INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




O Paradigma "Code Push-Down"

TIAGO LEÃO GOMES
Outubro de 2021

POLITÉCNICO
DO PORTO

The “Code Push-Down” Paradigm

Tiago Leão Gomes

Dissertation to obtain the master’s degree in Informatics Engineering,

Specialization in Software Engineering

Supervisor: António José Rocha De Oliveira

Porto, 2021

Resumo

A SAP é um dos maiores e mais bem-conceituados fornecedores de sistemas ERP. Tal como a maioria dos sistemas ERP, estes também têm estado em constante evolução.

Desde a disponibilização da sua base de dados SAP High-Speed Analytical Appliance (HANA), a SAP tem tentado persuadir os seus clientes a adotar esta base de dados.

Em 2018, a SAP anunciou que iria acabar o suporte do seu ERP, SAP ECC, favorecendo a adoção do seu novo ERP, SAP S/4 HANA, que apenas suporta o uso de bases de dados SAP HANA. O suporte estava previsto acabar em 2025, no entanto foi adiado para 2027 a pedido dos seus clientes.

O fim deste suporte significa que uma porção significativa dos clientes da SAP irão migrar para o ERP SAP S/4 HANA (+ SAP HANA) e, como recomendado pela SAP, provavelmente também irão adotar o paradigma de desenvolvimento "Code Push-Down", que se foca em empurrar lógica aplicacional para a camada/nível da base de dados.

Apesar desta mudança no paradigma de desenvolvimento poder, supostamente, trazer benefícios significativos de desempenho, também pode ter consequências no que toca às outras qualidades do software desenvolvido.

Este trabalho tem como objetivo analisar o paradigma de desenvolvimento "Code Push-Down", descobrir possíveis desvantagens/limitações e tentar elaborar um guião geral de como aplicar o paradigma de forma a tentar mitigá-las.

E talvez, ao suceder nos seus objetivos, também incentivar a realização de mais trabalhos sobre o tema.

Palavras-chave: "Code Push-Down", SAP ABAP, SAP HANA, Paradigma de Desenvolvimento, Guião.

Abstract

SAP is one of the biggest and most well-established ERP system providers. Like most ERP systems, their ERP systems and surrounding ecosystems have been in constant evolution.

Since the introduction of their SAP High-Speed Analytical Appliance (HANA) database, they have been pushing their clients towards its adoption.

In 2018, they announced the end of support for their SAP ECC ERP in favor of the new SAP S/4 HANA ERP, which only supports SAP HANA. This end of support was to take place in 2025 but, due to requests by their customers, it has since been extended to 2027.

This end of support means a significant portion of SAP's clients are migrating to SAP S/4 HANA (+ SAP HANA) and, as recommended by SAP, will most likely also adopt their "Code Push-Down" development paradigm, which is based around pushing application logic down to the database tier/layer.

Although this shift in development paradigms can, supposedly, bring significant gains in performance, it may also have consequences when it comes to other qualities of the developed software.

This work aims to analyze the "Code Push-Down" development paradigm, discover possible downsides/tradeoffs and try to provide general guidelines on how to apply it in order to possibly mitigate them.

And perhaps, by succeeding in meeting the objectives, to incentivize further work about this topic.

Keywords: "Code Push-Down", SAP ABAP, SAP HANA, Development Paradigm, Guidelines

Acknowledgments

To keep this short and sweet.

I would like to thank my family for all the support and love they give me, as without it I wouldn't be where I am.

I would like to thank my girlfriend for all the support, patience, and encouragement she has provided me.

I would also like to thank José Bento from Inetum, who was my mentor when I joined ROFF (now Inetum) and helped me with the infrastructure for the developments and the problem statement amongst others.

I would also like to thank my thesis advisor António Rocha for all the help and encouragement provided.

Finally, I would like to thank ISEP for the great master's degree from which I've learned so much.

Table of Contents

Table of Figures

List of Tables

Acronyms

ERP	Enterprise Resource Planning
SaaS	Software-as-a-Service
PaaS	Platform-as-a-Service
DBMS	Database Management System
ISO	International Organization for Standardization
CRUD	Create – Read – Update – Delete
OMG	Object Management Group
SQL	Structured Query Language
UC	Use Case
UR	Usability Requirement
SR	Supportability Requirement
IPR	Implementation Requirement
API	Application Interface

1 Introduction

This chapter intends to present an overview of this dissertation and the work done. Focusing on contextualizing the problem, describing the objectives set to overcome the problem and the motivation behind the work done. This chapter also describes the structure of the present document itself.

1.1 Context

Since their inception, enterprise resource planning (ERP) systems have brought a lot of value to enterprises [1] and, as such, their adoption has been increasing throughout the years, making it a multi-billion-dollar market [2], even when looking at “old” data.

One of the ERP system providers which has consistently had a significant market share throughout the years is SAP, as shown in Figure 1.

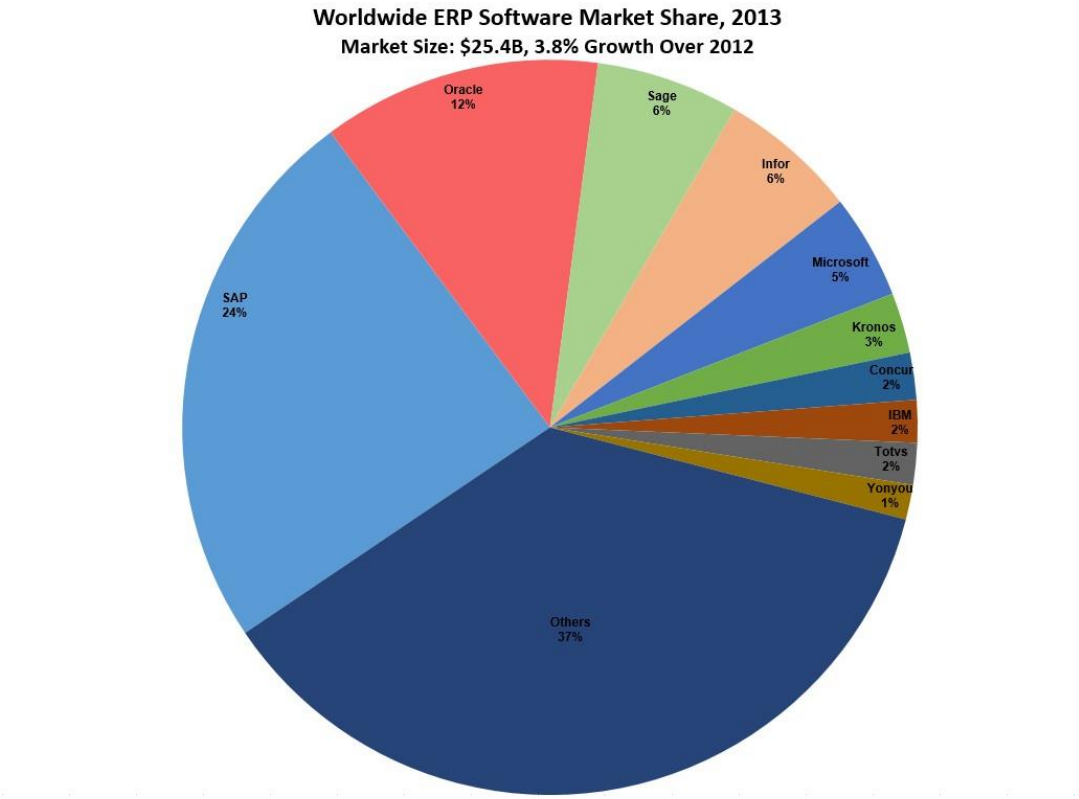


Figure 1 - ERP Market Share [3], [4]

According to SAP itself, it has “more than 440,000 customers in more than 180 countries” and “Today, 77% of all business transactions worldwide touch an SAP system. For example, SAP’s customers produce 78% of the world’s food products and 82% of the world’s medical devices.” [5]

Besides providing the usual Software-as-a-Service (SaaS), like most other ERP systems, SAP’s ERP systems give consumers the option of also using it as a Platform-as-a-Service (PaaS). This means they, themselves or by use of other organizations, can extend and/or develop their own software within their SAP ERP system by using the provided “tools” as they see fit.

Since its release in 2004, SAP’s “Flagship” ERP system has been SAP ECC. This ERP is database agnostic, meaning it can operate with different databases, such as Oracle, MySQL, and HANA.

In 2011, SAP’s high-performance analytic appliance (HANA) DBMS started to become available to its customers. This database, as the name implies, has a big focus on high performance and, its usage has been growing since.

In 2015, SAP released its new ERP system, SAP S/4 HANA, alongside other changes and improvements, this new ERP system focuses on leveraging the use of the SAP HANA DBMS and, therefore, stops supporting other DBMSs.

1.2 Problem

In 2018, SAP announced that they would be ending support for their older ERP systems, such as SAP ECC. The end of support date has been extended, since the announcement, to 2027, as of the release of the present document [6], [7]. For better context the release timeline can be observed in the figure below.

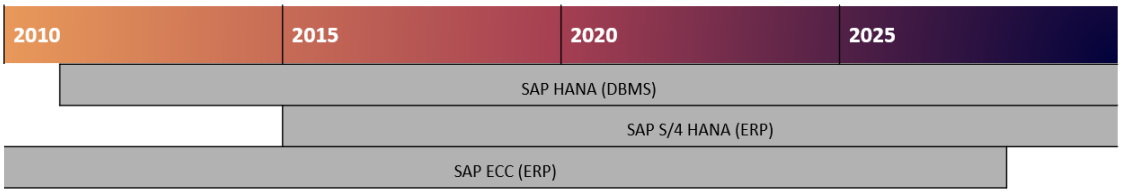


Figure 2 - SAP Release Timeline

Since the end of life announcement, in 2019, IDC conducted a survey with over 300 SAP customers, coming to the conclusion of: "It's not a matter of if companies will adopt SAP S/4 HANA anymore, but when."[8]

"Across a multitude of industries, IDC found that 73 percent of the surveyed businesses were planning to deploy and, 18 percent are currently deploying SAP S/4HANA. The remaining 9 percent of the companies stated they already have SAP S/4HANA in production. Customers were also asked about the planned timeline for their transition to our leading ERP system. According to the survey, 54 percent of SAP customers say they will make the switch within three years."[8] This last part of the survey can be observed in the figure below.

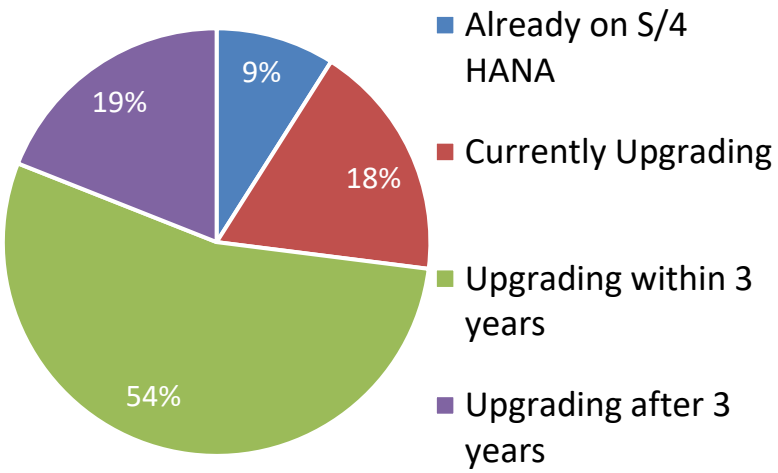


Figure 3 - IDC 2019 SAP S/4 HANA Survey

By ending the support of its older ERP systems, SAP will “force” their clients to migrate to their new ERP systems and therefore to the adoption of the SAP HANA DBMS as well.

Now this by itself raises quite a few different issues/questions regarding the migration process, the compatibility of new systems with old “custom” software, possible database lock in, etc.

And although these are important concerns that also deserve to be worked on, the present work focuses on the following.

With the migration to the new SAP S/4 HANA ERP system and SAP HANA DBMS, SAP recommends the adoption of their “Code Push-Down” developing paradigm, to better leverage the, alleged, high performance of their SAP HANA DBMS. [9], [10]

This new, to the SAP ecosystem, development paradigm seems to go against current development trends and patterns by “blurring” the segregation between traditional application/business and data layers and/or tiers while, possibly, compromising the solution’s architectural quality for performance benefits.

1.3 Objectives

The present work has the following objectives:

- Understand the current development paradigm, within the SAP Ecosystem;
- Understand SAP’s take on the “Code Push-Down” development paradigm and its tools;
- Apply & compare both development paradigms;
- Identify potential compromises & problems with the application of “Code Push-Down”;
- Develop “General Guidelines” on how to view/apply the “Code Push-Down” development paradigm (and hopefully mitigate any identified problems/compromises).

1.4 Document Structure

The present document follows the following structure:

1. **Introduction**: Problem context, identification and motivation that drives the present work;
2. **Background**: Explanation and definition of some concepts that may be helpful to better understand the work done;
3. **State of the Art**: Summary of the current context and tools existent to try to solve the problem(s) at hand;
4. **Value Analysis**: Description of the value analysis done for this work;
5. **Simulated Problem Statement**: A problem to be used to develop two solutions;
6. **Analysis**: Analysis and requirement engineering of the problem statement;
7. **Design**: Design of both solutions;
8. **Development**: Development of both solutions;
9. **Comparison**: Evaluation and comparison of both developed solutions;
10. **Guidelines**: Conclusions about the developed solutions and guidelines developed based on those conclusions;
11. **Conclusions**: Further conclusions about the developed work, opinions, improvements, and future work.

2 Background

This chapter focuses on explaining and defining some concepts and tools whose grasp may be helpful to better understanding the rest of the document.

2.1 ERP Systems

Taking into consideration the context of this document, this definition will be given from SAP's point of view.

Enterprise Resource Planning (ERP) Systems are systems that are composed of multiple applications, also commonly referred to as "modules", tightly connected that usually share a single database. Each application (or module) typically focuses on one business area (Finance, HR, manufacturing, Supply Chain, etc.). In SAP's ERP Systems it is usually possible to combine different modules as seen fit by the customer to better suit their needs.[11]

There are multiple types of ERP Systems from various providers, but SAP currently offers these types as to better suit the customer's needs[11]:

- Cloud ERP Systems;
- On-Premises ERP Systems;
- Hybrid ERP Systems.

Depending on the type of ERP system the client has, the provided solution can be what is classified as a Software-as-a-Service (SaaS) all the way to what is classified as a Platform-as-a-Service (PaaS), where additional software development/integration is supported. This specific type of ERP Systems is the one this work focuses on.

2.2 Database & Database Management System (DBMS)

As simply explained by Oracle, “A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Data within the most common types of databases in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. The data can then be easily accessed, managed, modified, updated, controlled, and organized. Most databases use structured query language (SQL) for writing and querying data.”[12]

A DBMS is an interface between the database and its end users or programs, allowing users to control, organize and optimize the data. A DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, and backup and recovery.[12]

2.3 Database Migration

As defined by Google, “Database migration is the process of migrating data from one or more source databases to one or more target databases by using a database migration service. When a migration is finished, the dataset in the source databases resides fully, though possibly restructured, in the target databases. Clients that accessed the source databases are then switched over to the target databases, and the source databases are turned down.”[13]

The following diagram illustrates this process in a simple way:

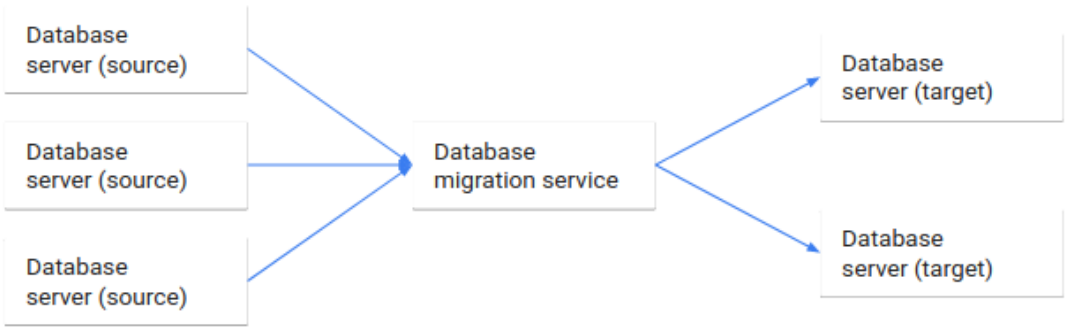


Figure 4 - Database Migration Diagram[14]

3 State of the Art

In order to achieve and understand what good solutions to the problem at hand can be, we need to understand the current state of the art.

Taking into consideration the current problem at hand and its context, it is pertinent to understand:

- The current state of the art of software architecture, with a focus on the common "Standard" practices within the SAP ecosystem;
- The state of the art when it comes to software development infrastructure and tools in an SAP context;
- The state of the art when it comes to both the "Standard" development paradigm and the "Code Push-Down" development paradigm.

3.1 Software Architecture

This subchapter focuses on explaining the concept of software architecture and its current state, in this work's context.

We can define software architecture, in the context of application development, as a set of patterns, techniques and best practices. This set of patterns, techniques and best practices should be specifically chosen as to best guide the development process in order to assure that the main desired qualities for the solution are achieved.[15]–[17]

Taking this definition into account, let us focus on the software qualities that may end up being compromised with the application of the "Code Push-Down" paradigm.

Knowing that this paradigm is centered around "pushing" the application processing load "down" to the database, we can narrow down the scope of the qualities, as defined by ISO, to the most likely to be affected by this change in responsibilities, to the ones highlighted in the figure below:

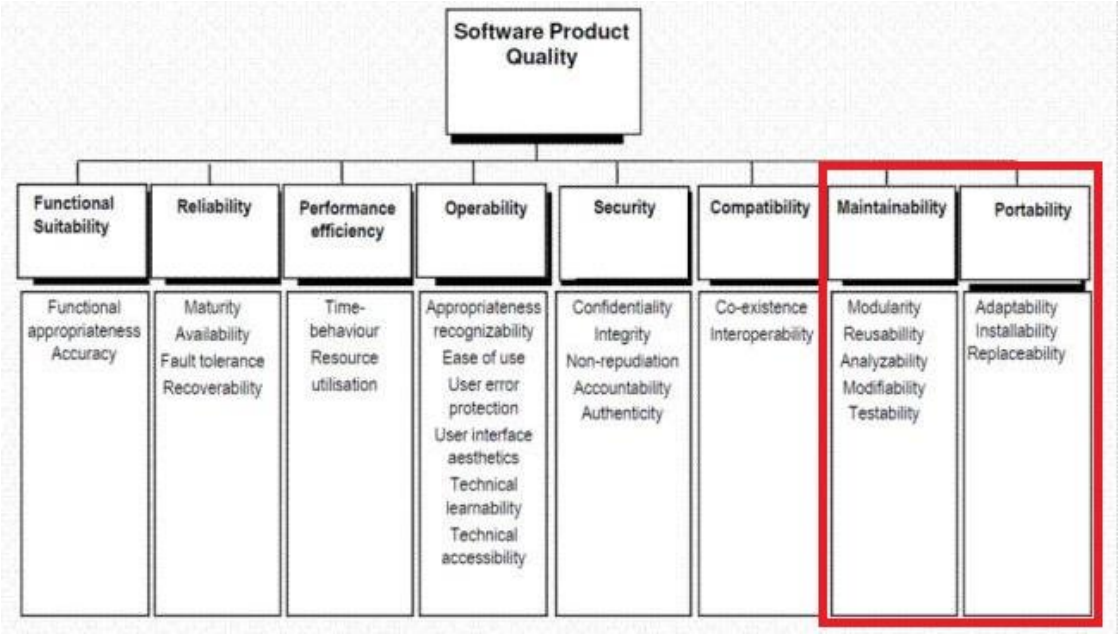


Figure 5 - Product Quality Model[18]

- **Maintainability**: The degree of effectiveness and efficiency with which a product or system can be modified (these modifications can include corrections, improvements or adaptations of the software to changes in environment, and in requirements and/or functional specifications) by the people/systems maintaining it, whoever they might be.[19]
    - **Modularity**: The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components. This attribute is usually a consequence of high cohesion and low coupling;[19]
    - **Reusability**: The degree to which an asset can be used in more than one system, context or in building other assets;[19]
    - **Analyzability**: The degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified;[19]

    - **Modifiability**: The degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality;[19]

    - **Testability**: The degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.[19]

- **Portability**: The degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another.[19]

    - **Adaptability**: The degree to which a product or system can effectively and efficiently be scaled or adapted for different or evolving hardware, software or other operational or usage environments;[19]

    - **Installability**: The degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment;[19]

    - **Replaceability**: The degree to which a product can replace another specified software product for the same purpose in the same environment.[19]

Keeping this in mind, in this subchapter we will focus on the patterns, practices, guidelines, techniques that focus and/or impact directly the mentioned quality attributes.

It is also important to note that, there are countless patterns, practices, guidelines, and techniques within the software architecture scope. But since the present work focuses on the SAP development landscape, the scope will be narrowed down to focus on tiered/layered architectures, as these are the core development principles applied in the SAP development landscape.

3.1.1 Tiered Architectures

These architectural styles are some of the most commonly used architectural patterns in software development for traditional client-server applications as it can, usually, create value and improve modularity and testability attributes.

These architectural styles are of particular importance to the present work as the changes, proposed by the “Code Push-Down” paradigm, are largely targeted to, considerably, change the way these architectural styles are applied.

3.1.1.1 3-Tier

The 3-tier architecture organizes applications into the following three logical and physical computing tiers, as shown in the figure below:

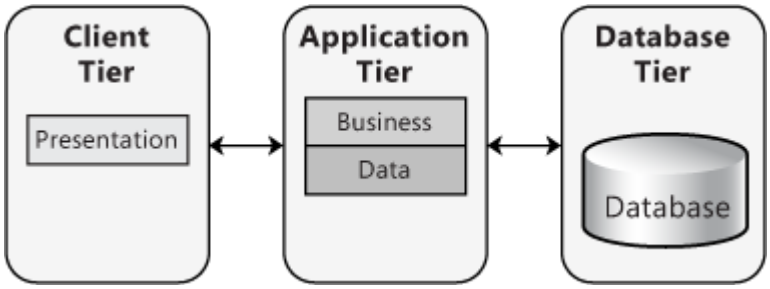


Figure 6 - 3-Tier Deployment[20]

- **Presentation Tier**: The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier usually runs on a web browser, as a desktop application or a graphical user interface (GUI) of some sort;[21], [22]

- **Application Tier**: The application tier, also known as, the logic or business tier, is the "heart" of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also perform CRUD operations to the data in the data tier; [21], [22]

- **Database Tier**: The database tier, or sometimes called data tier, is where the information processed by the application is stored and managed. This can be a relational database management system or a NoSQL Database server.[21], [22]

It is important to note that, in a 3-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another. [22]

The main benefits of this type of architecture are the level of modularity, and consequent testability provided by the tier's inherent high cohesion and low coupling. Additionally, another big benefit of 3-tier architecture is that, because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team and can be updated or scaled as needed without impacting the other tiers.[22]

3.1.1.2 N-Tier Architecture

The N-tier architecture style, also called multi-tier architecture, refers to any application architecture with more than one tier. But applications with more or less than three layers are much less common. As a result, n-tier architecture and multi-tier architecture are usually synonyms for three-tier architecture. The figure below represents an n-tier architecture.

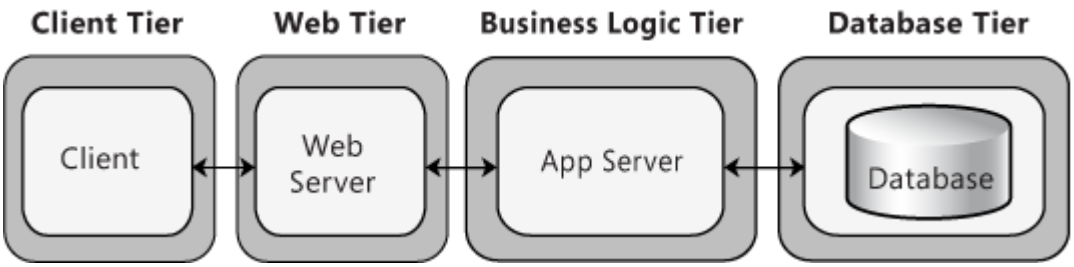


Figure 7 - 4-tier deployment[23]

3.1.2 Layered Architecture Style

This architectural style is one of the most commonly used architectural patterns in software development as it can, usually, create value and improve modularity and testability attributes in a lot of varying contexts.

This architectural style is of particular importance to the present work, as the changes proposed by the “Code Push-Down” paradigm, are largely targeted to, considerably, change the way this architectural style is applied.

3.1.2.1 Layered Architecture

This architectural style focuses on segregating and encapsulating a solution, or part of it, into broad functional layers. These layers, sometimes also referred to as modules, represent large portions of the solution which are, typically, loosely coupled between each other and their scope tends to be clearly defined.[24] Usually there are at least three functional layers, as shown in the figure below.

