ways to automate its collection, without the need for deploying a separate architectural team.

Therefore, the result is a template that supports the creation of a layered architecture, focusing on the Architecture Vision (Section 3.1), Business Architecture Layer (Section 3.2), Application Architecture Layer (Section 3.3), and Technology Architecture Layer (Section 3.4), corresponding to phases A, B, C, and D of the TOGAF ADM, respectively.

4.2 Assessment of the Proposed Solution

Looking at the impact of the solution, three categories are explored.

Regarding the objective of gathering information whilst avoiding the disruption of the team's workflow, this is achieved with the presented automation tools. These tools have a very low impact on the workflow, and automatically import information into the Atlas repository.

Regarding the objective of automatically generating architectural documentation without introducing manual overheads, this is achieved by the creation of the dynamic representations in Atlas shown above. Every time new information is added through any means, the maps are updated, ensuring that no manual effort is required to reconfigure the views.

Regarding the effort required to represent the architecture, there are some mixed results. On one hand, information that is structured and included in the identified data sources, can be shown at no extra cost to the development team. On the other hand, any information that is not possible to automatically collect, i.e. external documents, requires manual effort to be introduced into Atlas.

To summarize, a mapping between Agile development and the TOGAF framework is proposed and an Enterprise Architecture that follows the TOGAF ADM recommended artifacts is presented. The Atlas repository acts as the Architecture Repository, containing the updated assets and representations. The Architecture Vision layer of the Architecture supports the Architecture Principles, through the Goal and Requirement tables and representations. The Business, Application, and Technology layers of the Architecture include the assets that are required by the Architecture Definition Document. All of the representations include the lifecycle of the objects, allowing each user to look into the future of the project and look at the planned changes, supporting an Architecture Roadmap of the project.

5 CONCLUSION

After assessing the solution and resulting architecture, two main conclusions are reached. The first is that Agile development and Enterprise Architecture are not mutually exclusive. This study shows an example of how an Enterprise Architecture framework (TOGAF ADM) can be deployed in the context of an Agile development project, without disrupting the workflow, and utilizing the available resources to facilitate its implementation.

The second discovery is that by analyzing the different information sources of an Agile project, it was possible to deploy automation processes that translated such information into architectural data. This proved useful when creating the underlying architecture, as it significantly lowered the manual effort required to maintain the architectural assets, allowing the project members to focus on delivering software instead of constantly updating the architecture. This is, however, highly dependant on the available data sources. The automated generation of representations requires at least some updated information to ensure the value of the architecture as a whole. Failing to identify and create these automated features, will lead to an increase in required manual effort.

REFERENCES

[1] Eman A Altameem. 2015. Impact of agile methodology on software development. *Computer and Information Science* 8, 2 (2015), 9.

[2] Elijah Batkoski. 2022. Azure DevOps Services REST API Reference. https://learn.microsoft.com/en-us/rest/api/azure/devops/ publisher: Microsoft, Accessed 18/10/2022.

[3] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. Manifesto for Agile Software Development. https://www.agilemanifesto.org/ https://agilemanifesto.org/. Accessed 10/09/2022.

[4] Mert Canat, Núria Pol Català, Alexander Jourkovski, Svetlomir Petrov, Martin Wellme, and Robert Lagerström. 2018. Enterprise architecture and agile development: Friends or foes?. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE, 176–183.

[5] Wissal Daoudi, Karim Doumi, and Laila Kjiri. 2020. An Approach for Adaptive Enterprise Architecture.. In *ICEIS (2)*. 738–745.

[6] Digital.ai. 2021. 15th Annual State of Agile Report. (2021). https://info.digital.ai/rs/981-LQX-968/images/SOA15.pdf Accessed 03/10/2022.

[7] Sebastian Hanschke, Jan Ernsting, and Herbert Kuchen. 2015. Integrating agile software development and enterprise architecture management. In *2015 48th Hawaii International Conference on System Sciences*. IEEE, 4099–4108.

[8] Henk Jonkers, Marc M Lankhorst, Hugo WL ter Doest, Farhad Arbab, Hans Bosma, and Roel J Wieringa. 2006. Enterprise architecture: Management tool and blueprint for the organisation. *Information systems frontiers* 8, 2 (2006), 63–66.

[9] Marc Lankhorst et al. 2009. *Enterprise architecture at work*. Vol. 352. Springer.

[10] Ken Schwaber. 1997. Scrum development process. In *Business object design and implementation*. Springer, 117–134.

[11] Pedro Sousa, Ricardo Leal, and André Sampaio. 2018. Atlas: the enterprise cartography tool. In *Proceedings of 8th the Enterprise Engineering Working Conference Forum*, Vol. 2229.

[12] The Open Group. 2018. The TOGAF Standard, Version 9.2 - Introduction to Part II. https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap04.html https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap04.html. Accessed 14/10/2022.

[13] The Open Group. 2019. ArchiMate® 3.1 Specification. https://pubs.opengroup.org/architecture/archimate3-doc/ https://pubs.opengroup.org/architecture/archimate3-doc/. Accessed 18/09/2022.

[14] S Townson. 2008. Why does Enterprise Architecture Matter. *White Paper, The Open Group* (2008).

[15] Mohamed Watfa and Tarek Kaddoumi. 2021. A foundational framework for agile enterprise architecture. *International Journal of Lean Six Sigma* (2021).

[16] John A Zachman. 1997. Enterprise architecture: The issue of the century. *Database programming and design* 10, 3 (1997), 44–53.