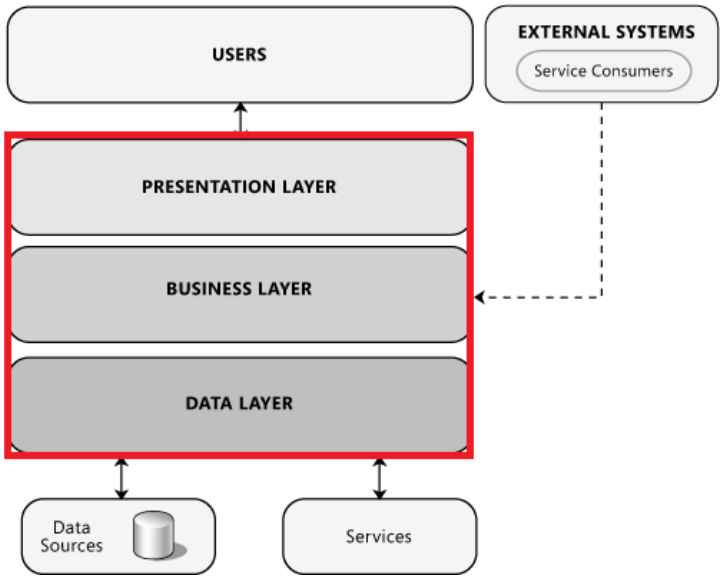


Figure 8 - The logical architecture view of a layered system[25]

- **Presentation**: The presentation layer corresponds to the user interface portion and part of the communication portion of the solution. It's main responsibility is to display information and turn user interaction into actionable requests for the domain layer;[26]

- **Domain**: The domain layer, also usually called as business layer, typically harbors most, if not all, business logic, rules, and domains that the solution provides. Its purpose is to process incoming requests by applying the business logic to the data provided by the data layer;[26]

- **Data**: The data layer is typically responsible for the data management portion of the solution, from data operations requested by the domain layer to dealing with the data in storage.[26]

These 3 layers tend to be a particularly common way of layering a solution particularly due to the inherent low coupling and high cohesion that these modules of the solution tend to have. That said, it is also common to find different and/or additional layers like Services or Controller layer depending on the context and the module of the solution being layered.[26], [27]

3.1.2.2 Tier vs. Layer

When discussing Tiered Architectures and Layered Architectures, the terms and concepts of "tier" and "layer" tend to, mistakenly, be used interchangeably. That said, they are not the same. A "layer" refers to a functional division of the software, but a "tier" refers to a functional division of the software that runs on infrastructure separate from the other divisions. And although all the layers of each component of a solution tend to be in the same tier, it is not always the case and they do not have to be confined to the same tier.[21], [22]

For example, a simple desktop application like a calculator app can have multiple layers but only a single tier.

3.1.3 General responsibility assignment software principles (GRASP)

General Responsibility Assignment Software Principles/Patterns are a set of guidelines to better assign responsibilities to classes and objects in object-oriented design. These guidelines aim to reduce the ambiguity of the responsibility assignment process by doing it in a methodical, rational, and comprehensive way. [28], [29]

Before diving into the principles let's define the concept of responsibility. OMG defines responsibility as "A contract or an obligation of an element in its relationship to other elements".[30] And these responsibilities/obligations tend to be of the following types:

- **Doing** Responsibilities:
    - Doing something itself, such as creating an object or doing a calculation;
    - Initiating actions in other objects;
    - Controlling and coordinating activities in other objects.
- **Knowing** Responsibilities:
    - Knowing about private encapsulated data;
    - Knowing about related objects;
    - Knowing about things it can derive or calculate.[28]

These types of responsibilities are the most usual reasons as to why you should attribute a responsibility to a specific object. The information to determine these responsibilities is usually gathered from the domain.[28]

These are the 4 main patterns of GRASP, as these are the ones that address the most common questions and fundamental design issues[28]:

3.1.3.1 Information Expert (or Expert)

Problem

What is the general principle of assigning responsibilities to objects?

An application may require hundreds or thousands of responsibilities to be fulfilled. At various stages we make choices about the assignment of responsibilities to software classes. Done well, systems tend to be easier to understand, maintain, and extend, and there is more opportunity to reuse components in future applications.[28], [29]

Solution

Assign a responsibility to the information expert. This being the class/object that has the information necessary to fulfill the responsibility.

This guideline tries to represent common “intuition” as objects should be related to the information they have. Notice that the fulfillment of a responsibility often requires information that is spread across different classes of objects. This implies that there are many "partial" information experts who will collaborate in the task. [28], [29]

Contraindications

There are situations where a solution suggested by Expert is undesirable, usually because of problems in coupling and cohesion (these principles are discussed later). These problems, usually, indicate violation of a basic architectural principle(s): design for a separation of major system concerns. Keep application logic in one place (such as the domain software objects), keep database logic in another place (such as a separate persistence services subsystem), and so forth, rather than intermingling different system concerns in the same component.[28], [29]

Key Benefits

- Information encapsulation is maintained since objects use their own information to fulfill tasks. This usually supports low coupling, which leads to more robust and maintainable systems;[28]

- Behavior is distributed across the classes that have the required information, thus encouraging more cohesive "lightweight" class definitions that are, usually, easier to understand and maintain, promoting high cohesion.[28]

3.1.3.2 Low Coupling

Problem

How to support low dependency, low change impact, and increased reuse?

Coupling is the measure of how strongly one element is connected to, has knowledge of, or relies on other elements. An element with low (or weak) coupling is not dependent on too, relatively, many other elements. These elements can be classes, subsystems, systems, and so on. [28], [31]

An element with high (or strong) coupling relies on many other elements. Such elements may become undesirable as its likely they will suffer from the following problems [28], [31]:

- Changes in related elements force local changes;
- Harder to understand in isolation;
- Harder to reuse because its use requires the additional presence of the elements on which it is dependent.

Solution

Assign responsibilities in a way that coupling remains low.

Low Coupling is a principle to keep in mind during all design decisions. It is an evaluative principle that a designer applies while evaluating all design decisions. Common forms of coupling from TypeX to TypeY include [28], [31]:

- TypeX has an attribute (data member or instance variable) that refers to a TypeY instance, or TypeY itself;
- A TypeX object calls on services of a TypeY object;
- TypeX has a method that references an instance of TypeY, or TypeY itself, by any means. These typically include a parameter or local variable of type TypeY, or the object returned from a message being an instance of TypeY;
- TypeX is a direct or indirect subclass of TypeY;
- TypeY is an interface, and TypeX implements that interface.

Low Coupling encourages assigning a responsibility so that its placement does not increase the coupling to such a level that it leads to the negative results that high coupling can produce.[28], [31]

Contraindications

Unless taken to extremes, low coupling is seldom a problem.

Key Benefits

- Not affected by changes in other components;
- Simple to understand in isolation;
- Convenient to reuse.

3.1.3.3 High Cohesion

Problem

How to keep complexity manageable?

Cohesion (or more specifically, functional cohesion) is a measure of how strongly related and focused the responsibilities of an element are. An element with highly related responsibilities, has high cohesion. These elements include classes, subsystems, and so on. [28], [31]

A class with low cohesion does many unrelated things. Such classes are undesirable as they, usually, suffer from the following problems [28], [31]:

- Hard to comprehend;
- Hard to reuse;
- Hard to maintain;
- Delicate. Constantly affected by change;
- Low cohesion classes often represent a very "large grain" of abstraction or have taken on responsibilities that should have been delegated to other objects.

Solution

Assign a single responsibility so that cohesion remains high.

Like Low Coupling, High Cohesion is a principle to keep in mind during all design decisions. It is an evaluative principle that a designer applies while evaluating all design decisions. [28], [31]

As a rule of thumb, a class with high cohesion has a relatively small number of methods, with highly related functionality, and does not do too much work. It collaborates with other objects to share the effort if the task is large.[28], [31]

A class with high cohesion is advantageous because it is relatively easy to maintain, understand, and reuse. The high degree of related functionality, combined with a small number of operations, also simplifies maintenance and enhancements. The fine grain of highly related functionality also supports increased reusability.[28], [31]

Contraindications

There are a few cases in which accepting lower cohesion is justified.

One case is the grouping of responsibilities or code into one class or component to simplify maintenance by one person although, be warned, that such grouping may also make maintenance worse.[28], [31]

Another case for components with lower cohesion is with distributed server objects. Because of overhead and performance implications associated with remote objects and remote communication, it is sometimes desirable to create fewer and larger, less cohesive server objects that provide an interface for many operations.[28], [31]

Key Benefits

- Clarity and ease of comprehension of the design is increased;
- Maintenance and enhancements are simplified;
- Low coupling is often supported;
- The fine grain of highly related functionality supports increased reuse because a cohesive class can be used for a very specific purpose.

3.2 SAP Technologies & Development Tools

This subchapter focuses on explaining the current state of the art of the SAP technology and development tools landscape.

Although SAPs ecosystem is quite vast and diverse, this subchapter will focus on the core technologies and tools used for more “standard” and common developments within the SAP landscape.

3.2.1 SAP ERP System’s Architecture

Since 1992, with the introduction of SAP R/3, until more recently with the introduction of SAP S/4 HANA, SAP’s “on-premise” ERP systems have followed client-server and 3-Tier architectures [32], [33], as shown in the figure below.

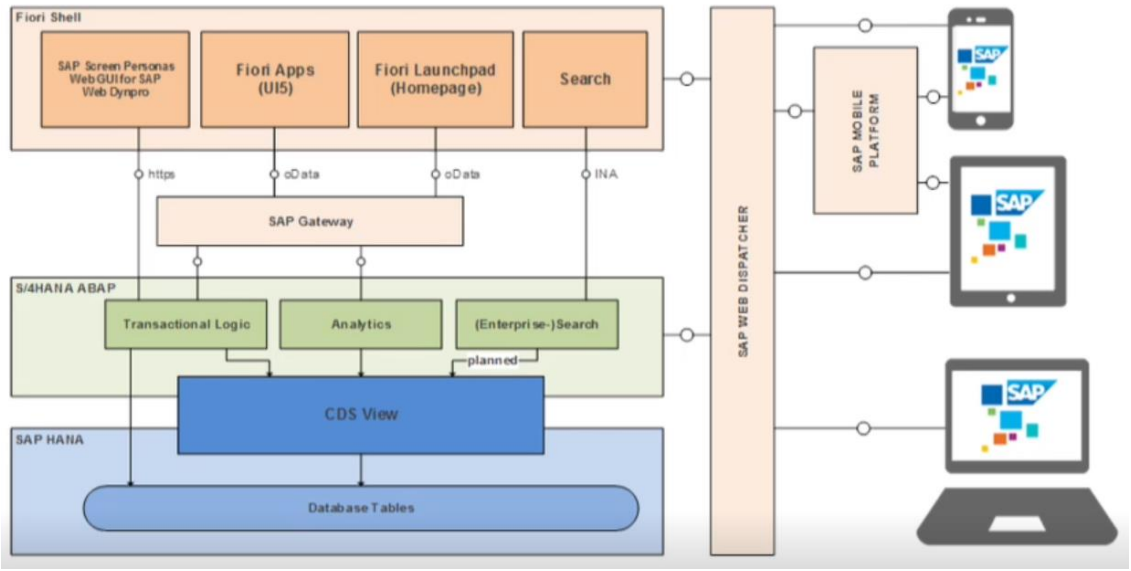


Figure 9 - SAP S/4 HANA 3-Tier Architecture[34]

Their version of the 3-Tier architecture follows the same pattern of the architecture explained in the subchapter 3-Tier.

3.2.2 SAP ABAP

This subsection focuses on explaining the current state of the art regarding the SAP Advanced Business Application Programing (ABAP) language and its tools.

3.2.2.1 ABAP Language / Application Server ABAP

SAP ABAP is a programming language developed by SAP for the development of business applications in the SAP environment.[35]

A prerequisite for the use of the ABAP programming language is the installation and use of an Application Server ABAP. Most of the components of an AS ABAP can be organized in the layers (presentation, application, and database) of a three-tier client-server architecture in accordance with their tasks.[35]

- The presentation layer is distributed to the workstations of individual users and represents the user interface of an AS ABAP (SAP GUI or Web browser);
- The application layer is implemented using one or more application servers. The application layer contains the ABAP runtime environment in which ABAP programs are executed;
- The database layer consists of a database system in which the central dataset of an AS ABAP is saved.

ABAP supports the following programming models [35]:

- An object-oriented programming model based on classes and interfaces;
- A procedural programming model based on function modules and subroutines.

Both programing models can be used simultaneously and interchangeably although SAP recommends the use of the object-oriented approach.[35]

3.2.2.2 Database Access

One of the fundamental properties of ABAP as a programming language for business applications is that access to database and data structures is fully integrated into the language. [35]

This integration can be split into two main parts:

- **Data Modeling**: ABAP data modeling makes it possible to create data models for business applications. ABAP has two major ways modeling data:
    - **ABAP Dictionary**: This is a persistent repository for data types and their dependencies;
    - **ABAP Core Data Services (ABAP CDS)**: This expands the ABAP Dictionary by adding an implementation of the CDS concept for AS ABAP.
- **Database Access**: Access to data in the database is fully integrated in ABAP. This data can be directly accessed using two different methods:
    - **Open SQL**: This makes it possible to access database objects defined in ABAP Dictionary or ABAP CDS and data from the database;
    - **Native SQL**: makes platform-specific access to databases possible.

3.2.2.3 ABAP Dictionary

The ABAP Dictionary is used to create and manage data definitions (metadata). It allows for a central description of all the data used in the system without redundancies. New or modified information is automatically provided for all the system components. This ensures data integrity, data consistency and data security.[35]–[37]

The ABAP Dictionary supports the definition and/or creation of the main following objects:

- Tables;
- Views;
- Types;
- Lock Objects;
- Domains.

It describes the logical structure of the objects used in application development and shows how they are mapped to the underlying relational database in tables or views. It also provides

standard functions for editing fields on the screen, for example for assigning input help to a screen field. [35]–[37]

The ABAP Dictionary is completely integrated in the ABAP Workbench. The SAP system works interpretatively, permitting the ABAP Dictionary to be actively integrated in the development environment. [35]–[37]

3.2.2.4 Open SQL

As said by SAP, Open SQL is the umbrella term for a subset of SQL realized using ABAP statements, including the Data Manipulation Language (DML) part. The Open SQL statements use the Open SQL interface of the database interface to access an AS ABAP database. [38]

The Open SQL Interface is part of the database interface that is responsible for Open SQL commands. The Open SQL interface converts all Open SQL commands that access the central database of an AS ABAP to manufacturer-specific SQL and forwards this to the database system. [39]

This is important to know for the present work because it means that the Open SQL expressions are passed to the database system, executed there, and the result is passed to the application server if necessary. [40]

3.2.3 Core Data Services (CDS)

This subsection focuses on explaining the current state of the art of the core data services (CDS) developed by SAP.

3.2.3.1 ABAP Core Data Services (ABAP CDS)

ABAP Core Data Services (CDS) are a data dictionary infrastructure that was introduced with SAP AS ABAP 7.40. It allows for the definition and consumption of semantically rich data models. The CDS framework, although database agnostic, was introduced to leverage the computational power of HANA DB as some of the main tools for the application of the “Code Push-Down” paradigm.[41]–[45]

As such, although ABAP CDS are platform independent, it does not mean that the performance, when accessing CDS entities, is the same for all platforms.[45]

The structure of these abstract data models is mapped directly to entities in the database. At the same time, the application logic is moved from the application server to the database server ("“Code Push-Down”"). This means that the data is processed where it is saved, which in principle should result in an improvement in performance.[46]

For the purpose of defining and consuming data models, the CDS framework has been enhanced by SQL with the languages shown in the figure below:[42], [44], [45], [47]

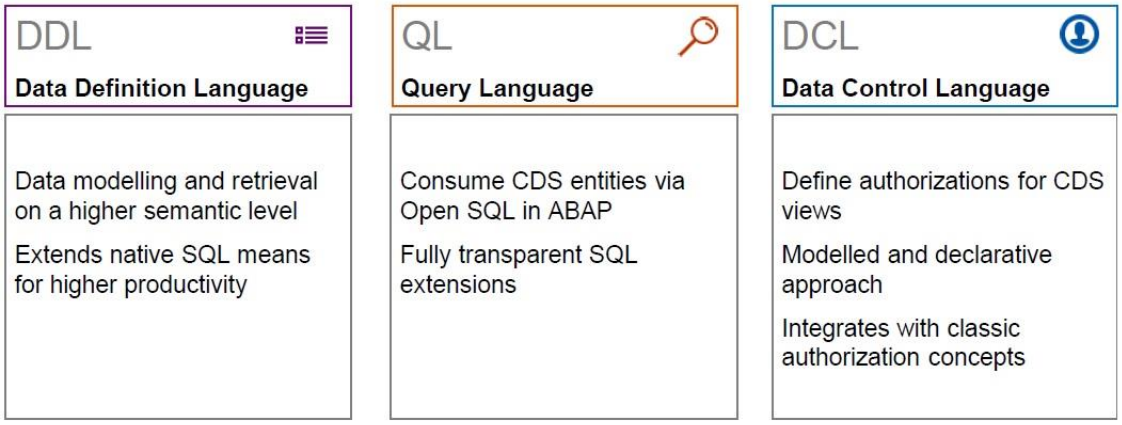


Figure 10 - CDS Languages [48]

Currently, this framework is focused on leveraging the functionalities of the data definition language (DDL), as shown in the figure below, to create what are called CDS views and CDS table functions (usually referred to as AMDP).[44], [45], [47], [49]

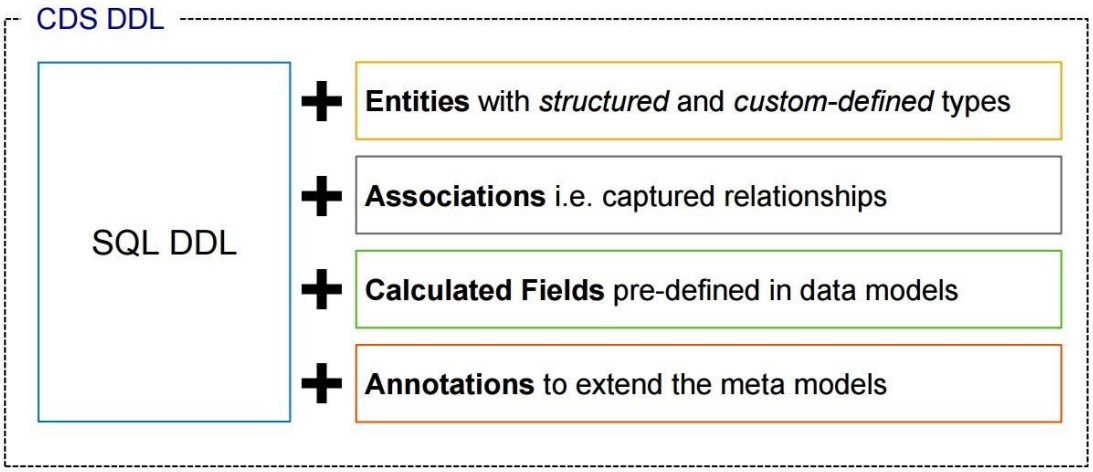


Figure 11 - CDS DDL [50]

- **CDS Views:** A CDS view is defined for existing database tables and any other views or CDS views in ABAP Dictionary using the CDS DDL in ABAP Core Data Services (CDS) [51]. For the development of CDS views, the use of ABAP Development Tools (ADT) is required;

- **CDS Table Functions:** A CDS table function is defined in CDS source code of a CDS data definition in the ABAP Development Tools (ADT) using the ABAP Core Data Services (CDS) DDL. CDS table functions can only be used in a database system that supports AMDP (ABAP Managed Database Procedures)[52]. Since its release, it seems that the SAP HANA DB is still the only one supporting these AMDP [53], [54].

One thing to note is that, according to the SAP's AMDP Programing Guidelines, "The use of AMDP is not recommended if the same task can be achieved using Open SQL (or ABAP CDS)." [53], [55] as "AMDP should be used only if it enables database-specific functions to be accessed that do not exist in Open SQL (see the example for Currency Conversion) or if large process flows or analyses that incur repeated transports of large amounts of data between the database and the application server can be swapped out." [53], [55].

As stated previously, these ABAP CDS objects are maintained and managed by the ABAP Dictionary, ending up extending its original functionality.

3.2.3.2 HANA Core Data Services (HANA CDS)

Much like the ABAP CDS, HANA Core Data Services (CDS) are a data dictionary infrastructure that was introduced with the SAP HANA DBMS for the definition and consumption of semantically rich data models. The CDS framework was introduced to leverage the computational power of HANA DB for the application of the "Code Push-Down" paradigm.[56]–[58]

3.2.3.3 ABAP CDS vs. HANA CDS

Although these core data services serve very similar purposes, they are not the same, and are not, usually, interchangeable. If you know the DDL of CDS, you should be able to understand definitions of both CDS entities.

The main difference is that HANA CDS, being proprietary to the SAP HANA DB, can use specific functions and operations (such as specific arithmetic & cast expressions) that are not necessarily available on other databases.[41], [56]

3.3 SAP Software Development

This subchapter focuses on explaining the current state of the art of software development within SAP ERP systems.

The focus will be on their “On-premises” ERPs that are platforms (SAP R/3, SAP ECC and SAP S/4 HANA), allowing for the development of software within themselves.

3.3.1 SAP “Standard” Development Paradigm

Although there can be some variation to the architecture of developments within the SAP ERP platform, due to the architecture of the system itself and the way the ABAP dictionary is integrated within the SAP ERP system, the “Standard” development follows a typical layered & tiered architecture, as shown in the figure below.

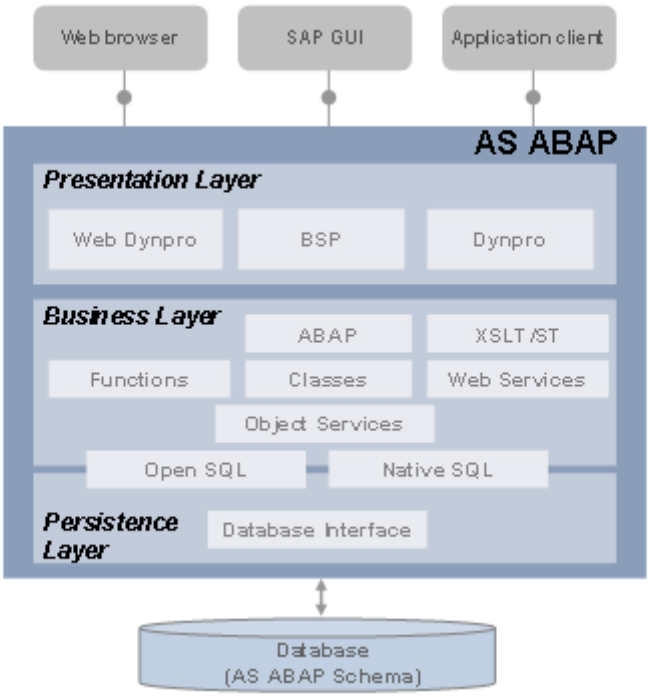


Figure 12 - "Standard" SAP Development Architecture[59]

These layers have the exact same purpose as the ones detailed in the subchapter “Layered Architecture”

These tiers have the exact same purpose as the ones detailed in the subchapter “Tiered Architectures”

3.3.2 "Code Push-Down" Development Paradigm

As stated previously in this document, SAP recommends the adoption of their "Code Push-Down" developing paradigm, to better leverage, the alleged, high performance of their SAP HANA DBMS. [9], [10]

This new, to the SAP ecosystem, development paradigm was introduced in SAP NetWeaver Application Server ABAP 7.4 (NW AS ABAP 7.4) as Core Data Services (CDS).[10] These CDS are the main tools, alongside ABAP + Open SQL, to be used for the application of the "Code Push-Down" paradigm (although Open SQL was already present before, the way it is to be applied changes when applied to the "Code Push-Down" paradigm).

The focus of this development paradigm is to be applied on/over the SAP "Standard" Development Paradigm by, as the name implies, "pushing" the application code/logic "down" to the database tier. This is done, mainly, to exploit the following traits in search of performance:[10], [42]

- Use the, alleged, performance of the SAP HANA database to do most of the heavy processing tasks as possible.
- Reduce the amount of data to be transferred between tiers and layers.

The figure below shows an overview of the "Code Push-Down" development paradigm as a whole.

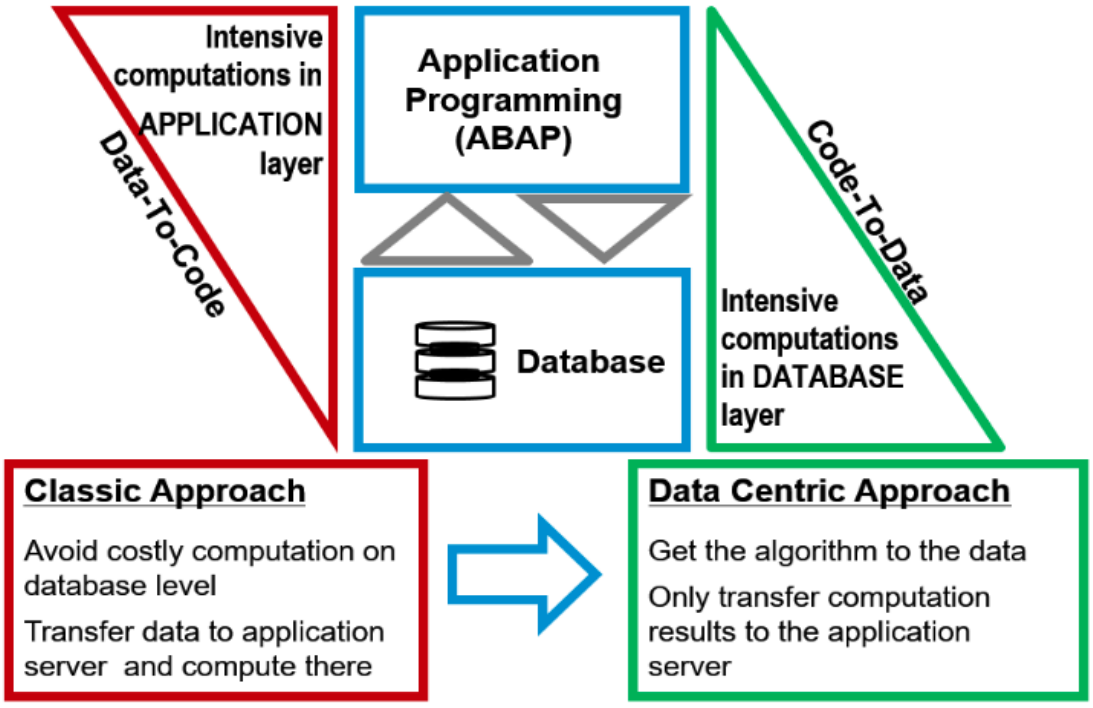


Figure 13 - Code Push-Down[42]

It is also important to note that although SAP does not provide public specific guidelines/processes, at least none were found, on how to choose what type of data/processes to apply the "Code Push-Down" paradigm to, they, however, do state that: *"Code pushdown means delegating data intense calculations to the database layer. It does not mean push ALL calculations to the database, but only those that make sense."*[10], [42]

So, according to SAP, the result should be that only a portion of the application logic should be pushed down, as shown in the figure below. That said, how big of a portion or what could make sense is up to the developer(s).

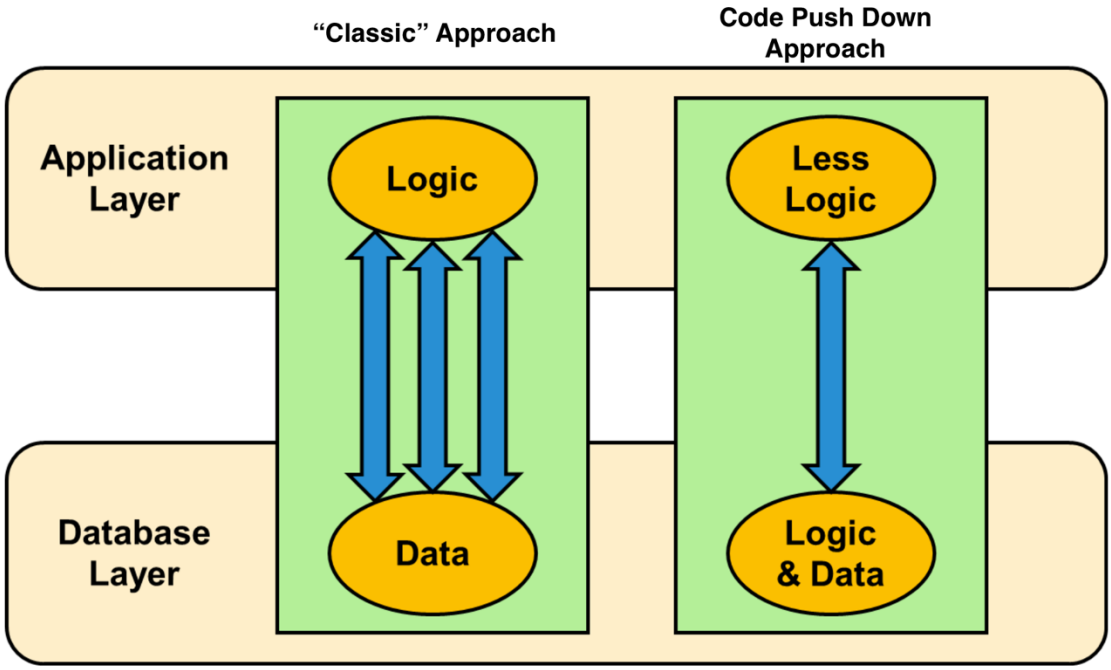


Figure 14 - Code Push-Down Result[42]

Now, focusing on the changes to the use of ABAP + Open SQL to apply the "Code Push-Down" paradigm. Traditionally its use, usually, resulted in "simpler" queries and the remaining business logic, regarding what to do with the data, was done with ABAP. Now, the logic should be integrated with the Open SQL queries themselves, as this, as explained before, will result in the business/application logic being push down and executed in the database.

If using a CDS (ABAP CDS or HANA CDS), depending on its type, the "Code Push-Down" paradigm can be applied in one of two ways.

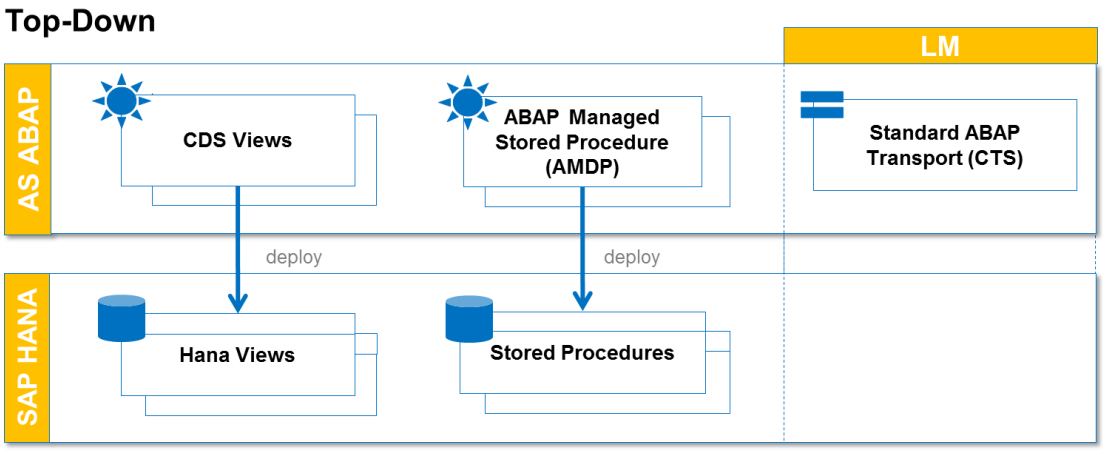


Figure 15 - Top-Down approach to “Code Push-Down”[60]

This approach, as shown in the figure above, is the approach based on the use of ABAP CDS, and is the one recommended by SAP for most cases [10]. This is due to ABAP CDS objects being maintained in the ABAP dictionary.

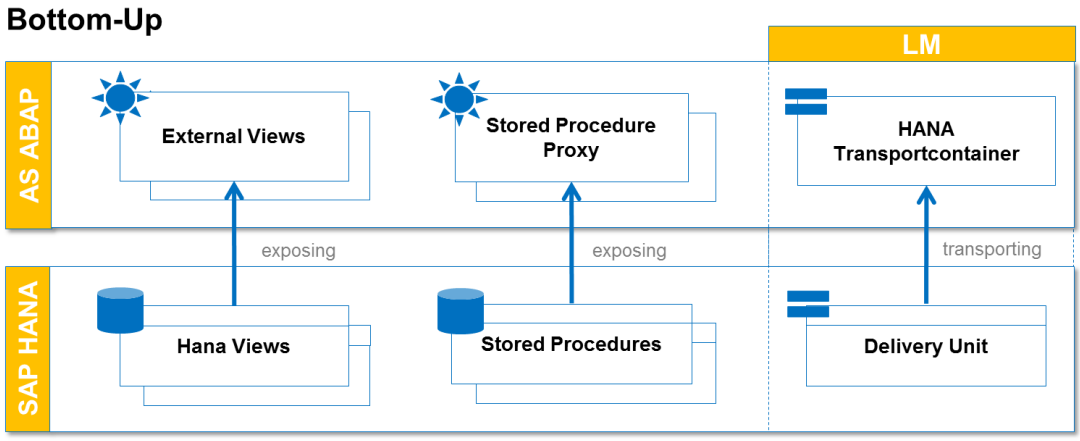


Figure 16 - Bottom-Up approach to “Code Push-Down”[61]

This approach, as shown in the figure above, is the approach based on the use of HANA CDS, and although it can be used, it should only be used when HANA CDS are required.[10] This is due to HANA CDS being on the HANA database itself.

One thing to also keep in mind is that, as was implied before, this paradigm, although new in the SAP ecosystem, is not new to software development in general. One example of this is Oracle, who had the following to say regarding the “Code Push-Down” paradigm:

“SAP used to think of a database as a dumb data store. Whenever a user wants to do something useful with the data, it must be transferred, because the intelligence sits in the SAP Application Server. The disadvantages of this approach are obvious: If the sum of 1 million values needs to be calculated and if those values represent money in different currencies, 1 million individual values are transferred from the database server to the application server – only to be thrown away after the calculation has been done.

As a response to this insight, SAP developed the “Code Push-Down“ strategy: push down code that requires data-intensive computations from the application layer to the database layer. They developed a completely new programming model that allows ABAP code to (implicitly or explicitly) call procedures stored in the database. And they defined a library of standard procedures, called SAP NetWeaver Core Data Services (CDS).

20 years earlier, Oracle had already had the same idea and made the same decision. Since version 7 Oracle Database allows developers to create procedures and functions that can be stored and run within the database. It was therefore possible to make CDS available for Oracle Database as well, and today SAP application developers can make use of it.”[62]

Another thing to keep in mind is that, for some time now, software development in general has been moving away from this type of development paradigms. Possibly meaning that, generally, the benefits of these paradigms may not outweigh the drawbacks.

4 Value Analysis

Value analysis can be defined as a systematic, formal, and organized process of analysis and evaluation that takes into account the function of a product and its ability to fulfill its purpose being used by the customer(s). [63]

So, in order for a product to create/have value for the customer, it must meet their needs by, usually, solving/preventing need/problems, reducing costs and/or improving product performance. Otherwise, it may bring no value to the customer.

In this chapter, the value analysis will be done according to the "New Concept Development Model" while the value proposition part will be demonstrated according to the business model canvas.

4.1 New Concept Development Model (NCD)

The New Concept Development Model (NCD) is a model that attempts to provide a common language and insights to the Fuzzy Front-End Concept (FFE). [64], [65]

The Fuzzy Front End (FFE) being "activities that come before the formal, and well structured, New Product and Process Development (NPPD)" [65].

The NCD model consists of three key parts, as shown in the diagram below:

- The engine or the portion of the diagram is the leadership, culture, and business strategy of the organization that drives the five key elements that are controllable by the corporation [64], [65];

- The inner gray area defines the five controllable activity elements (opportunity identification, opportunity analysis, idea generation and enrichment, idea selection, and concept definition) of the FFE [64], [65];

- The influencing factors, on the outside of the diagram, consist of organizational capabilities, the outside world (distribution channels, law, government policy, customers, competitors, and political and economic climate), and the enabling sciences (internal and external) that may be involved. These factors affect the entire innovation process through to commercialization. These influencing factors are relatively uncontrollable by the corporation [64], [65].

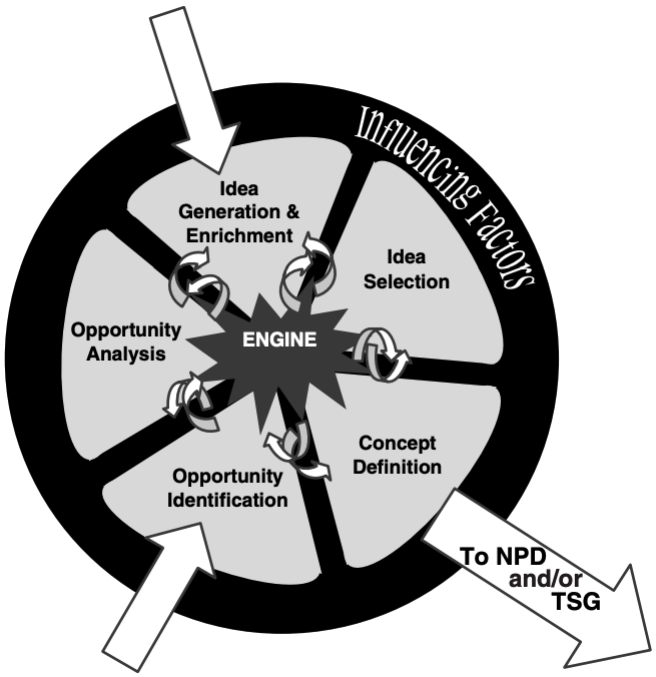


Figure 17 - The New Concept Development (NCD) diagram [64]

It is important to note that the NCD is not linear. The circular shape of the NCD diagram above is meant to suggest that ideas and concepts are expected to iterate across the five elements. The arrows pointing into the model represent starting points and indicate that projects begin at either opportunity identification or idea generation and enrichment. The outbound arrow represents how concepts leave the model and enter the new product development (NPD) or technology stage gate (TSG) process [64], [65].

4.2 Opportunity Identification

This subchapter attempts to identify opportunities that might be worth pursuing. An opportunity can be defined as "… a near-term response to a competitive threat, a breakthrough possibility for capturing competitive advantage, or a means to simplify/speed-up/reduce the cost of operations. The opportunity could be an entirely new direction for the business or a minor upgrade to an existing product. It could also be a new product platform, a new manufacturing process, a new service offering, or a new marketing or sales approach." [64], [65].

One of the most effective methods/tools/techniques to be used is technology and trend analysis [64], [65]. This tool is particularly suited when looking at the scope of the present work.

Taking in the knowledge learned in the chapters "Context" and "Problem" of the present work and using the aforementioned, "Technology trend analysis" technique one can identify that the "sudden" technology shift in ERP system, Database System and development paradigm to SAP S/4 HANA + SAP HANA + "Code Push-Down" can, possibly, provide opportunities to take advantage of.

4.3 Opportunity Analysis

This subchapter attempts to analyze the identified opportunity in order to determine if it might be worth pursuing. For this, additional information and research is needed. The amount of effort put into this step versus the early and/or uncertain decisions depend on the value of the information and the impact it may have on the attractiveness of the opportunity [64], [65].

For this step, many of the tools that can be used for the "Opportunity Identification" step are also used here. That said, this step requires more effort and focus into determining the attractiveness of said opportunity [64], [65].

Taking this into account and the knowledge learned in the chapters "Context" and "Problem" of the present work, we can see that this technological shift seems inevitable, and given the market share of the SAP ERP systems and the surveys related to the application of this technological shift, we can assess that this opportunity may be worth pursuing.

4.4 Idea Generation and Enrichment

This subchapter attempts to generate ideas, and enrich those ideas, to try to best take advantage of the identified opportunity. Idea generation is evolutionary. Ideas can be built up, torn down, combined, reshaped, modified, and upgraded. An idea may go through many iterations and changes as it is examined, studied, and discussed [64], [65].

This process may also be formal, coming up from brainstorming for example or may be informal coming up from outside sources or unusual experiments [64], [65].

For this work the brainstorming approach was taken to generate and enrich ideas to try to best take advantage of the opportunity, in this particular case the sudden technological shift prompted by SAP. These are the resulting ideas:

1. SAP HANA Database Performance Comparison: The focus of this idea would be to independently test SAP's claims on SAP HANA performance and how it compares to other databases applying their "version" of the "Code Push-Down" tools:

- **"Code Push-Down" Development Guidelines**: The focus of this idea would be to provide a set of guidelines to complete what little guidelines exist regarding the application of the "Code Push-Down" development paradigm on standard development contexts;

- **Set of Surveys regarding the "Code Push-Down" Development Paradigm**: The focus of this idea would be to gain a better understanding of the current state of the application of the "Code Push-Down" Development Paradigm, the felt impact and the quality attributes that tend to be prioritized by the required solutions;

- **Guidelines on the Migration of existing solutions to the "Code Push-Down" Development Paradigm**: The focus of this idea would be to provide a set of guidelines to complete what little guidelines exist regarding the application of the "Code Push-Down" development paradigm on already existing solutions;

- **"Code Push-Down" Development Guidelines for Edge Cases**: The focus of this idea would be to provide a set of guidelines to complete what little guidelines exist regarding the application of the "Code Push-Down" development paradigm on edge case solutions that don't follow the traditional development paradigm;

- **A study on the apparent Database Lock-in and possible "middleware" creation**: The focus of this idea would be to determine the possible state of the SAP HANA database lock-in and, if possible/needed, develop a compatibility layer as a sort of "middleware"/Application interface to allow for the use of other databases alongside SAP S/4 HANA.

4.5 Idea Selection

This subchapter attempts to find and choose the idea(s) that may provide the biggest value at the present time for the current state-of-the-art. Selection may be as simple as an individual's choice among many self-generated options or as a formalized and/or complex process [64], [65].

To help with the idea selection step, the analytic hierarchy process by Thomas L. Saaty will be used.

4.5.1 Analytic Hierarchy Process (AHP)

The AHP is a decision-making tool which was developed in 1980. It allows us to make an organized decision via the following process, as detailed in the document "Decision making with the analytic hierarchy process" by Thomas L. Saaty [66]:

1. Define the problem and determine the kind of knowledge sought;

2. Structure the decision hierarchy from the top with the goal of the decision, then the objectives from a broad perspective, through the intermediate levels (criteria on which subsequent elements depend) to the lowest level (which usually is a set of the alternatives);

3. Construct a set of pairwise comparison matrices. Each element in an upper level is used to compare the elements in the level immediately below with respect to it;

4. Use the priorities obtained from the comparisons to weigh the priorities in the level immediately below. Do this for every element. Then, for each element in the level below, add its weighted values and obtain its overall or global priority. Continue this process of weighing and adding until the final priorities of the alternatives in the bottom most level are obtained.

To make comparisons, we need a scale of numbers that indicates how many times more important or dominant one element is over another element with respect to the criterion or property with respect to which they are compared. The following table exhibits the scale:

Table 1 - AHP Activity Comparison Importance Scale

Intensity of Importance*	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective.
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favour one activity over another.
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favour one activity over another.
6	Strong plus	
7	Very strong	An activity is favoured very strongly over another; its dominance is demonstrated in practice.
8	Very strong plus	
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation.

* If an activity "X" has one of the above non-zero numbers assigned to it when compared with an activity "Y", then "Y" has the reciprocal value when compared with "X".

In cases where the difference between each importance may be too great, one can also use decimals (e.g., X.1 – X.9)

4.5.2 Applying AHP

Following the steps defined above, we can start by stating that the goal is to select the most valuable, feasible idea from the generated ones.

Then we must define criteria by which to evaluate each idea. These criteria were chosen considering the ideas themselves, the scope of this work and the knowledge of the author. The following are the chosen criteria:

- **Time Restrictions**: If the idea presents time restrictions as this work has a pre-defined due date and is limited by it;
- **Idea Precedence**: If the knowledge gained by implementing the idea could benefit the other;
- **Impact**: The impact this idea may have in the current state-of-the-art;
- **Time Sensitivity**: The importance of the idea being implemented sooner than later.

This results in the following diagram:

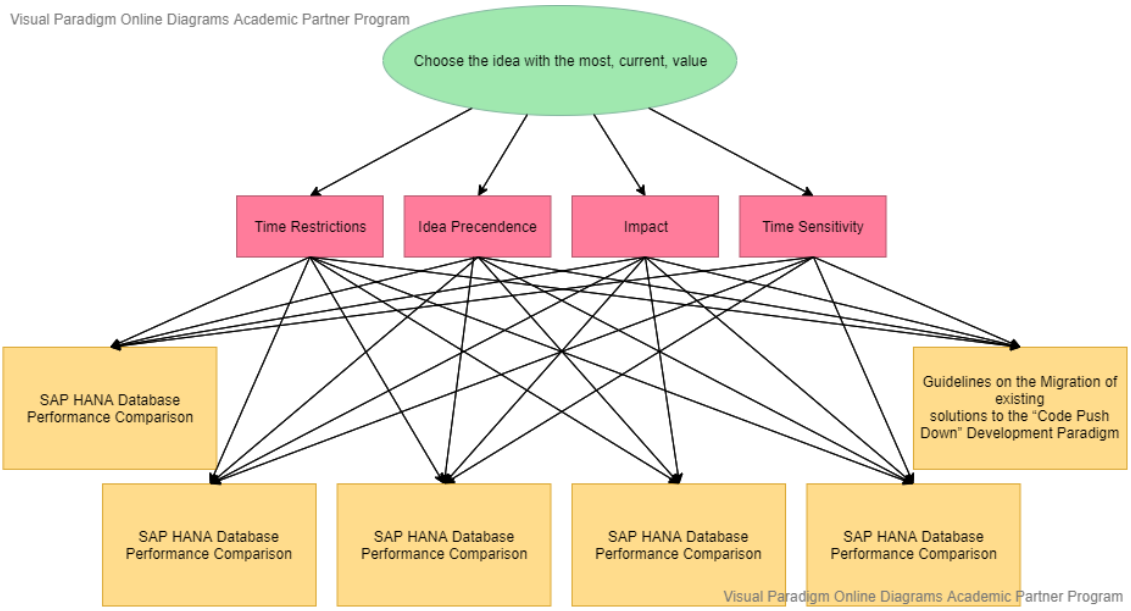


Figure 18 - AHP Hierarchical Model Tree

Now, in order to evaluate the ideas, let's create an evaluation table based on the "Table 1 - AHP Activity Comparison Importance Scale"

Table 2 - AHP Evaluation Tree

Criterion	Time Restriction	Idea Precedence	Impact	Time Sensitivity
Time Restriction	1	6	3	5
Idea Precedence	1/6	1	1/5	1/3
Impact	1/3	5	1	2
Time Sensitivity	1/5	3	1/2	1
Sum	1,70	15	4,70	8,33

After defining the importance of each criteria, using the pairwise comparison of the table above, it must be normalized in order to retrieve the priorities of each measure. This is done by calculating the mean value of each row.

The following table does just that:

Table 3 - Normalized AHP Evaluation Tree

Criterion	Time Restriction	Idea Precedence	Impact	Time Sensitivity	Mean
Time Restriction	0,588	0,400	0,638	0,600	0,557
Idea Precedence	0,098	0,067	0,043	0,040	0,062
Impact	0,196	0,333	0,213	0,240	0,246
Time Sensitivity	0,118	0,200	0,106	0,120	0,136

With the normalized AHP evaluation tree complete, we now can define the overall priority each criterion has.

Table 4 - AHP Criteria Priorities

Priority	Criterion	Significance
1	Time Restriction	55,7%
2	Impact	24,6%
3	Time Sensitivity	13,6%
4	Idea Precedence	6,2%

Based on these priorities, shown in the table above, let's classify each of the ideas in order to obtain the most valuable one, currently.

- **SAP HANA Database Performance Comparison**:
    - **Time Restriction**: This idea seems to be doable in the available time;
    - **Impact**: This idea's impact would most likely be low as the use of the HANA DB is required;
    - **Time Sensitivity:** This idea isn't time sensitive as its impact would be the same now or in the near future;
    - **Idea Precedence:** This idea would not bring benefits to be implemented before the others.

- **“Code Push-Down” Development Guidelines**:
    - **Time Restriction**: This idea seems to be doable in the available time;
    - **Impact**: This idea’s impact would likely be significant as the technological shift leverages its use;
    - **Time Sensitivity:** This idea is time sensitive as the technological shift is starting to occur now;
    - **Idea Precedence:** This idea would be useful if it were implemented before the others.

- **Set of Surveys regarding the “Code Push-Down” Development Paradigm**:
    - **Time Restriction**: This idea seems like it wouldn’t be doable in the available time;
    - **Impact**: This idea’s impact would likely be moderate as its use would be more informative than anything to act on;
    - **Time Sensitivity:** This idea is somewhat time sensitive as it might have an impact on the application of the “Code Push-Down” development Paradigm;
    - **Idea Precedence:** This idea would be useful if it were implemented before the others.

- **Guidelines on the Migration of existing solutions to the “Code Push-Down” Development Paradigm**:
    - **Time Restriction**: This idea seems to be doable in the available time;
    - **Impact**: This idea’s impact would likely be significant as the technological shift leverages its use;
    - **Time Sensitivity:** This idea is time sensitive as the technological shift is starting to occur now;
    - **Idea Precedence:** This idea would be useful if it were implemented before the others.

- **“Code Push-Down” Development Guidelines for Edge Cases**:
    - **Time Restriction**: This idea seems to be doable in the available time;
    - **Impact**: This idea’s impact would likely be moderately significant as these are niche use cases;
    - **Time Sensitivity:** This idea is time sensitive as the technological shift is starting to occur now;
    - **Idea Precedence:** This idea would be useful if it were implemented before the others.

- **A study on the apparent Database Lock-in and possible “middleware” creation**:
    - **Time Restriction**: This idea seems like it wouldn’t be doable in the available time;
    - **Impact**: This idea’s impact would likely be very significant as the technological shift may significantly shift because of this;
    - **Time Sensitivity:** This idea is time sensitive as the technological shift is starting to occur now;
    - **Idea Precedence:** This idea would not bring benefits to be implemented before the others.

By following the analysis done and taking into account the criteria priorities we can conclude that the idea that would bring the most value to be done now is “**“Code Push-Down” Development Guidelines**”.

4.6 Concept Definition

This final subchapter serves to present the purpose and value of the current project alongside the implementation of the current idea.

The current technological shift occurring within the SAP Ecosystem serves as an opportunity that the present work attempted to present several ideas for. From those the “**“Code Push-Down” Development Guidelines**” idea was selected for development in the current work.

4.6.1 Value Proposition

If done successfully, the present work should allow for a more unbiased point of view on the technological shift that is happening within the SAP Ecosystem. This work also strives to validate SAP's claims on the "Code Push-Down" development paradigm while also providing a set of simple to follow guidelines supported by an experiment, in order to complement the existing guidelines/documentations about the subject.

5 Simulated Problem Statement

In order to apply and compare both development paradigms, a simulated problem statement will be used.

This problem statement has been provided by the ROFF Consulting [67] (now called Inetum) ABAP Academy. The problem statement is used in the ABAP academy to simulate a realistic development in the SAP context, in terms of scope, size and complexity. As such, this statement should be an adequate candidate on which to build our solutions.

The problem statement has been provided in Portuguese. As such, it has been translated and shortened by removing some sections that, although relevant in a business context, are not as important for the present work.

The original document is appended to the present work as "Appendix A".

5.1 Problem Statement – "S00124_2017 - Management of External Suppliers"

Table 5 - Problem Statement General Information

System:	**SAP ERP**
Reference:	DF_SAP_ MM _S00124_2017
Date:	27.07.2017
Version:	4.0
Author:	José Bento
Description:	Functional design of the management module for external suppliers

5.1.1 Scope of the Document

This document's purpose is to specify the technical and functional requirements necessary for the development of a management module for ROFF's external suppliers.

Currently, ROFF is supplied, both in terms of goods and services, by several suppliers, having been identified that these also use their own suppliers. Since the organization's control and management unit (UCG) will start to audit the cost structure as of the next fiscal year, it is necessary to implement a module that allows storing and consulting monthly data on the volume of goods and services supplied indirectly by external suppliers with the intention of perhaps contracting them directly with the aim of reducing costs.

The following diagram should help understand the current situation.

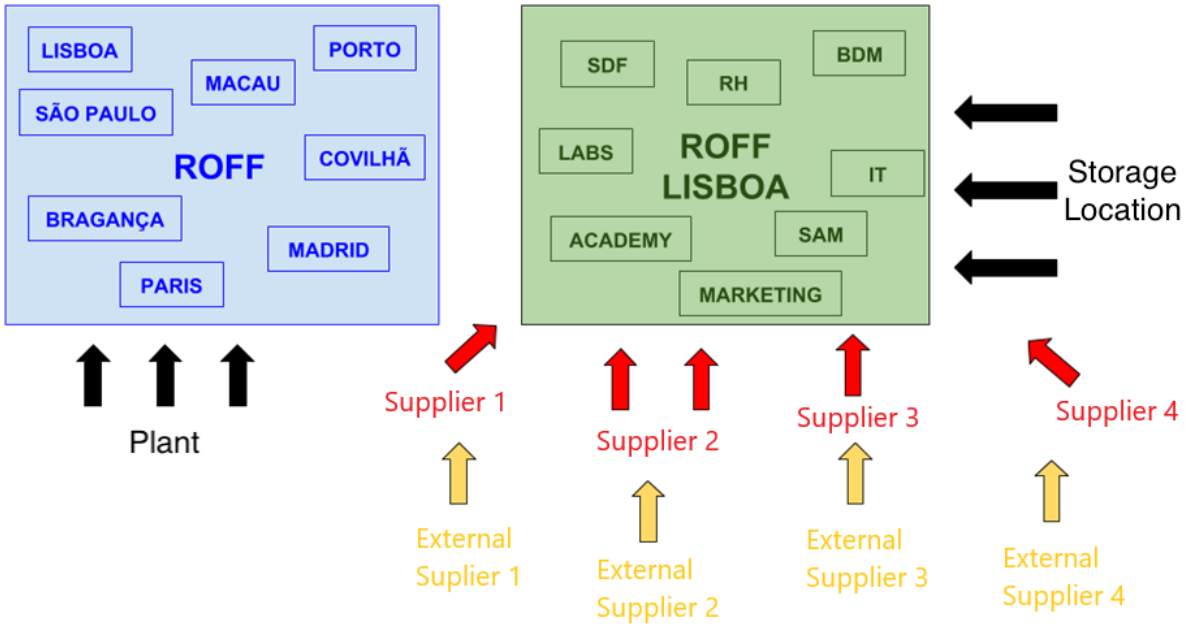


Figure 19 - Problem Overview

5.1.2 Technical and Functional Requirements

To manage data from external suppliers, it will be necessary to define two tables in the database, one to store individual data from external suppliers and the other to store data from monthly accounting movements. Both tables will be non-expandable.

The table where the individual data will be saved must have a maintenance screen and an access transaction to allow maintenance by the UCG. This table will be named **ZTMM _FEX _ <Initials>** and will have the following structure:

Table 6 - External Supplier Table

Field name	Data Element	Description
MANDT*	MANDT	Client
FEXNR*	ZMM _ FEXNR_<Initials>	External Supplier. It will be a CHAR field of size 10. It will have a conversion routine associated in order to allow the control of leading zeros.
NAME	BAPITDOBNAME	External Supplier Name
FEXTY	ZMM _ FEXTY_<Initials>	External Supplier Type. It will be a CHAR field of size 1 and should take only 1 value from the following list: 1 – Active Supplier 2 - Old Supplier 3 – Eventual Supplier 4 – Auxiliary Supplier 5 – Other
STCEG	STCEG	VAT Registration Number
WAERS	WAERS	Currency Key
STRAS	STRAS_GP	Street and House Number
ORT01	ORT01_GP	City
PSTLZ	PSTLZ	Postal Code
LAND1	LAND1_GP	Country Key

* Key fields

The transaction to access the maintenance screen of this table will be named **ZMM _FEX _<Initials>**.

The table where the data of monthly accounting movements will be saved will have the name **ZTMM _MOV FEX_<Initials>** and will have the following structure:

Table 7 - External Accounting Movements Table

Field name	Data Element	Description
MANDT*	MANDT	Client
WERKS*	WERKS_D	Plant
LGORT*	LGORT_D	Storage Location
FEXNR*	ZMM _ FEXNR_<Initials>	External Supplier. It will be a CHAR field of size 10. It will have associated a conversion routine to allow the control of leading zeros.
GJAHR*	GJAHR	Fiscal Year
MONAT*	MONAT	Fiscal Period
WRBTR	WRBTR	Amount in Document Currency
GSWRT	GSWRT	Total Value of Item
ANZLI	MC_ANZLI	No. of deliveries

* Key fields

The search help **ZSH MM _ FEXNR_<Initials>** must be defined to allow the search for external suppliers. This search help helps us use the table **ZTMM _FEX_ <Initial>** as a selection method and it will be available on all screens that have the External Supplier as a selection field.

A reporting program will be developed with a view to consult monthly accounting movement data. This program will have the name **ZRMM _MOVMES FEX_<Initials>** and it will have the following elements in its selection screen:

- **Plant**: Simple selection parameter, mandatory;
- **Storage Location**: Simple selection parameter, mandatory;
- **External Supplier**: Multiple selection parameter;
- **Current Period**: Radio button parameter, selected by default;
- **Current Year**: Radio button parameter;
- **Free Selection**: Radio button parameter;
- **Fiscal Year**: Multiple selection parameter;
- **Period**: Multiple selection parameter.

To execute the program, the user must fill in the “Plant” and “Storage Location” fields and indicate, through the radio button parameters, which period of time he wishes to consult. The fields “Fiscal Year” and “Period” must be hidden except when the user selects the option “Free Selection”, in which case they will be available.

At the end of the program execution, an output listing will be presented, in ALV Grid format, which will have the following structure:

Table 8 - List Format Table

Field name	Data Element	Description
GJAHR	GJAHR	Fiscal Year
MONAT	MONAT	Fiscal Period
FEXNR	Z MM_ FEXNR_<Initials>	External Supplier
FEXTY	Z MM_ FEXTY_<Initials>	External Supplier Type
NAME	BAPITDOBNAME	External Supplier Name
WRBTR	WRBTR	Amount in Document Currency
GSWRT	GSWRT	Total Value of Item
PERVT	PRZOPKUM	Percentage of Total Value of Item (WRBTR / GSWRT)
ANZLI	MC_ANZLI	No. of deliveries
VALMF	BINV_VALUE	Average Value per Supply (WRBTR / ANZLI)
WAERS	WAERS	Currency Key

All entries whose Total Value of Item percentage has a value between 50% and 80% should be colored yellow; all entries whose percentage of the Total Value of Item is more than 80% should be painted red.

The ALV output listing must be ordered by Year, Period and External Supplier, and these fields must also be marked as key fields.

It should be displayed at the end of the list, the sum of all the fields Amount in Document Currency, Total Value of Item, and number of supplies.

At the top of the output listing, an ALV header should be presented containing three pieces of information:

- The Plant indicated on the selection screen and its description;
- The Storage Location indicated on the selection screen and its description;
- Search time period indicated on the selection screen:
    - Month and Year, if the Current Period has been selected;
    - Year, if only the Current Year has been selected;
    - Free selection, if any free selection has been selected.

The transaction for accessing the **ZRMM_MOVMESFEX_<Initials>** program will be **ZMM_MOVMESFEX_<Initials>**.

In the button bar of the output listing in ALV Grid format, a new button **"Export to PDF"** should be added, which will have associated the functionality to export the output listing to a Smartform form and save it in a local file in PDF format.

The user should be presented with a popup window to indicate the city and name of the file to be saved on his computer. The name of the file must be pre-filled which will be **<PLANT>_<STORAGE LOCATION>_ <CURRENT_DATE>_<CURRENT_TIME>.PDF**

The form must have the following structure:

- The ROFF company logo in the upper left corner;
- The Plant and the Storage Location and their descriptions at the top;
- Time period of research in the upper right corner:
    - Month and Year, if the Current Period has been selected;
    - Year, if the Current Year was selected;
    - Free Selection, if any free selection has been selected.

- A table with the data from the ALV Grid output listing with the respective headers listing the fields Year, Period, External Supplier, External Supplier Type, Amount in Document Currency, Total Value of Item, Number of Deliveries and Currency and a footer with the total of the Amount in Document Currency, Total Value of Item and Number of Supplies;

- The indication of the page/total of pages, of the user who generated the file and the date of generation of the file in the lower right corner.

A new button **"Export to XML"** should be added to the button bar of the output listing in ALV Grid format, which will have associated the functionality of **exporting** the output listing to a local file in XML format with the following structure:

```
<MOVMESFEX>
          <HEAD>
               <WERKS>…</WERKS> * the Plant and its description
               <LGORT>...</LGORT> *Storage Location and their descriptions
will
               <PERIO>…</PERIO> * o Time period of research
               <WRBTR>…</WRBTR> * the total Amount in Document Currency
               <GSWRT>…</GSWRT> * the total of the Total Value of Item
               <ANZLI>…</ANZLI> * the total number of deliveries
          </HEAD>
          <ITEM> *content of each line in the listing
               <GJAHR >… </GJAHR >
               <MONAT >… </MONAT >
               <FEXNR >… </FEXNR >
               <FEXTY >… </FEXTY >
               <NAMEF >… </NAMEF >
               <WRBTR >… </WRBTR >
               <GSWRT >… </GSWRT >
               <PERVT >… </PERVT >
               <ANZLI >… </ANZLI >
               <VALMF >… </VALMF >
               <WAERS >… </WAERS >
          </ITEM>
              …
</MOVMESFEX>
```

The user should be presented with a popup window to indicate the city and name of the file to be saved on his computer. The name of the file must be pre-filled, which will be **MOV MES FEX _<CURRENT_DATE>_<CURRENT_TIME> .XML** .

In order to centralize access to current (and possible future) transactions developed within the scope of this module for managing external suppliers, a cockpit should be developed that allows access to them using indicative and illustrative buttons. These buttons should be on the cockpit input screen. Additionally, a third button must be defined that will allow access to a second screen where the user can manually insert records in the **ZTMM_MOVFEX_ <Initials>** table.

As such, this screen must have input fields that allow filling in the information present in the table and specified above. Data recording functionalities, cleaning of field content and others that may be useful should be implemented.

This cockpit will be implemented via the Module Pool **ZMM_GESTAO_FEX_<Initials>** and will be accessed via transaction **ZMM _ G FEX_<Initials>**.

5.1.3 Identification of Interfaces

Apart from the **ZMM_FEX_<Initials>**, **ZMM_MOVMESFEX_<Initials>** and **ZMM_GFEX_<Initials>** transactions, it will not be necessary to develop any other interface or accesses.

5.1.4 Identification of Access Profiles

Transactions to be defined within the scope of this request will be open access, without any restrictions on the access profiles of UCG users.

5.1.5 Data Migration Needs

The process of feeding the table where the accounting movement data will be saved will be defined and programmed by the UCG through daily uploads.

6 Analysis

This chapter will attempt to analyze and correctly interpret the problem statement at hand. This will be done by modelling the business and defining the requirements in order to help define the scope and design the solutions.

6.1 Domain Model

The following domain model allows a visual representation of the business and the way it relates to its entities.

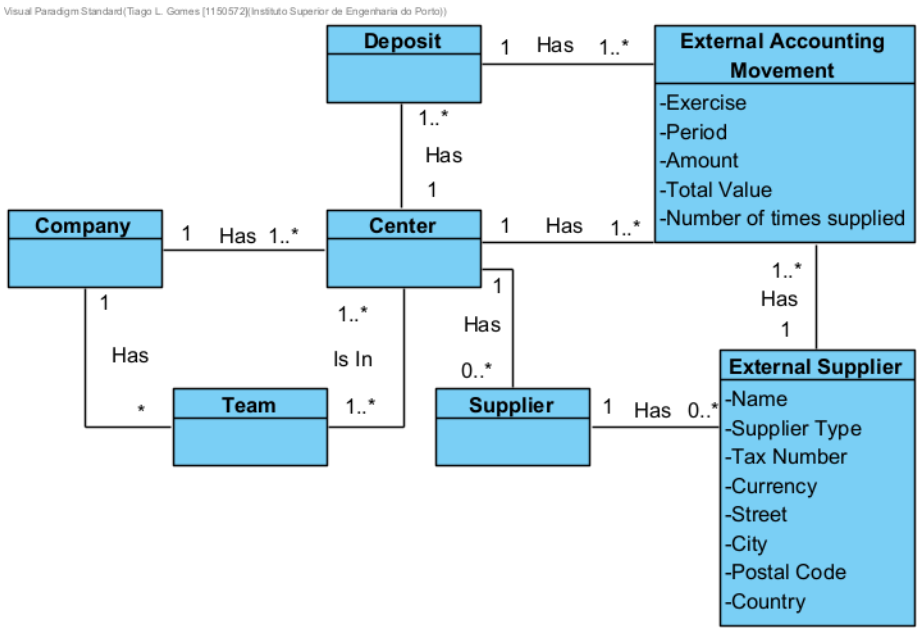


Figure 20 - Domain Model

One thing to note is that this problem statement is focused on the interaction of the organization with suppliers and external suppliers. As such, the domain model focuses on modeling these entities:

- **Supplier**: The entity that is directly contracted by the organization to provide goods and/or services.
- **External Supplier**: The entity that is contracted by a supplier, in order to provide goods and/or services to the organization.
- **External Accounting Movement**: This entity represents an instance (or multiple instances for the same key fields) where an external supplier provided goods and/or services to the organization.

6.2 Requirement Engineering

Although the problem statement itself is already detailed and well sectioned to start with, there is some further work that can be done to better define the scope of the problem at hand.

To achieve this, this subchapter will try to clearly interpret and define the requirements for the solutions. A requirement can be defined as "…a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document."[68]

In order to define these requirements, the FURPS+ system for requirement classification will be used.

6.2.1 FURPS+ System for Requirement Classification

This system was devised by Robert Grady with the intention of better classifying and understanding functional and non-functional requirements.

For this purpose, the acronym FURPS+ was devised representing the following classifications [68]:

- **Functionality**: This classification represents the main product features of the solution. These features may be domain specific or not;
- **Usability**: This classification represents concerns such as documentation, UI/UX consistency, and accessibility;
- **Reliability**: This classification represents concerns such as availability, accuracy, and failure frequency;
- **Performance**: This classification represents concerns such as response time, system startup time, recovery time, and processing capacity;
- **Supportability**: This classification represents concerns such as testability, maintainability, adaptability, and scalability.

The "+" in this acronym was added to also take into consideration concerns/constraints such as:

- **Design Requirements**: This classification represents architectural/design constraints;
- **Implementation Requirements**: This classification represents constraints regarding coding or the construction of the solution;
- **Interface Requirements**: This classification represents constraints regarding the interaction with other systems;
- **Physical Requirements**: This classification represents constraints regarding hardware.

6.2.2 Requirements

This subchapter will attempt to identify and classify the problem statement's requirements according to the FURPS+ classification system.

It is to note that some specific information about the requirements is omitted in order to better summarize the requirements (e.g., The specific color or name of something).

6.2.2.1 Functionality

This chapter contains the use case diagram below to give a visual representation of the identified functional requirements.

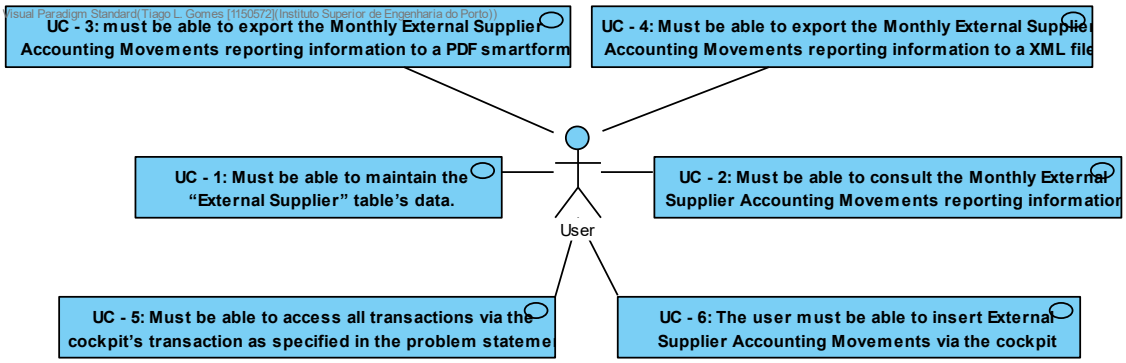


Figure 21 - Use case Diagram

UC - 1: The user must be able to maintain the "External Supplier" table's data.

UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information.

UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform.

UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file.

UC - 5: The user must be able to access all transactions via the cockpit's transaction as specified in the problem statement.

UC - 6: The user must be able to insert External Supplier Accounting Movements via the cockpit.

6.2.2.2 Usability

UR - 1: All screens that contain the “External Supplier” field must have a search help as specified in the problem statement.

UR - 2: The reporting program’s selection screen must have specific input fields as specified in the problem statement.

UR - 3: The reporting program’s results must be color coded as specified in the problem statement.

UR - 4: The reporting program’s results must be ordered as specified in the problem statement.

UR - 5: The reporting program must contain sums for specific fields as specified in the problem statement.

UR - 6: The reporting program’s output must have a header as specified in the problem statement.

UR - 7: The reporting program’s output to a PDF Smartform must have a button as specified in the problem statement.

UR - 8: The reporting program’s output to a PDF Smartform must have a directory selection popup window.

UR - 9: The reporting program’s output to a PDF Smartform must be structured as specified in the problem statement.

UR - 10: The reporting program’s output to an XML file must have a button as specified in the problem statement.

UR - 11: The reporting program’s output to an XML file must have a directory selection popup window.

UR - 12: The reporting program’s output to an XML file must be structured as specified in the problem statement.

UR - 13: The cockpit must be structured as specified in the problem statement.

6.2.2.3 Supportability

SR - 1: The cockpit will allow for future extensions as the addition of further navigation to other transactions.

6.2.2.4 Implementation Requirements

IPR - 1: The database must have a non-extensible “External Supplier” table structured as specified in the problem statement.

IPR - 2: The database must have a non-extensible “Monthly External Supplier Accounting Movements” table structured as specified in the problem statement.

IPR - 3: The “External Supplier” table must have a maintenance screen.

IPR - 4: The “External Supplier” table’s maintenance screen must be accessible via a transaction.

IPR - 5: The Monthly External Supplier Accounting Movements information must be implemented in a reporting program.

IPR - 6: The reporting program’s main output must be an ALV Grid as specified in the problem statement.

IPR - 7: The reporting program must be accessible via a transaction.

IPR - 8: The Monthly External Supplier Accounting Movements should be visible via a reporting program.

IPR - 9: The cockpit must be implemented using module pools.

7 Design

This chapter tries to provide suitable designs, taking into account the requirements and analysis done, for both development paradigms in order to achieve what would be considered "Realistic" solutions.

One thing to take into consideration is that, although there are multiple other sensible Architectural Styles/Patterns, Principles, Guidelines, etc., that could also be used in these solutions, we have to consider the scope of this work and the focus on what is the "Standard" SAP general development paradigm and the "Code Push-Down" development paradigm.

7.1 "Standard" development paradigm

This subchapter attempts to provide a suitable design for a solution that follows the "Standard" development paradigm.

7.1.1 General Solution Architecture

A general solution architecture is a set of coherent and compatible design patterns chosen and applied, considering all the project requirements, restrictions, and recommendations, in an effort to provide a solution suitable to the current needs.

Considering the fact that the provided problem statement tries to mimic what a "Standard" development problem is, the general solution architecture, and that the focus of this solution is the "SAP "Standard" Development ", the following component diagram will be the resulting general solution architecture.

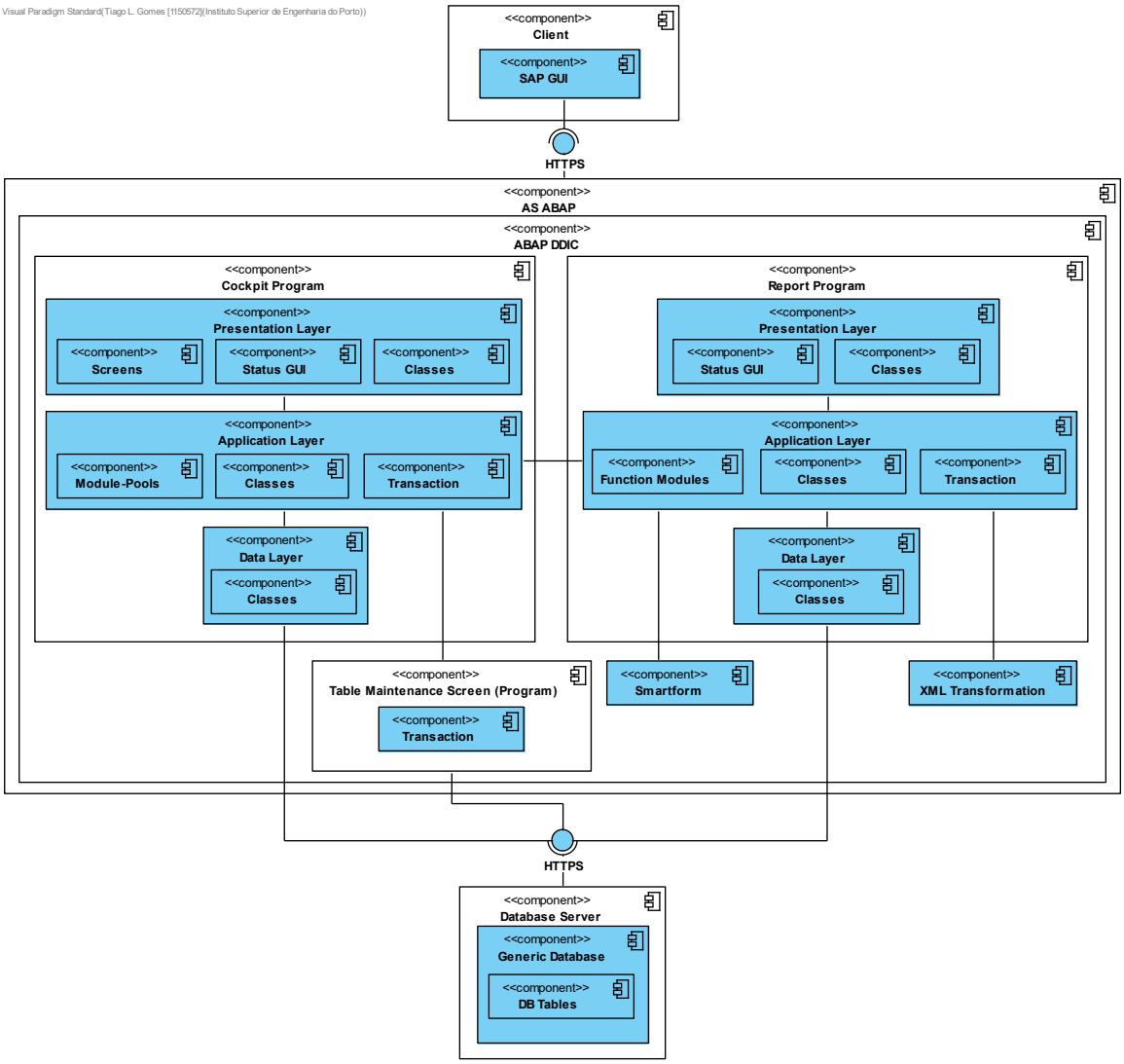


Figure 22 - "Standard" Component Diagram

One important thing to note is that, when created, the Table Maintenance “Screen” program is automatically generated and maintained by the SAP ERP system and is independent from the remaining solution. As such, in the component diagram above and remaining diagrams, it will be represented as a “black box”, except the Transaction component as this is the only component that is relevant to represent for this work.

7.1.2 Use Case Specification

This subchapter intends to provide a design to the main problem statement's identified functional requirements.

7.1.2.1 UC - 1: The user must be able to maintain the "External Supplier" table's data

For this use case, the user wants to be able to access what is designated as a "Maintenance Screen" program via a transaction in order to maintain the correspondent database table's data.

The following sequence diagram shows the design of this use case.

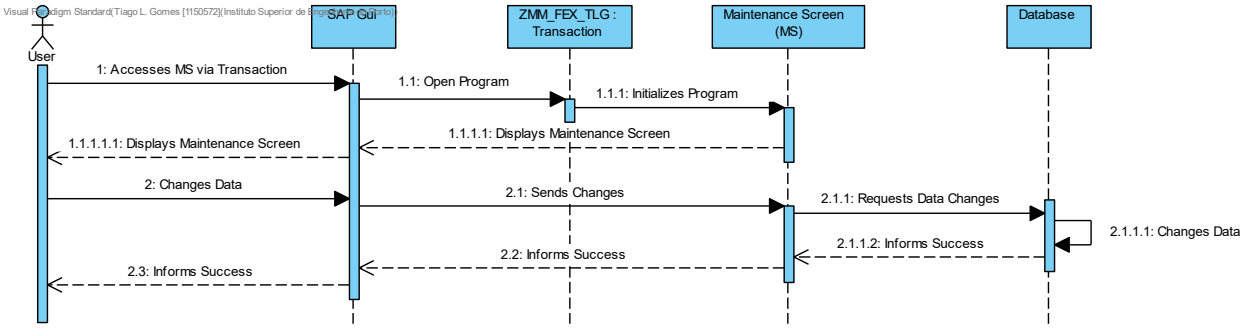


Figure 23 - UC1 Sequence Diagram

7.1.2.2 UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information

For this use case, the user wants to be able to consult the Monthly External Supplier Accounting Movements according to the chosen filtering criteria.

The following sequence diagram shows the design of this use case.

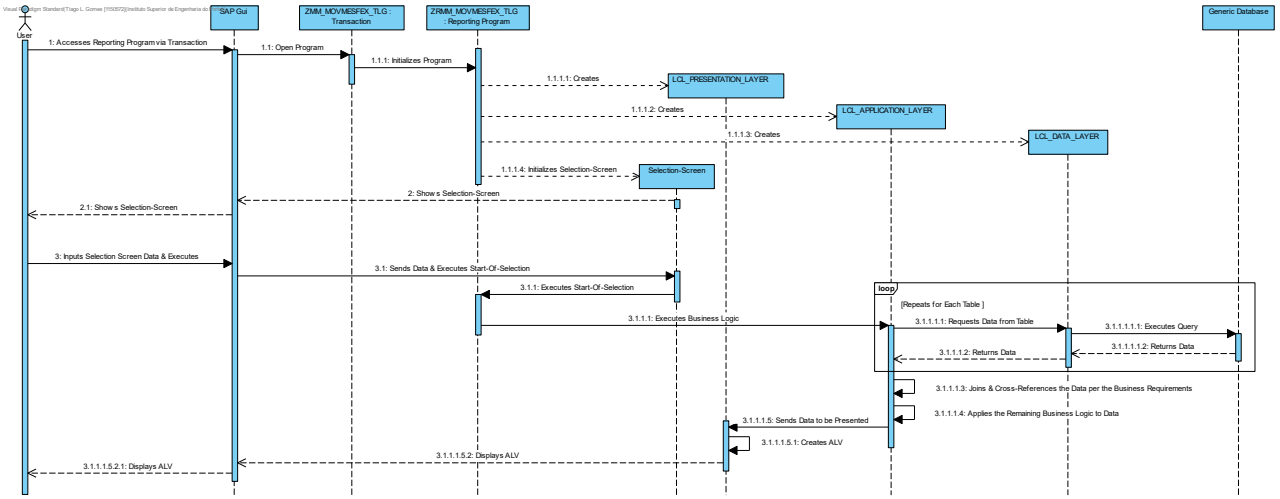


Figure 24 - UC2 Sequence Diagram

7.1.2.3 UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform

For this use case, the user wants to be able to export to a PDF file the Monthly External Supplier Accounting Movements currently displayed in the ALV.

To simplify the following sequence diagram, it will start where the sequence diagram for "UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information" ends, as this functional requirement is related to the results provided by the previous use case.

The following sequence diagram shows the design of this use case.

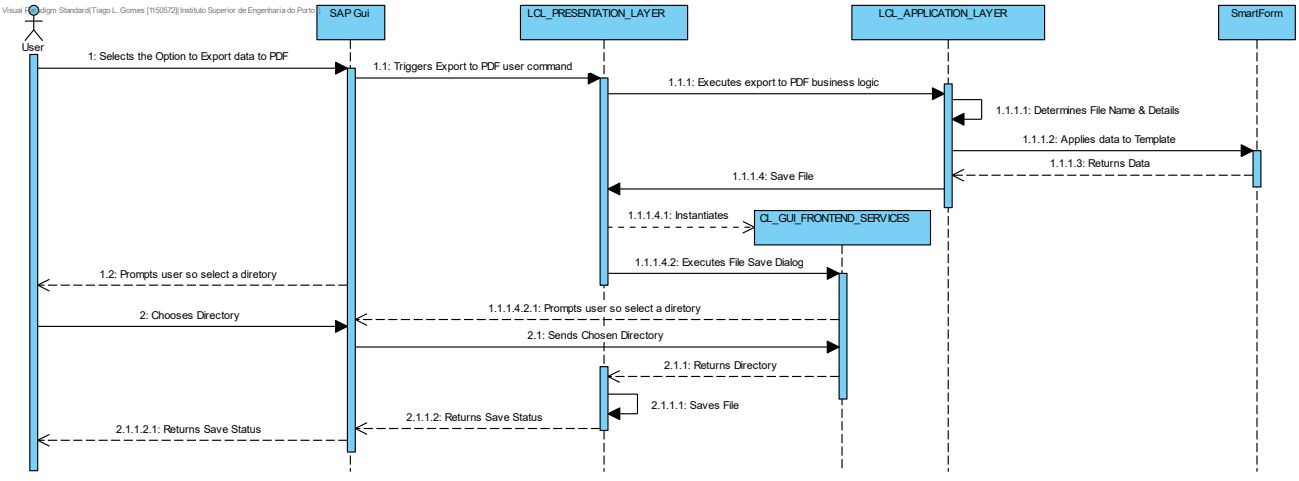


Figure 25 - UC3 Sequence Diagram

7.1.2.4 UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file

For this use case, the user wants to be able to export to an XML file the Monthly External Supplier Accounting Movements currently displayed in the ALV.

To simplify the following sequence diagram, it will start where the sequence diagram for "UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information" ends, as this functional requirement is related to the results provided by the previous use case.

The following sequence diagram shows the design of this use case.

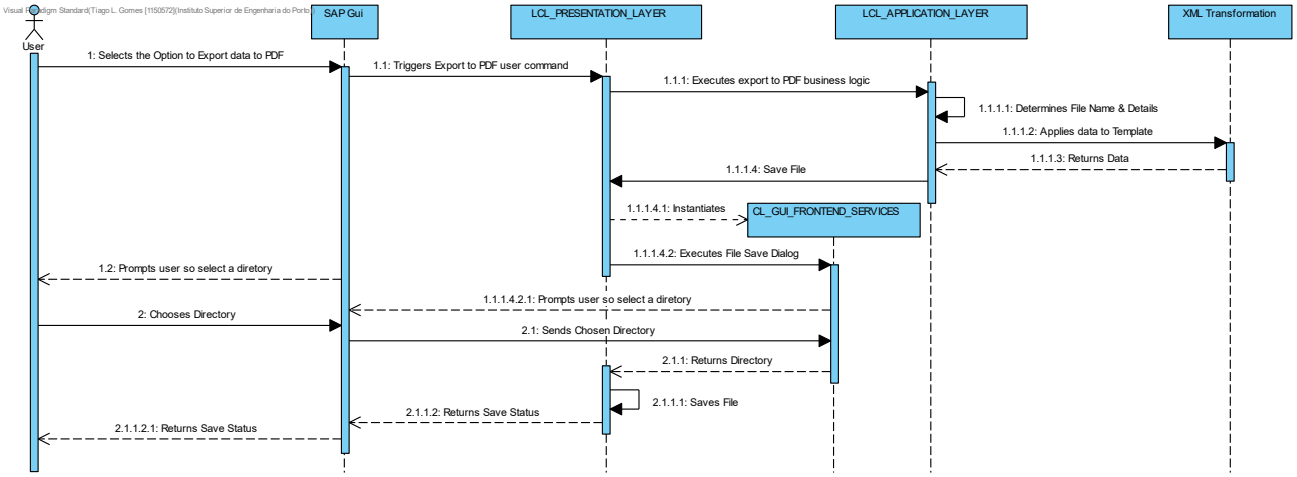


Figure 26 - UC4 Sequence Diagram

7.1.2.5 UC - 5: The user must be able to access all transactions via the cockpit's transaction as specified in the problem statement

For this use case, the user wants to access a Cockpit that aggregates all the solution components for easy navigation. This sequence diagram is somewhat simplified as some of the details of the inner workings of the program are not pertinent to this use case

The following sequence diagram shows the design of this use case.

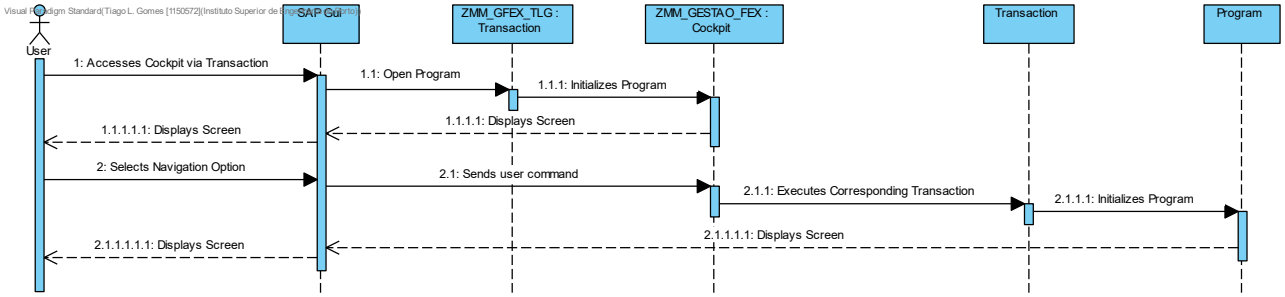


Figure 27 - UC5 Sequence Diagram

7.1.2.6 UC - 6: The user must be able to insert External Supplier Accounting Movements via the cockpit

For this use case, the user wants to be able to insert new external supplier accounting movements via the cockpit program.

The following sequence diagram shows the design of this use case.

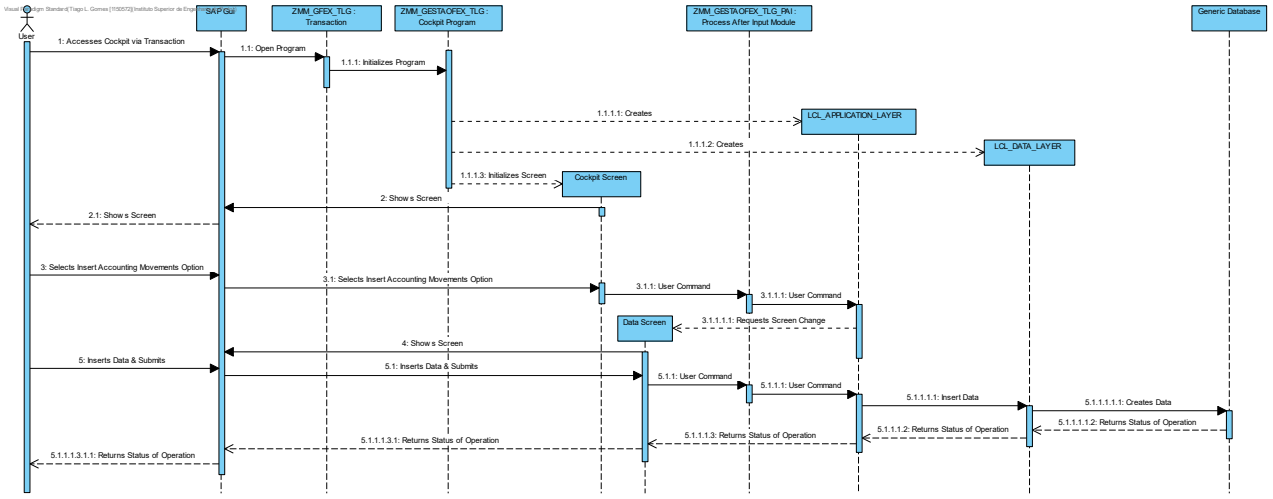


Figure 28 - UC6 Sequence Diagram

7.1.3 Data Model

This subchapter details the data design for the solution. The data model design was complete enough in the provided problem statement. As such, the diagram below will try to represent the defined data model as it was provided.

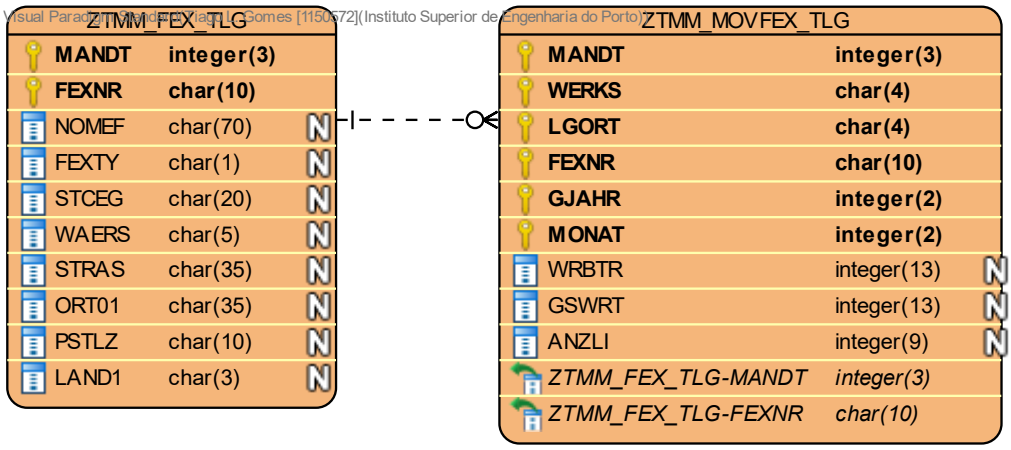


Figure 29 - "Standard" Data Model

7.2 “Code Push-Down” development paradigm

This subchapter attempts to provide a suitable design for a solution that follows the “Code Push-Down” development paradigm.

7.2.1 General Solution Architecture

As stated previously, a general solution architecture is a set of coherent and compatible design patterns chosen and applied, considering all the project requirements, restrictions, and recommendations, in an effort to provide a solution suitable to the current needs.

Considering that the “Code Push-Down” Development Paradigm focuses on pushing down the application/business logic code, to the data layer or even database. This general application architecture will reflect the changes done to the “Standard” architecture in order to focus this subchapter.

Currently, as shown in Figure 22 - "Standard" Component Diagram, there are three components in the solution who access the database. As such, these would be candidates for the use of the “Code Push-Down” paradigm, but since the Maintenance Screen (Program) is generated and maintained by the SAP ERP, this component will be excluded from the candidate components. As such, the “Code Push-Down” paradigm will be applied to the Reporting Program and the Cockpit program.

The following component diagram represents the created general solution architecture.

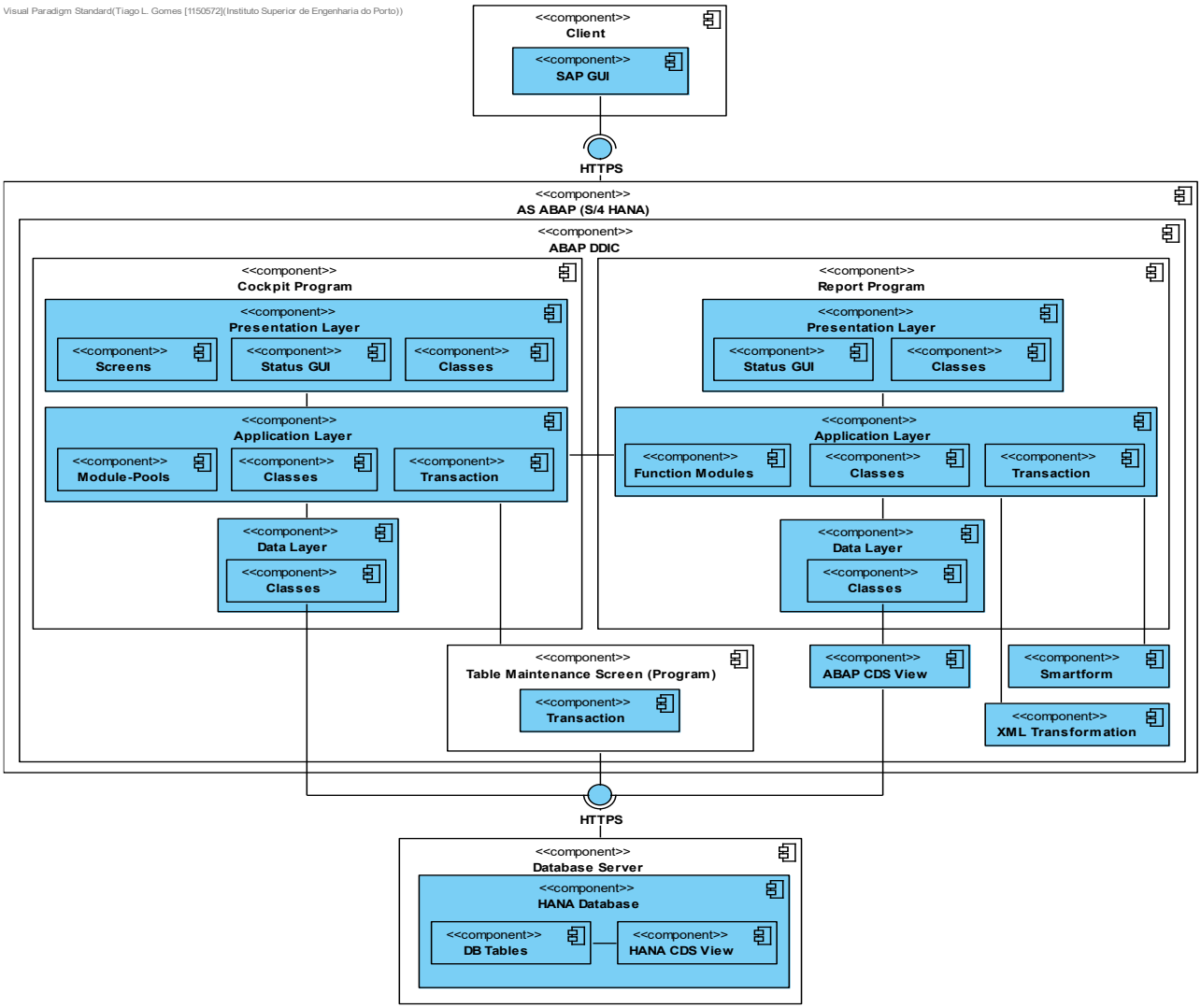


Figure 30 - "Code Push-Down" Component Diagram

As shown in the diagram above, in terms of the changes to the general solution architecture itself, there was the introduction of an ABAP CDS View (to be used alongside ABAP + Open SQL) and, consequently, a database view (a HANA CDS View in this particular case) that is deployed, as per the use of the recommended “Top-Down” approach in the “Code Push-Down” Development Paradigm.

The inclusion of an ABAP CDS View alongside ABAP + Open SQL was to promote reusability by facilitating access to data in this format, as its logic seems to be a good candidate for use by other solutions in the future. That said, the use of an ABAP CDS AMDP would also be possible and would, perhaps, simplify the distribution of the business logic for the data. But its use would force the use of an SAP HANA database.

It is important to mention that the current solution and the “Code Push-Down” tools it uses do not necessarily force the use of SAP HANA.

7.2.2 Use Case Specification

As stated previously, this subchapter intends to provide a design to the main problem statement’s identified functional requirements.

As mentioned before, this subchapter is taking into account that the “Code Push-Down” Development Paradigm focuses on pushing down the application/business logic code to the data layer or even database. As such, this subchapter will focus on what changed with the application of this paradigm.

7.2.2.1 UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information

For this use case, the user wants to be able to consult the Monthly External Supplier Accounting Movements according to the chosen filtering criteria.

This use case is the one that ends up having the most significant changes. Although the same effect could be achieved with “just” a more complex Open SQL query, there was value to be had with the introduction of the ABAP CDS view. This value is due to the nature of the information provided by this view and the fact that this business logic seems like it can be useful to other future solutions.

The following sequence diagram shows the design of this use case.

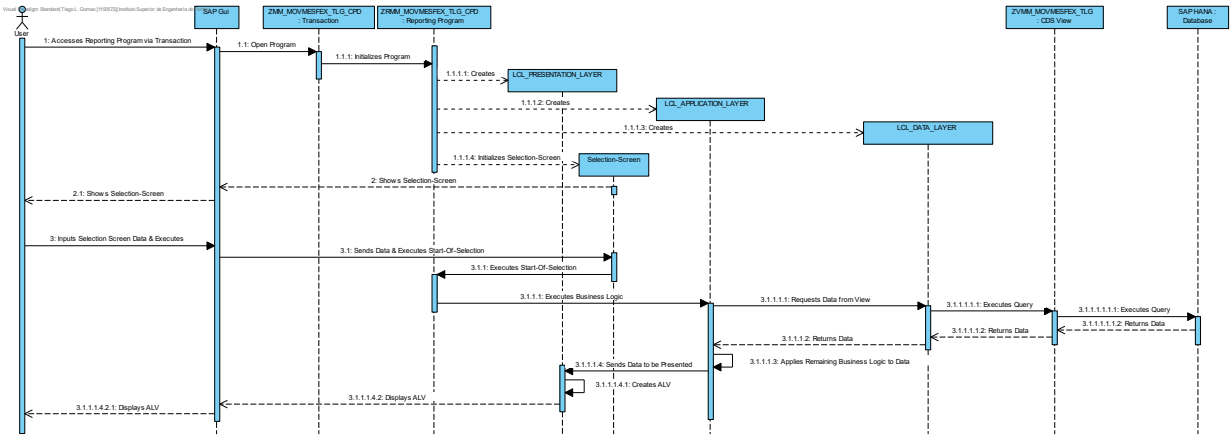


Figure 31 - UC2 "Code Push-Down" Sequence Diagram

7.2.2.2 UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform

For this use case, the user wants to be able to export to a PDF file the Monthly External Supplier Accounting Movements currently displayed in the ALV.

Although this use case is affected by the changes as this starts by having the same business flow as the UC2, the proprietary business logic for this use case ends up being the same as the one depicted in UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform

7.2.2.3 UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file

For this use case, the user wants to be able to export to an XML file the Monthly External Supplier Accounting Movements currently displayed in the ALV.

Although this use case is affected by the changes as this starts by having the same business flow as the UC2, the proprietary business logic for this use case ends up being the same as the one depicted in UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file.

7.2.2.4 UC - 6: The user must be able to insert External Supplier Accounting Movements via the cockpit

For this use case, the user wants to be able to insert new external supplier accounting movements via the cockpit program.

Although this use case accesses the database, the way it is implemented stays the same as before. The way it accesses the database via ABAP + Open SQL (which in itself is a form of "Code Push-Down") is quite simple and basic in terms of business logic and was already pushing down the application logic that could be pushed down.

7.2.3 Data Model

This subchapter details the data design for the solution. This data model design was done by applying the "Code Push-Down" development paradigm to the existing data model.

This resulted in the addition of an ABAP CDS View (and consequent HANA CDS View) following SAP's Top-Down approach. As such, this ABAP CDS View will be maintained in the ABAP DDIC and the system will deploy it as a HANA CDS View to the database.

The diagram below represents the resulting data model.

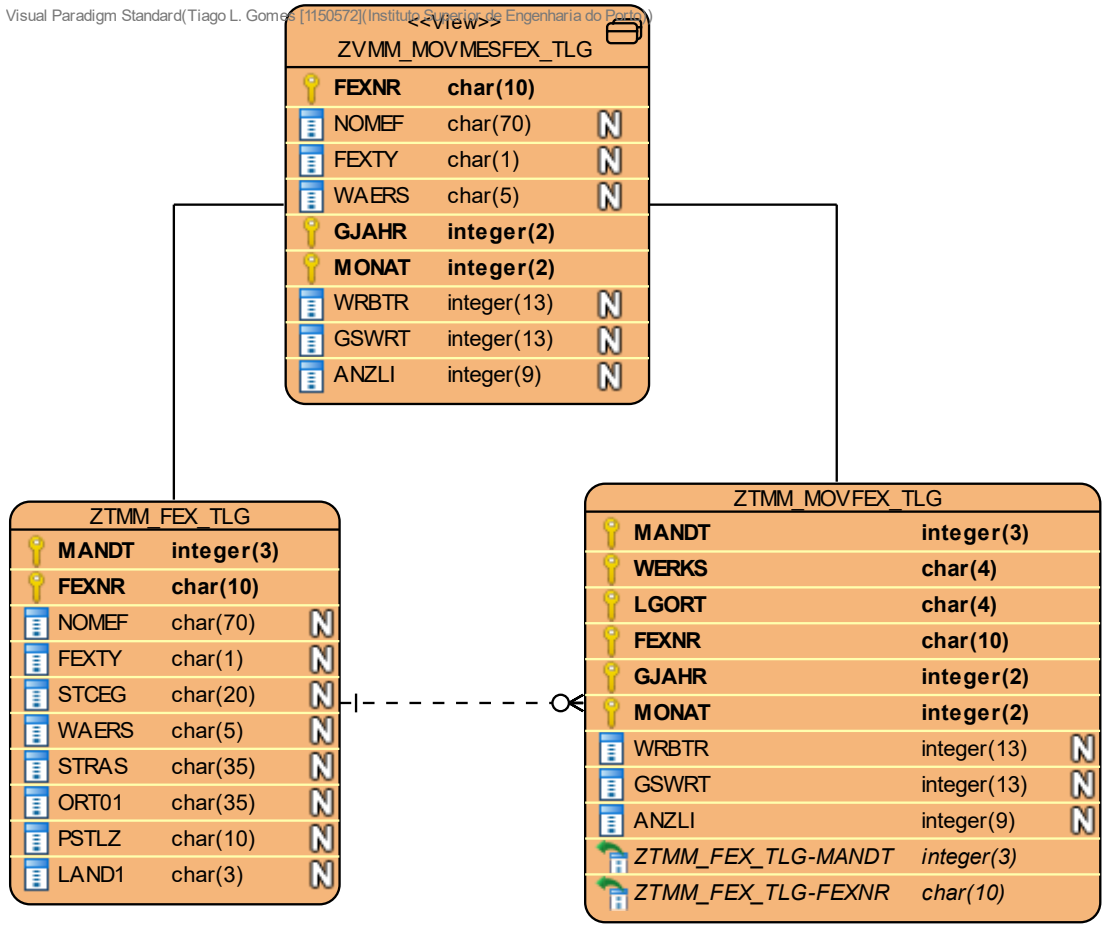


Figure 32 - "Code Push-Down" Data Model

8 Development

This chapter serves to provide information regarding the way the solutions were developed and the developing environment and tools.

This chapter also tries to show how the solutions solve the use cases and remaining requirements. Since from the user's perspective there is no difference between the solutions the information shown will be the same for both.

The developing environments and tools were all gracefully provided by ROFF Consulting [67].

8.1 Development Environment & Tools

This subchapter focuses on showing the developing environment and tools used for the development of both solutions.

8.1.1 SAP ERP

These two solutions were developed using an instance of **SAP S/4 HANA**. Although both these solutions were developed using this ERP version, the "Standard" solution could also be developed in an older version of it.

8.1.2 Database

As implied by the version of the SAP ERP being used, the database is an instance of **SAP HANA**. This said, and as stated previously the developed solutions do not require an SAP HANA database necessarily.

8.1.3 ABAP Application Server

These two solutions were developed in an **SAP ABAP AS 7.54 SP0** application server. This is important as most CDS tools were available after the 7.40 version.

8.1.4 Development Tools

Although there was the use of more tools than the ones mentioned here, these were the ones required for the development of the solutions.

8.1.4.1 SAP GUI

For the development of most of both solutions, **SAP GUI 7.60** was used alongside developing programs inside of SAP ERP itself (SE11, SE38, SE80 mainly).

8.1.4.2 Eclipse + SAP ADT

These tools are a requirement for the use of most CDS tools. These were used specifically to develop the ABAP CDS View used in the "Code Push-Down" solution. **Eclipse 2021-06 + SAP ABAP Development Tools (ADT)** were used.

8.2 Solution

This subchapter intends to try to demonstrate the solution to each of the functional requirements. Although the remaining requirements are not specified in this subchapter, they are all addressed by both the solutions.

It is also important to mention that this subsection will be quite short. Although there is some value to showing how the solutions turned out, their quality and suitability as solutions to this problem, has little to no importance to the present work. What is of importance are the comparisons that can and will be made between them and the resulting conclusions that are in the following chapters.

8.2.1.1 UC - 1: The user must be able to maintain the “External Supplier” table’s data.

The following figure shows the main portion of the resulting solution to UC-1.

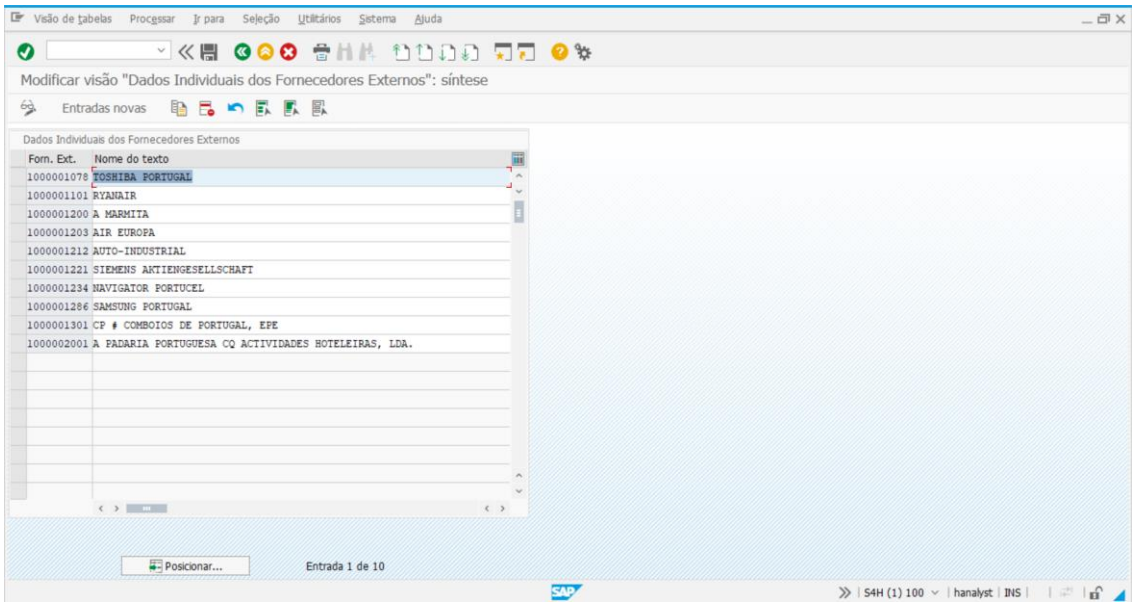


Figure 33 - External Supplier table maintenance “screen” program

8.2.1.2 UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information.

The following figure shows the selection screen portion of the resulting solution to UC-2.

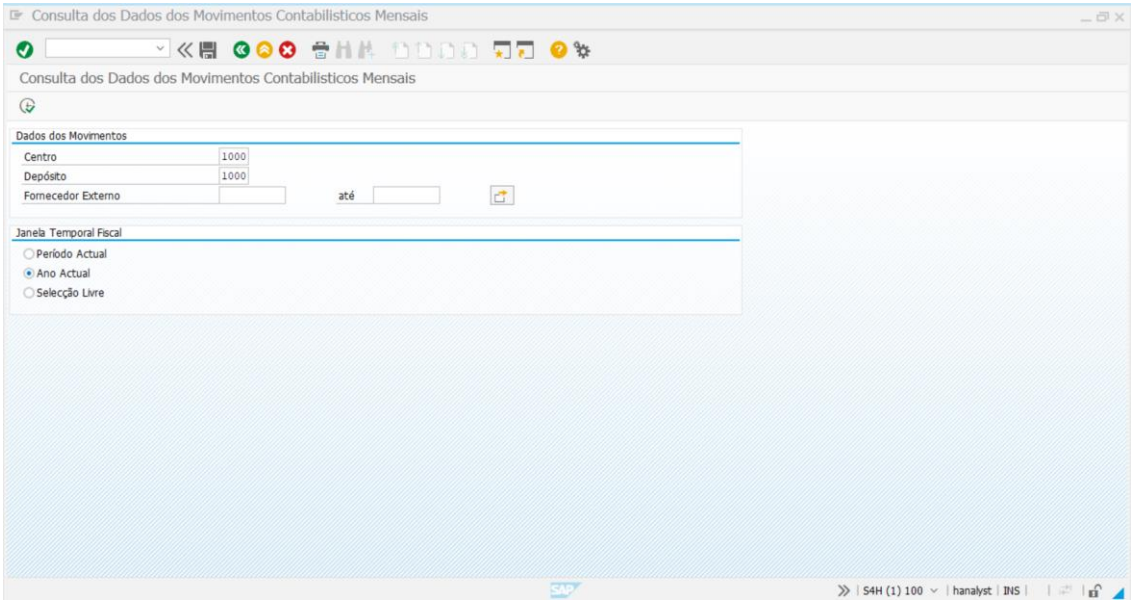
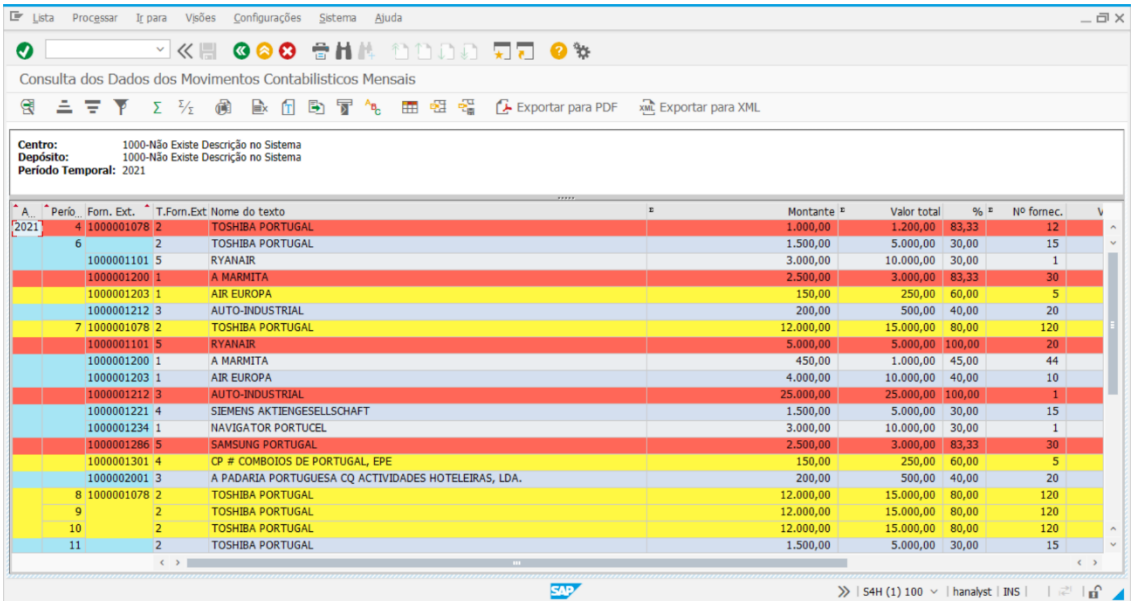


Figure 34 - Reporting Program's Selection-Screen

The following figure shows the main portion of the resulting solution to UC-2.



A...	Perío...	Forn. Ext.	T.Forn.Ext	Nome do texto	Montante	Valor total	%	Nº fornec.
2021	4	1000001078	2	TOSHIBA PORTUGAL	1.000,00	1.200,00	83,33	12
	6		2	TOSHIBA PORTUGAL	1.500,00	5.000,00	30,00	15
		1000001101	5	RYANAIR	3.000,00	10.000,00	30,00	1
		1000001200	1	A MARMITA	2.500,00	3.000,00	83,33	30
		1000001203	1	AIR EUROPA	150,00	250,00	60,00	5
		1000001212	3	AUTO-INDUSTRIAL	200,00	500,00	40,00	20
	7	1000001078	2	TOSHIBA PORTUGAL	12.000,00	15.000,00	80,00	120
		1000001101	5	RYANAIR	5.000,00	5.000,00	100,00	20
		1000001200	1	A MARMITA	450,00	1.000,00	45,00	44
		1000001203	1	AIR EUROPA	4.000,00	10.000,00	40,00	10
		1000001212	3	AUTO-INDUSTRIAL	25.000,00	25.000,00	100,00	1
		1000001221	4	SIEMENS AKTIENGESELLSCHAFT	1.500,00	5.000,00	30,00	15
		1000001234	1	NAVIGATOR PORTUCEL	3.000,00	10.000,00	30,00	1
		1000001286	5	SAMSUNG PORTUGAL	2.500,00	3.000,00	83,33	30
		1000001301	4	CP # COMBOIOS DE PORTUGAL, EPE	150,00	250,00	60,00	5
		1000002001	3	A PADARIA PORTUGUESA CQ ACTIVIDADES HOTELEIRAS, LDA.	200,00	500,00	40,00	20
	8	1000001078	2	TOSHIBA PORTUGAL	12.000,00	15.000,00	80,00	120
	9		2	TOSHIBA PORTUGAL	12.000,00	15.000,00	80,00	120
	10		2	TOSHIBA PORTUGAL	12.000,00	15.000,00	80,00	120
	11		2	TOSHIBA PORTUGAL	1.500,00	5.000,00	30,00	15

Figure 35 - Reporting Program's Report

8.2.1.3 UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform.

The following figure shows the main portion of the resulting solution to UC-3.

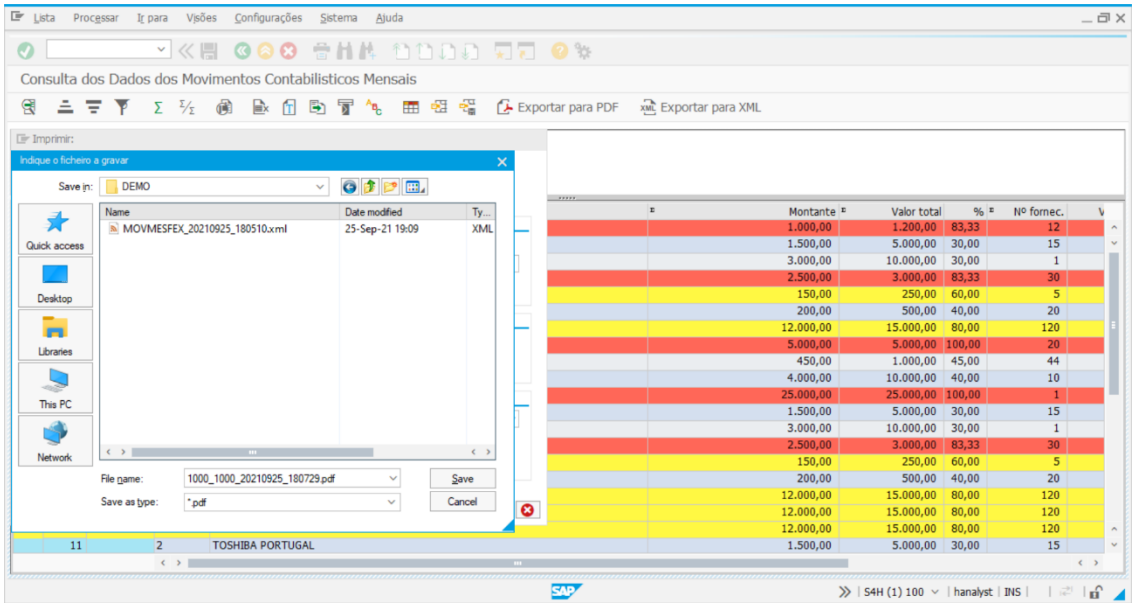


Figure 36 - Reporting Program's PDF Export

The following figure shows the resulting PDF file.

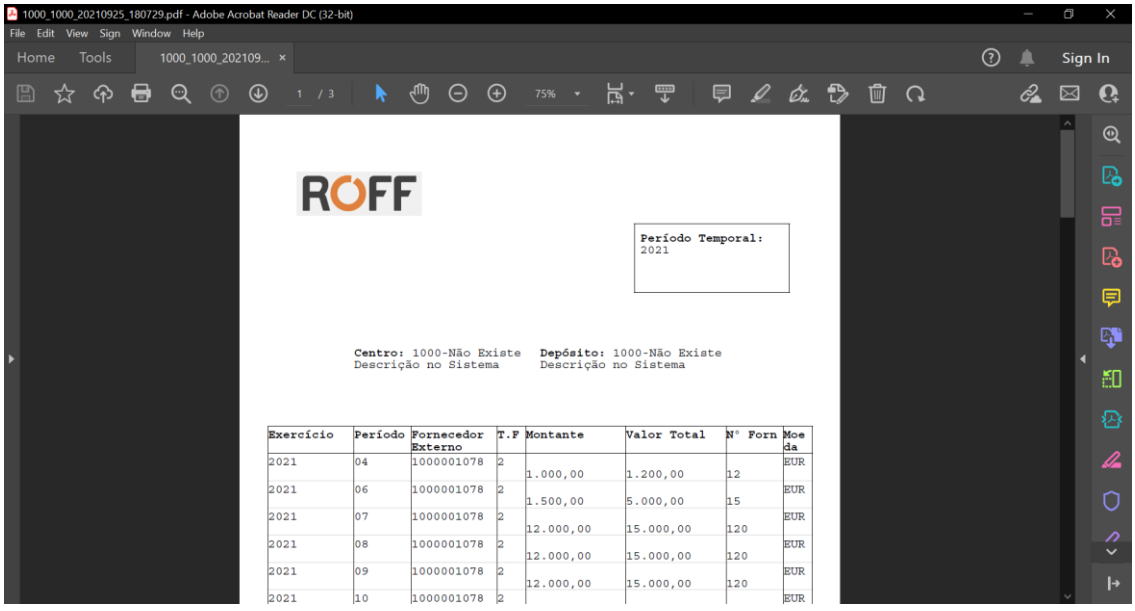


Figure 37 - Reporting Program's Exported PDF File (in Adobe Reader)

8.2.1.4 UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file.

The following figure shows the main portion of the resulting solution to UC-4.

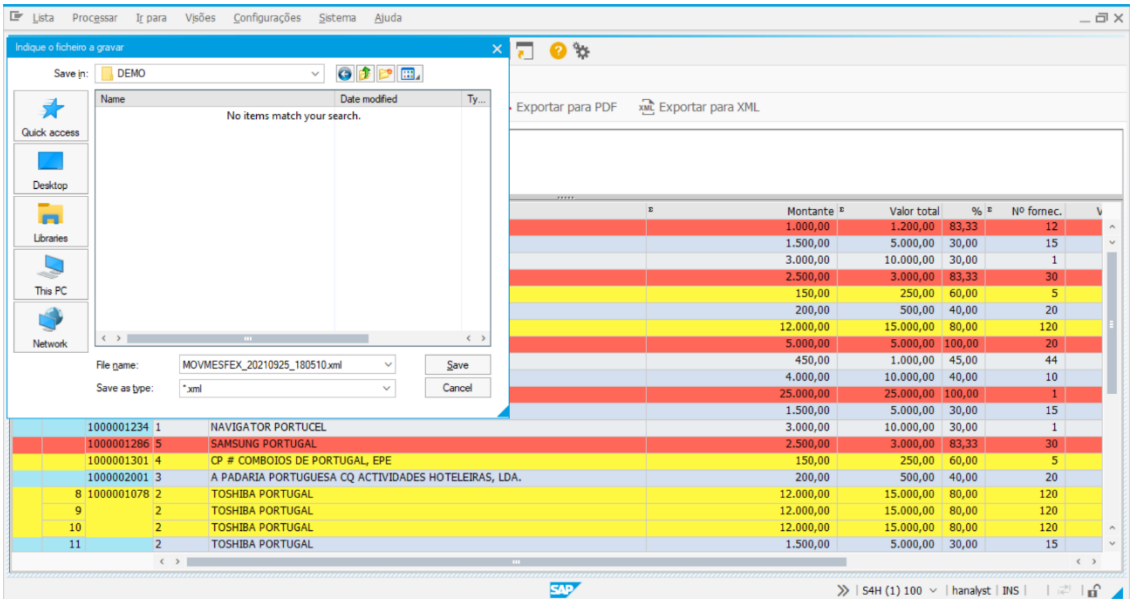


Figure 38 - Reporting Program's XML Export

The following figure shows the resulting XML file.

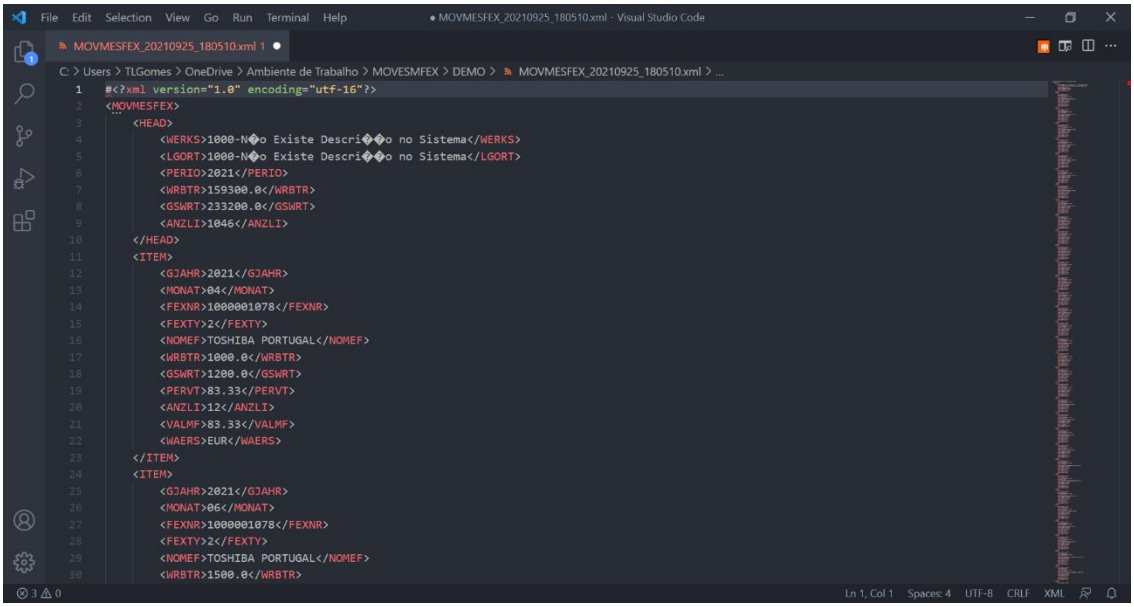


Figure 39 - Reporting Program's XML Exported File (in VS Code)

8.2.1.5 UC - 5: The user must be able to access all transactions via the cockpit's transaction as specified in the problem statement.

The following figure shows the main portion of the resulting solution to UC-5.

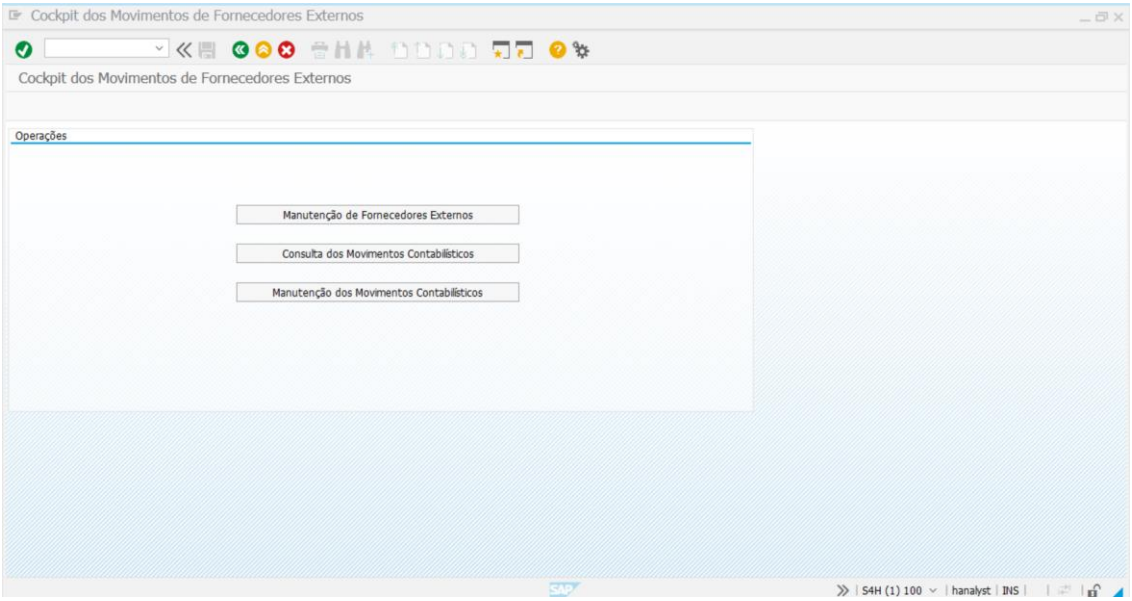


Figure 40 - Cockpit Program's Main Screen

8.2.1.6 UC - 6: The user must be able to insert External Supplier Accounting Movements via the cockpit.

The following figure shows the main portion of the resulting solution to UC-6.

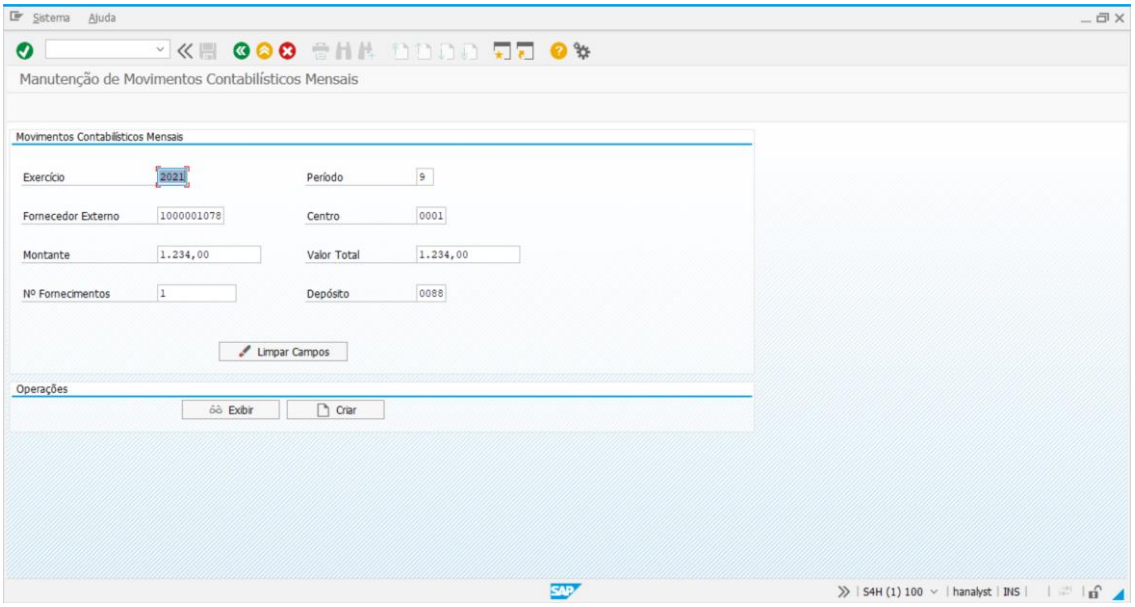


Figure 41 - Cockpits Program's Data Insertion Screen

9 Comparison

This section focuses on comparing both solutions according to the ISO's maintainability and portability criteria for Software Product Quality, as described in the section "Software Architecture".

As also stated in this section, the two criteria were selected because they are the ones that may be more significantly affected by the application of the "Code Push-Down" paradigm.

Since the present work focuses on identifying if there are possible tradeoffs and compromises by applying the "Code Push-Down" development paradigm, this section will focus on where the two solutions differ due to the application of the paradigm.

9.1 Implementation Comparison

Let's start then by comparing both solutions where they differ, and therefore may change the solution's properties.

This comparison will be made on a general solution architecture level, to try and find possible problems on an architecture/solution level. And it will also be made on a functionality level to try and find more specific problems that may appear from a development standpoint.

9.1.1 General Solution Architecture

As stated in the section "General Solution Architecture" for the "Code Push-Down" section of this work, the main differences to the general solution architecture are highlighted in the diagrams below.

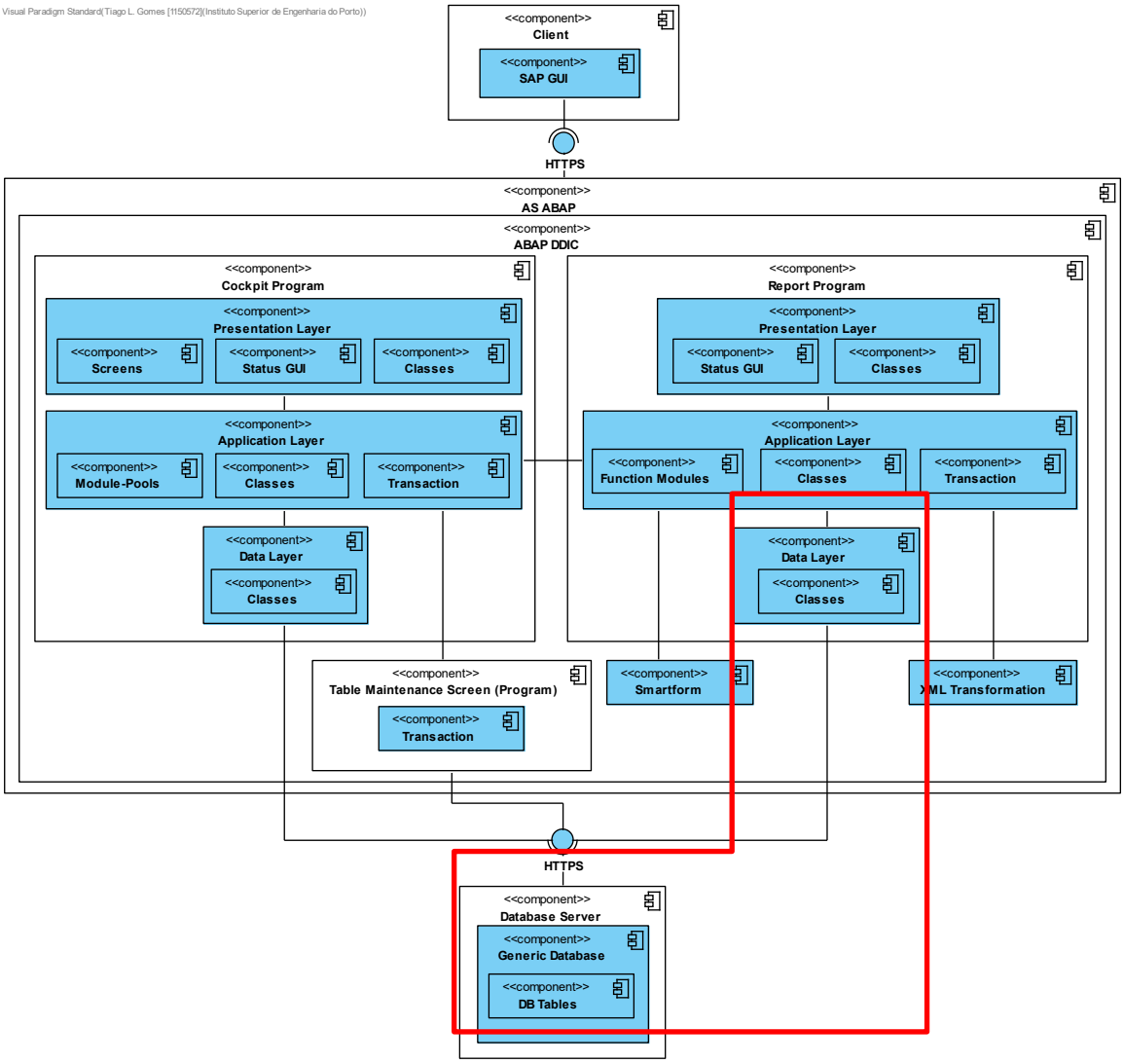


Figure 42 - "Standard" Component Diagram (Changes)

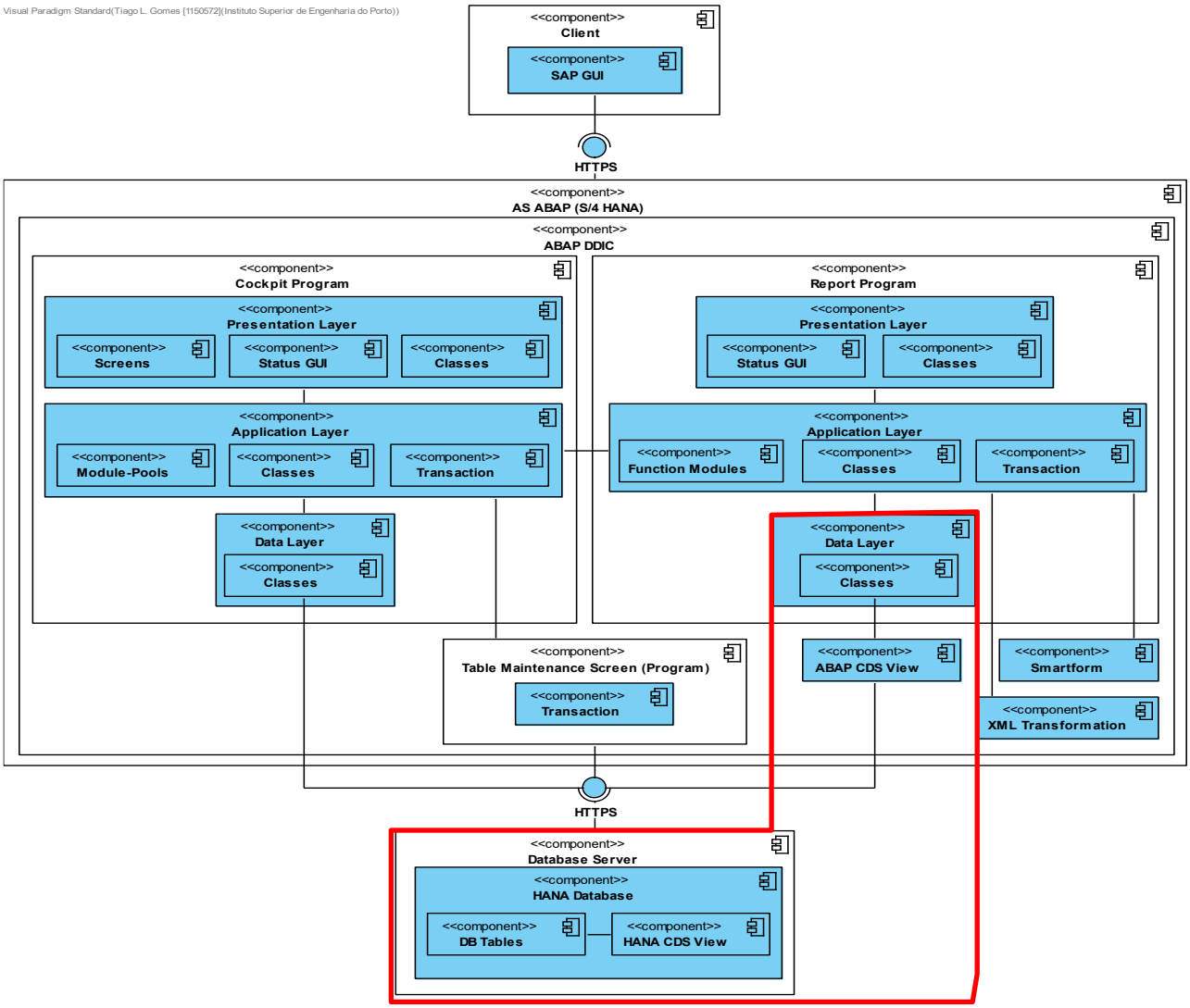


Figure 43 - "Code Push-Down" Component Diagram (Changes)

On a general architecture level, the changes done were the introduction of an ABAP CDS View (and its correspondent database view) maintained by the ABAP data dictionary. This view ends up being used by the report program to query data.

This introduced the need to develop, maintain and use a new database view. For this, the developer has the need to understand and know how to apply ABAP core data services and know how to use SAP ABAP development toolset for its development (Eclipse + SAP ADT).

In this particular case, since the CDS used were ABAP, these are maintained by the ABAP DDIC, meaning that transporting these developments between systems and keeping track of changes are integrated into the ABAP DDIC version control and transport system, if it were the case that HANA CDS were used the difficulty to maintain, and transport said development would be much bigger.

It is also relevant to remember, as mentioned before in the subsection "ABAP Core Data Services (ABAP CDS)", that, although not used for this solution in particular, if ABAP CDS AMDPs were to be used, this solution would end up being only possible with the use of an SAP HANA database as, currently, no other databases support this type of procedure.

9.1.2 Functional Requirement Implementation.

This subchapter will focus on showing the differences in implementation caused by the application of the "Code Push-Down" development paradigm.

Since the new ABAP CDS View is only used by the reporting program, the only functional requirements implemented that could, and end up, being affected are:

- UC - 2: The user must be able to consult the Monthly External Supplier Accounting Movements reporting information;
- UC - 3: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to a PDF Smartform;
- UC - 4: The user must be able to export the Monthly External Supplier Accounting Movements reporting information to an XML file.

This is the case because UC2 is where the access to the database is done and UC3 & UC4 are directly dependent on UC2.

Before the application of the "Code Push-Down" paradigm to this use case, all the business logic related to the data was being done in the application layer.

The following code snippet has the business flow to retrieve and combine the needed data:

```
* Obtem os dados

DATA lr_fexnr TYPE ty_r_fexnr.
DATA lr_gjahr TYPE ty_r_gjahr.
DATA lr_monat TYPE ty_r_monat.

APPEND LINES OF s_fexnr TO lr_fexnr.
APPEND LINES OF s_gjahr TO lr_gjahr.
APPEND LINES OF s_monat TO lr_monat.

go_data_layer->obter_for_ext(
  EXPORTING
    ir_fexnr   = lr_fexnr
  IMPORTING
    et_data    = DATA(lt_for_ext)
  EXCEPTIONS
    ex_no_data = 1
    OTHERS     = 2
).
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  LEAVE PROGRAM.
ENDIF.

go_data_layer->obter_mov_cont_mens(
  EXPORTING
    ir_fexnr   = lr_fexnr
    iv_werks   = p_werks
    iv_lgort   = p_lgort
    ir_gjahr   = lr_gjahr
    ir_monat   = lr_monat
  IMPORTING
    et_data    = DATA(lt_mov_cont_mens)
  EXCEPTIONS
    ex_no_data = 1
    OTHERS     = 2
).
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
*      LEAVE PROGRAM.
  RETURN.
ENDIF.

* Cruza os dados

cruza_dados(
  EXPORTING
    it_for_ext  = lt_for_ext
    it_mov_cont = lt_mov_cont_mens
  IMPORTING
    et_fex_movfex   = DATA(lt_fex_movfex)
).
```

This implementation was retrieving the data from each of the database tables and combining the data, according to the business logic, to later show in the report.

When the “Code Push-Down” paradigm was applied and the ABAP CDS View was introduced, the data related business logic was spread across the Application and Data layer as well as the Application and Data tiers.

The previous snippet was replaced by the following one:

```
* Obtem os dados

    DATA lr_fexnr TYPE ty_r_fexnr.
    DATA lr_gjahr TYPE ty_r_gjahr.
    DATA lr_monat TYPE ty_r_monat.

    APPEND LINES OF s_fexnr TO lr_fexnr.
    APPEND LINES OF s_gjahr TO lr_gjahr.
    APPEND LINES OF s_monat TO lr_monat.

    go_data_layer->obter_dados(
      EXPORTING
        ir_fexnr   = lr_fexnr
        iv_werks   = p_werks
        iv_lgort   = p_lgort
        ir_gjahr   = lr_gjahr
        ir_monat   = lr_monat
      IMPORTING
        et_data    = DATA(lt_fex_movfex)
      EXCEPTIONS
        ex_no_data = 1
        OTHERS     = 2
    ).

    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
      RETURN.
    ENDIF.
```

And the business logic for the data filtering and combining was spread across the data layer with the following Open SQL snippet:

```
SELECT * FROM zvmm_movmesfex_tlg(
                    in_werks = @iv_werks,
                    in_lgort = @iv_lgort )
       WHERE fexnr IN @ir_fexnr
         AND gjahr IN @ir_gjahr
         AND monat IN @ir_monat
       INTO TABLE @et_data.


    IF sy-subrc <> 0.
      MESSAGE s789(m7) DISPLAY LIKE 'E'
        RAISING ex_no_data.
    ENDIF.
```

And spread across the database tier with the following ABAP CDS view:

```
@AbapCatalog.sqlViewName: 'ZV_MOVMESFEX_DB'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@EndUserText.label: 'Dados contabilisticos fornecedores ext.'
define view ZVMM_MOVMESFEX_TLG with parameters in_werks : werks_d,
                                               in_lgort : lgort_d
  as select from    ztmm_fex_tlg
    join ztmm_movfex_tlg on ztmm_movfex_tlg.fexnr = ztmm_fex_tlg.fexnr
{
  key ztmm_fex_tlg.fexnr,
      ztmm_fex_tlg.nomef,
      ztmm_fex_tlg.fexty,
      ztmm_fex_tlg.waers,
      ztmm_movfex_tlg.gjahr,
      ztmm_movfex_tlg.monat,
      ztmm_movfex_tlg.wrbtr,
      ztmm_movfex_tlg.gswrt,
      ztmm_movfex_tlg.anzli
}
where
    werks = $parameters.in_werks and
    lgort = $parameters.in_lgort
```

As mentioned in the subchapter “General Solution Architecture”, the use of an ABAP CDS AMDP would probably allow to concentrate the business logic for the data in the procedure, but the limitations this would impose do not seem to outweigh the benefit in this case.

9.2 Partial Software Product Quality Comparison

Using the information gathered from the comparison of the solutions, this subsection will try to evaluate if and which aspects of the chosen evaluation criteria may be compromised.

As described in the subsection "Software Architecture", these are the criteria to evaluate.

9.2.1 Maintainability

Looking at ISO's description for maintainability in the mentioned subsection, we can say that, yes, the maintainability of the solution is affected in a, mostly, negative way. The following subchapters will show why.

9.2.1.1 Modularity

By spreading the business logic across multiple layers and even multiple tiers, the cohesion of each component is decreased, and the coupling increased. As a result, the modularity was reduced by applying the "Code Push-Down" paradigm.

9.2.1.2 Reusability

By introducing an ABAP CDS View, the reusability of this portion of the business logic itself was improved externally. Before, the data itself was being done exclusively by the Reporting Program and was not able to be easily reused by any future components to the solution. As a result, the reusability was improved by applying the "Code Push-Down" paradigm, at least on an external level.

9.2.1.3 Analyzability

Since the business logic is now spread across multiple layers and tiers and there was the introduction of a new "framework" and as a consequence, new development tools, the analyzability of the solution was hindered as it is harder to find and track possible problems in the solution. As a result, the analyzability was reduced by applying the "Code Push-Down" paradigm.

9.2.1.4 Modifiability

With the spread of business logic across multiple layers and tiers and with the introduction of components that can be reused across multiple other components, the modifiability of the solution may be compromised. This is so because changes to the view can affect multiple other components (even ones external to the solution) and, since the business logic is spread, the number of changes needed to the existing components is likely to be higher. As a result, the modifiability was reduced by applying the “Code Push-Down” paradigm.

9.2.1.5 Testability

The solution’s testability can be affected depending on the type of tests being done. That said, it is likely that this was affected in a negative way, as the cohesion of the components decreased, its coupling increased, and this tends to affect testability in a negative way.

9.2.2 Portability

Looking at ISO’s description for portability in the mentioned subsection, we can say that, in this particular case, the solution’s portability seems to be affected, mostly, in a negative way. The following subchapters will show why.

9.2.2.1 Adaptability

The use of ABAP core data services in the “Code Push-Down” solution means that the version of the ABAP application server version must be equal to or more recent than the 7.40 version. As a result, the adaptability of the solution was reduced by the application of the “Code Push-Down” paradigm.

9.2.2.2 Installability

The changes done to the solution by the use of the “Code Push-Down” paradigm do not seem to affect the installability of the solution.

9.2.2.3 Replaceability

The changes done to the solution by the use of the “Code Push-Down” paradigm do not seem to affect the replaceability of the solution.

9.3 Conclusions

Analyzing the comparison done, one can clearly see that there are significant tradeoffs when applying the “Code Push-Down” development paradigm.

As stated in the chapter “Software Architecture”, the scope of this comparison was narrowed down to ISO’s maintainability and portability software quality attributes. And, as expected, like most development paradigms that leverage the use of database related tools for performance gains, significant compromises were encountered for both software quality attributes.

9.3.1 Implementation

Regarding the implementation, the application logic was spread out more across multiple locations and tools.

The compromises found here are that, in this particular case, and generally when using CDS, the application of the “Code Push-Down” development paradigm requires that the developer knows how to use and implement a bigger set of tools, development environments, and good development practices.

9.3.2 Maintainability

From a maintainability standpoint, a total of 4 out of 5 characteristics were affected in a negative way and 1 of them was affected in a positive way.

This means that, according to ISO, the degree of effectiveness and efficiency with which a product or system can be modified was, mostly, negatively impacted in a significant way.

9.3.3 Portability

From a portability standpoint, a total of 1 out of 3 characteristics were affected in a negative way, while the other remaining 2 remained unchanged.

This means that the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another was also negatively impacted.

It is important to note that, although this particular solution wasn't heavily affected, the impacts of the "Code Push-Down" development paradigm can, quite easily, be more severe when it comes to portability. This is due to the fact that a significant portion of the tools that can be used for "Code Push-Down" can also lock in the solution to a particular database (SAP HANA in most cases), and this can impact one or both of the remaining characteristics in a negative way as well.

10 Guidelines

This chapter takes into account what was learned from the present work, especially from the state-of-the-art and the solution comparison to try to provide general guidelines regarding the "Code Push-Down" paradigm and its tools.

These guidelines will try to provide a pragmatic view on what the "Code Push-Down" development paradigm is and attempts to guide how and when to apply its tools in order to, hopefully, mitigate some of its drawbacks.

10.1 "Code Push-Down" – A Pragmatic Point-Of-View

After analyzing the current state-of-the-art regarding the "Code Push-Down" development paradigm and software development (focused on the type of development targeted by this), one can see that this, despite what SAP seems to make it look like, is not a new development paradigm. It is, however, a new take on an existing one(s).

Leveraging database "tools" to, hopefully, maximize performance isn't anything new, as tools such as database views and procedures have been around for decades. From what we can conclude from the present work, like the existing takes on these existing development paradigms, there are benefits and drawbacks/tradeoffs to the "Code Push-Down" development paradigm. The same ones as why software development, in general, has somewhat moved away from such paradigms.

As such, this paradigm should not be looked at as "THE" development paradigm, but as an alternative solution (or partial solution), or as a set of tools that can be used to achieve certain goals or to promote certain quality attributes within a solution, while sacrificing others.

Good Software development practices teach us that, one should try to provide the best suited solution(s) for a problem(s) based on a solid requirement engineering process, not to try to force a solution to fit every problem.

10.2 "Code Push-Down" – Applying the Paradigm

Although this work determined that there are trade-offs to the application of this development paradigm (mainly maintainability and portability), there are also benefits, according to SAP, mainly when it comes to performance. Performance that is theoretically gained by reducing data transfers while also leveraging the database processing power (in high performance databases)

As such, if performance is of importance to the solution, the application of this development paradigm may be one of the best, easiest, and most supported ways (within the SAP Ecosystem) to promote this solution quality attribute.

Assuming one wants to apply this development paradigm, let us focus on these two points:

10.2.1 When/Where to apply the "Code Push-Down" developing paradigm

As mentioned in the subchapter ""Code Push-Down" Development Paradigm", according to SAP: "Code pushdown means delegating data intensive calculations to the database layer. It does not mean push ALL calculations to the database, but only those that make sense." [10], [42]

From what was learned from the present work we can be more specific about what "… makes sense". So, "Code Push-Down" should be applied to:

Application logic that requires, relative, intense data processing and/or processing of large datasets, of data present in the database, and whose performance is directly correlated to the solutions requirements.

If this guideline is followed, this paradigm should only be applied in cases where there will be a significant performance impact, and in cases where that performance impact is relevant to the requirements.

10.2.2 How to apply the "Code Push-Down" developing paradigm

Now that when this paradigm is to be applied is specified, we can focus on how to apply it.

To apply this paradigm, one needs to use the tools provided by SAP to do it (within the SAP Ecosystem). And to use such tools, one must start by knowing the tools and then know which to choose.

It is important to mention that SAP's Development Guidelines for Database Accesses [69] and specific tool guidelines [38], [51], [52], amongst others, were closely followed alongside the present work to create this subchapter.

Another important thing to note is that, as explained in the chapters "HANA Core Data Services (HANA CDS)" and "ABAP CDS vs. HANA CDS", SAP states that HANA CDS are tools to be used in very specific use cases. And, as explained in the chapter ""Code Push-Down" Development Paradigm", SAP doesn't recommend its use unless strictly necessary.

As such, these guidelines can't be more specific or clear regarding the use of HANA CDS, since the cases where their use is required there is no alternative.

10.2.2.1 Tool Analysis

The available tools and their purpose/use have been specified in the chapter ""Code Push-Down" Development Paradigm", so here let's focus on defining their **main** benefits and drawbacks/Limitations.

- **Open SQL**:
    - **Benefits**:
        - Database Agnostic;
        - Doesn't require new knowledge to use;
        - "Simple".
    - **Drawbacks/Limitations**:
        - Not Reusable;
        - Limited Functionality (When compared CDS Views & CDS AMDP).

- **CDS Views**:
    - **Benefits**:
        - Database Agnostic;
        - Reusable;
        - Extensible;
        - Supports more complex logic (When compared to Open SQL) (e.g.: annotations).
    - **Drawbacks/Limitations**:
        - Requires CDS Knowledge;
        - Requires the use of different development tools;
        - Limited Functionality (When compared to CDS AMDP).
- **CDS ABAP Managed Database Procedures (AMDP)**:
    - **Benefits**:
        - Reusable;
        - Allows for the use of database specific tools;
        - Supports more complex logic (When compared to Open SQL & CDS Views) (e.g.: return of multiple datasets; more complex input parameters; etc.).
    - **Drawbacks/Limitations**:
        - Requires CDS Knowledge;
        - Requires the use of different development tools;
        - Database specific (Currently limited to SAP HANA databases);
        - Requires the use of database specific language (Native SQL; SQL Script; …).

10.2.2.2 Tool Selection

Having knowledge of the available tools, what they do, their benefits and drawbacks/limitations, one now must choose what tool(s) to use. To simplify this choice the following flow chart was made:

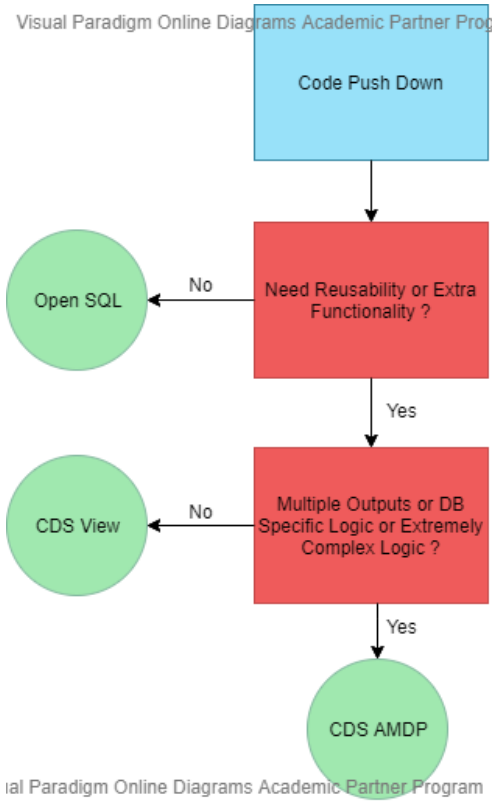


Figure 44 - "Code Push-Down" Decision Tree

It is important to note that this decision tree should only be followed if there aren't requirements that specify what tool to use.

Although there will be a lot of cases where there is more than one tool that can be used to "solve" the requirement(s), this decision tree should help narrow down the choices to the one(s) that provide the least amount, or the least significant, drawbacks/limitations/compromises.

11 Conclusions

This chapter concludes this document. It evaluates the work as a whole and provides some further deductions/conclusions, some opinions, a comparison of what was achieved versus the objectives, alongside difficulties, possible improvements, and future work.

11.1 The “Code Push-Down” Development Paradigm

After the present work, taking out of the “equation” SAP’s “marketing” we can see that this development paradigm, although not new, is a valid take on existing ones. And if applied correctly, it can perhaps have less, or less significant, downsides/compromises, than more “traditional” versions of “Code Push-Down”.

This is due to, when applying Open SQL and/or ABAP CDS, the architectural tiers don’t end up being crossed, unlike the more “traditional” versions of “Code Push-Down”, “just” the architectural layers. This distinction is explained in the subchapter “Tier vs. Layer”.

11.2 SAP’s Posture Regarding “Code Push-Down”

After sifting through a significant portion of documentation, varying from more marketing directed documentation, technical documentation, official blog posts, tutorials, guides, paid/non-public courses (from SAP Learning Hub [70]), etc., my opinion is that SAP’s posture when presenting/pitching the “Code Push-Down” development paradigm, from a marketing perspective is not truthful.

- This development paradigm is not new;
- This development paradigm does not only bring benefits;

- This development paradigm should not always be used.

And SAP's posture when presenting/pitching the "Code Push-Down" development paradigm, from a development/implementation perspective is not suitable.

- Documentation is scarce and hard to procure;
- Documentation is focused on why this paradigm can be beneficial and how it works but rarely/never on drawbacks/limitations;
- No Guidelines (or not specific enough) on how/when to apply the "Code Push-Down" tools.

11.3 Achieved Objectives

I believe the present work achieves with success all the objectives set.

The present work was set to provide insight on the status of the SAP Ecosystem, scrutinize the "Code Push-Down" development paradigm, and the claims surrounding it and finally to provide guidelines on how it can be used, if/when suitable.

The present work, although not set as an objective, was also meant to try to identify other possible problems in the current SAP Ecosystem and try to instigate further discussion/work about this topic the others found. And, independently of the actual quality of the present work, I think it manages to do so as well.

11.4 Difficulties

During the elaboration of the present work, these were the main difficulties felt:

- Focusing the scope of the work, when contextualizing and defining the current SAP Ecosystem multiple problems became apparent regarding this shift in paradigm;
- Determining a suitable way to analyze and present the "Code Push-Down" development paradigm;
- Finding suitable documentation: SAP's available documentation, is often redundant, spread across many platforms (some of whom are not public) and its quality varies significantly;
- Finding a balance of relevance and completeness regarding the development of the state-of-the-art for software development;

- Keeping consistency throughout the document when the knowledge about the topic and circumstances changes so drastically throughout time.

11.5 Improvements

Throughout the development of the present work, there were a few compromises that had to be made in order to try to stay within schedule. These are the main ones done or work that could be done to add value to the work already done.

- The present work could benefit from the following surveys:
    - Survey to assess the current awareness of this development paradigm amongst developers, their thoughts on it, and their commitment on its use;
    - Survey to gauge the priority given to ISO's software quality attributes regarding custom developments;
    - Survey to gauge the "felt" benefits & tradeoffs of the application of the "Code Push-Down" development paradigm
- The present work could also benefit from comparing both solutions from a performance standpoint;
- The present work could also benefit from studying "edge cases" when it comes to developments, both in terms of abnormality in requirements and in terms of complexity.

11.6 Future Work

This section refers to work that could be done standalone or as a continuation of the present work. This section is heavily based on the subchapter "Idea Generation and Enrichment". It is also important to mention that the improvements mentioned above also are "future work".

- Studies on the other possible problems raised by the switch to SAP S/4 HANA & SAP HANA:
    - Possible SAP HANA database lock-in;
    - Guidelines on the Migration of existing solutions to the "Code Push-Down" Development Paradigm;
    - "Code Push-Down" Development Guidelines for Edge Cases.

- SAP HANA Database Performance Comparison;
- A study on the apparent Database Lock-in and possible “middleware” creation.

References

[1] D. E. O'Leary, *Enterprise Resource Planning Systems : Systems, Life Cycle, Electronic Commerce, and Risk*. Cambridge, UK: Cambridge University Press, 2000. Accessed: Dec. 07, 2020. [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=77574&lang=pt-pt&site=ehost-live&scope=site

[2] S. Jacobson, J. Shepherd, M. D'Aquila, and K. Carter, "The ERP Market Sizing Report, 2006–201," p. 36, 2007.

[3] L. Columbus, "Gartner's ERP Market Share Update Shows The Future Of Cloud ERP Is Now," *Forbes*, May 12, 2014. https://www.forbes.com/sites/louiscolumbus/2014/05/12/gartners-erp-market-share-update-shows-the-future-of-cloud-erp-is-now/ (accessed Dec. 07, 2020).

[4] Y. Dharmasthira, C. Eschinger, C. Pang, K. Brant, and K. Motoyoshi, "Market Share Analysis: ERP Software, Worldwide, 2012," *Gartner*, May 07, 2013. https://www.gartner.com/en/documents/2477517/market-share-analysis-erp-software-worldwide-2012 (accessed Dec. 07, 2020).

[5] "From inventing the enterprise software sector to helping the world run better," *SAP*, 2020. https://www.sap.com/documents/2020/02/70eee289-847d-0010-87a3-c30de2ffd8ff.html (accessed Dec. 07, 2020).

[6] SAP, "SAP Support Strategy," 2020. https://support.sap.com/en/offerings-programs/strategy.html (accessed Dec. 07, 2020).

[7] SAP, "End of Maintenance Information," Jul. 26, 2016. https://archive.sap.com/documents/docs/DOC-8280 (accessed Dec. 07, 2020).

[8] T. Saueressig, "IDC Survey on SAP S/4HANA Customer Migration," *SAP News Center*, Jun. 27, 2019. https://news.sap.com/2019/06/sap-s4hana-customer-migration-idc-survey/ (accessed Mar. 01, 2021).

[9] S. Krishnamurthy, "Code Push-Down for HANA Starts with ABAP Open SQL | SAP Blogs," Sep. 26, 2014. https://blogs.sap.com/2014/09/26/code-push-down-for-hana-from-abap-starts-with-open-sql/ (accessed Nov. 29, 2020).

[10] J. Weiler, "ABAP for HANA and 'Code Push-Down' | SAP Blogs," Feb. 03, 2014. https://blogs.sap.com/2014/02/03/abap-for-hana-code-push-down/ (accessed Nov. 29, 2020).

[11] SAP, "What is ERP?," *SAP Insights*, Jun. 11, 2020. https://insights.sap.com/what-is-erp/ (accessed Jan. 03, 2021).

[12] Oracle, "What is a database?," *Database Topics*, Jan. 26, 2021. https://www.oracle.com/pt/database/what-is-database/ (accessed Jan. 26, 2021).

[13] Google, "Database migration: Concepts and principles (Part 1) | Solutions," *Google Cloud*, Jan. 26, 2021. https://cloud.google.com/solutions/database-migration-concepts-principles-part-1 (accessed Jan. 26, 2021).

[14] Google, "database-migration-concepts-principles-part-1-migration-process.png (639×214)," *DB Migration Process*, Jan. 26, 2021. https://cloud.google.com/solutions/images/database-migration-concepts-principles-part-1-migration-process.png (accessed Jan. 26, 2021).

[15] M. Fowler, "Design - Who needs an architect?," *IEEE Softw.*, vol. 20, no. 5, pp. 11–13, Sep. 2003, doi: 10.1109/MS.2003.1231144.

[16] M. W. Maier, D. Emery, and R. Hilliard, “Software architecture: introducing IEEE Standard 1471,” *Computer*, vol. 34, no. 4, pp. 107–109, Apr. 2001, doi: 10.1109/2.917550.
[17] Red Hat, “What is an application architecture?,” Jan. 28, 2021. https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture (accessed Jan. 30, 2021).
[18] M. Bertoa, “Fig. 2. Software product quality model in ISO/IEC 25010,” *ResearchGate*, Sep. 2013. https://www.researchgate.net/figure/Software-product-quality-model-in-ISO-IEC-25010_fig1_256460076 (accessed Feb. 02, 2021).
[19] ISO, “ISO/IEC 25010:2011,” *ISO*, Mar. 2011. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/57/35733.html (accessed Jan. 31, 2021).
[20] Microsoft, “3-tier deployment,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/images/ee658120.2e360038-9224-4ce3-a901-4eebd2eef2c7(en-us,pandp.10).png (accessed Feb. 04, 2021).
[21] Microsoft, “Microsoft Application Architecture Guide, 2nd Edition - Chapter 19: Physical Tiers and Deployment,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658120(v=pandp.10) (accessed Feb. 04, 2021).
[22] IBM, “What is Three-Tier Architecture,” Oct. 30, 2020. https://www.ibm.com/cloud/learn/three-tier-architecture (accessed Feb. 02, 2021).
[23] Microsoft, “4-tier deployment,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/images/ee658120.0dcbd491-f321-408e-9f94-b0561bf46478(en-us,pandp.10).png (accessed Feb. 04, 2021).
[24] M. Fowler, “PresentationDomainDataLayering,” *martinfowler.com*, Aug. 25, 2015. https://martinfowler.com/bliki/PresentationDomainDataLayering.html (accessed Jan. 30, 2021).
[25] Microsoft, “The logical architecture view of a layered system,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/images/ee658109.a4691b48-1b2c-4102-984d-4fd1233f369d(en-us,pandp.10).png (accessed Feb. 03, 2021).
[26] Microsoft, “Microsoft Application Architecture Guide, 2nd Edition - Chapter 5: Layered Application Guidelines,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109(v=pandp.10) (accessed Feb. 03, 2021).
[27] Microsoft, “Microsoft Application Architecture Guide, 2nd Edition - Chapter 23: Designing Rich Internet Applications,” Jan. 13, 2010. https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658083(v=pandp.10) (accessed Feb. 06, 2021).
[28] C. Larman, “Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Pr,” p. 616, Jan. 2002.
[29] “ESOFT 2015-2016 Requisitos, Análise e Design (T) (IT1).pdf.” Accessed: Feb. 08, 2021. [Online]. Available: https://moodle2.isep.ipp.pt/pluginfile.php/101967/mod_resource/content/0/ESOFT%202015-2016%20Requisitos%2C%20Ana%CC%81lise%20e%20Design%20%28T%29%20%28IT1%29.pdf
[30] OMG, “Unified Modeling Language, v2.5.1,” *Unified Model. Lang.*, p. 796, Dec. 2017.
[31] “ESOFT 2015-2016 Design OO (T) IT2.pdf.” Accessed: Feb. 08, 2021. [Online]. Available: https://moodle2.isep.ipp.pt/pluginfile.php/106043/mod_resource/content/0/ESOFT%202015-2016%20Design%20OO%20%28T%29%20IT2.pdf

[32] SAP, “The SAP R/3 era | SAP History | About SAP SE,” *SAP*, 2020. https://www.sap.com/corporate/en/company/history/1991-2000.html (accessed Feb. 28, 2021).
[33] SAP, “SAP HANA in a Classic 3-tier Architecture - SAP Help Portal,” Jun. 26, 2020. https://help.sap.com/viewer/b3ee5778bc2e4a089d3299b82ec762a7/2.0.05/en-US/059031f3ba4b40bcb5dbbbe46cb21235.html (accessed Feb. 27, 2021).
[34] S. Elfner, “TEC206: Architecture and Components of SAP S/4HANA, 2015 Las Vegas,” *SAP TechEd Archive*, 2015. http://events.sap.com/teched/en/session/22662 (accessed Feb. 28, 2021).
[35] SAP, “ABAP Programming Language - Overview - ABAP Keyword Documentation,” 2020. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenabap_overview.htm (accessed Mar. 01, 2021).
[36] SAP, “ABAP Dictionary,” 2020. https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/en-US/cf/21ea0b446011d189700000e8322d00/frameset.htm (accessed Mar. 01, 2021).
[37] “ABAP Dictionary - ABAP Keyword Documentation.” https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenabap_dictionary_glosry.htm (accessed Mar. 03, 2021).
[38] “Open SQL - ABAP Keyword Documentation.” https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-US/abenopen_sql_glosry.htm (accessed Sep. 11, 2021).
[39] “Open SQL interface - ABAP Keyword Documentation.” https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-US/abenopen_sql_interface_glosry.htm (accessed Sep. 11, 2021).
[40] SAP, “ABAP Keyword Documentation - Select Expression.” https://help.sap.com/doc/abapdocu_740_index_htm/7.40/en-US/index.htm?file=ABAPSQL_EXPR.htm (accessed Sep. 11, 2021).
[41] T. Sharma, “ABAP Core Data Services – Introduction (ABAP CDS view) | SAP Blogs,” Jul. 09, 2017. https://blogs.sap.com/2017/09/09/abap-core-data-services-introduction-abap-cds-view/ (accessed Nov. 29, 2020).
[42] SAP, “ABAP Core Data Services | S/4HANA - Best Practice Guide,” *SAP*, Feb. 2020. https://www.sap.com/documents/2019/01/0e6d5904-367d-0010-87a3-c30de2ffd8ff.html (accessed Mar. 05, 2021).
[43] R. Kumaria, “The Semantically Rich Data Model – An ABAP based CDS Views example | SAP Blogs,” Aug. 30, 2016. https://blogs.sap.com/2016/08/30/the-semantically-rich-data-model-an-abap-based-cds-views-example/ (accessed Mar. 06, 2021).
[44] A. Belati and F. Alomari, *An Overview of SAP Core Data Services*. 2020.
[45] SAP, “ABAP CDS in ABAP Dictionary - ABAP Keyword Documentation,” 2016. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abencds.htm (accessed Mar. 06, 2021).
[46] SAP, “ABAP CDS - Performance Note - ABAP Keyword Documentation,” 2016. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenabap_cds_perfo.htm (accessed Mar. 06, 2021).
[47] M. Ahmed, “ABAP on SAP HANA. Part IV. Core Data Services | SAP Yard |,” Jun. 28, 2016. https://sapyard.com/abap-on-sap-hana-part-iv/ (accessed Mar. 06, 2021).
[48] SAP, “CDS Languages,” 2016. https://sapyard.com/wp-content/uploads/2016/06/9-2.jpg?x55810 (accessed Mar. 06, 2021).

[49] SAP, “ABAP Core Data Services - ABAP Keyword Documentation,” 2016. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenabap_core_data_services_glosry.htm (accessed Mar. 06, 2021).
[50] SAP, “CDS DDL,” 2016. https://blogs.sap.com/wp-content/uploads/2017/08/2-8.jpg (accessed Mar. 06, 2021).
[51] SAP, “ABAP CDS - Views - ABAP Keyword Documentation,” 2016. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenddic_cds_views.htm (accessed Mar. 06, 2021).
[52] SAP, “ABAP CDS - Table Functions - ABAP Keyword Documentation,” 2016. https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-us/abenddic_cds_table_functions.htm (accessed Mar. 06, 2021).
[53] H. Keller, “ABAP Language News for Release 7.40, SP05 | SAP Blogs.” https://blogs.sap.com/2014/02/06/abap-news-for-release-740-sp05/ (accessed Sep. 11, 2021).
[54] SAP, “AMDP - ABAP Managed Database Procedures - ABAP Keyword Documentation.” https://help.sap.com/doc/abapdocu_752_index_htm/7.52/en-US/abenamdp.htm?file=abenamdp.htm (accessed Sep. 11, 2021).
[55] SAP, “ABAP Keyword Documentation - AMDP - ABAP Managed Database Procedures.” https://help.sap.com/doc/abapdocu_740_index_htm/7.40/en-US/index.htm?file=ABENAMDP.htm (accessed Sep. 11, 2021).
[56] H. Keller, “CDS – One Concept, Two Flavors | SAP Blogs,” Jul. 20, 2015. https://blogs.sap.com/2015/07/20/cds-one-model-two-flavors/ (accessed Mar. 06, 2021).
[57] “New Core Data Services Features in SAP HANA 1.0 SPS 10 | SAP Blogs.” https://blogs.sap.com/2015/07/01/new-core-data-services-features-in-sap-hana-10-sps-10/ (accessed May 08, 2021).
[58] “Getting Started with Core Data Services - SAP Help Portal.” https://help.sap.com/viewer/09b6623836854766b682356393c6c416/2.0.02/en-US/b710731496cf43b7ba76e15a928f1a80.html (accessed Mar. 05, 2021).
[59] “TEMPLATE_image004.gif (480×359),” Dec. 05, 2021. https://help.sap.com/saphelp_scm70/helpdata/ru/97/68d64260752a78e10000000a155106/TEMPLATE_image004.gif (accessed May 12, 2021).
[60] “topdown_377308.png (1525×625).” https://blogs.sap.com/wp-content/uploads/2014/02/topdown_377308.png (accessed May 22, 2021).
[61] “bottomup_377307.png (1558×671).” https://blogs.sap.com/wp-content/uploads/2014/02/bottomup_377307.png (accessed May 22, 2021).
[62] C. Kersten, “Oracle for SAP Database Update,” p. 16, Jun. 2020.
[63] N. Rich and H. Matthias, “Value analysis. Value engineering: Innoregio: dissemination of innovation and knowledge management techniques,” p. 32, Jan. 2000.
[64] P. A. Koen *et al.*, “Fuzzy Front End: Effective Methods, Tools, and Techniques,” *PDMA ToolBook New Prod. Dev.*, p. 32, 2002.
[65] P. Koen *et al.*, “Providing Clarity and Common Language to the Fuzzy Front End,” *Res.-Technol. Manag.*, vol. 44, pp. 46–55, Mar. 2001.
[66] T. L. Saaty, “Decision making with the analytic hierarchy process,” *Int. J. Serv. Sci.*, vol. 1, no. 1, pp. 83–98, Jan. 2008, doi: 10.1504/IJSSci.2008.01759.
[67] “ROFF Consulting,” *ROFF Consulting*. https://www.roffconsulting.com/en (accessed Jul. 03, 2021).
[68] P. Eeles, “Capturing Architectural Requirements,” Nov. 2001.

[69] SAP, “ABAP Keyword Documentation - Database Accesses.” https://help.sap.com/doc/abapdocu_740_index_htm/7.40/en-US/index.htm?file=abendatabase_access_guidl.htm (accessed Sep. 11, 2021).
[70] “SAP Learning Hub.” https://learninghub.sap.com/ (accessed Sep. 30, 2021).

Appendix A

Desenho Funcional

S00124_2017 – Gestão dos Fornecedores Externos

Sistema:	SAP ERP
Referência:	DF_SAP_MM_S00124_2017
Data:	27.07.2017
Versão:	4.0

Autor:	José Bento
Descrição:	Desenho funcional do módulo de gestão dos fornecedores externos

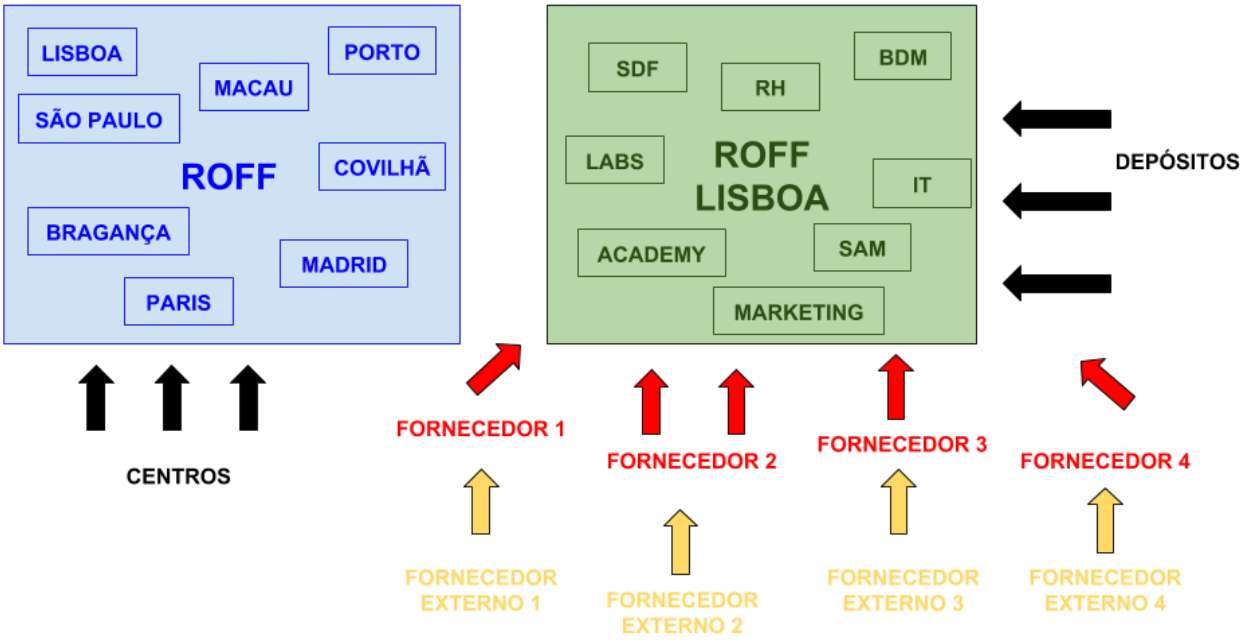
Histórico do Documento:

Versão	Data	Responsável	Descrição de Alterações
1.0	10.07.2017	José Bento	Versão Inicial
2.0	17.07.2017	José Bento	Versão revista
3.0	18.07.2017	José Bento	Versão revista
4.0	27.07.2017	José Bento	Versão revista

ÍNDICE

1 - Âmbito do Documento

Este documento tem como objetivo especificar os requisitos técnicos e funcionais necessários ao desenvolvimento dum módulo de gestão dos fornecedores externos da ROFF. Presentemente, a ROFF é fornecida, tanto ao nível de bens como de serviços, por diversos fornecedores tendo-se identificado que estes também recorrem a fornecedores próprios. Uma vez que a Unidade de Controlo e Gestão (UCG) da ROFF irá começar a efetuar auditorias à estrutura de custos a partir do próximo ano fiscal, é necessário implementar um módulo que permita guardar e consultar dados mensais do volume de bens e serviços fornecidos indiretamente pelos fornecedores externos com vista a uma eventual adjudicação direta com o objetivo de reduzir custos.



2 - Plano

Para o desenvolvimento requerido será necessário um recurso técnico para tratar das implementações ABAP. Os testes e posteriores ajustes serão efetuados pela equipa de 1ª linha da ROFF.

3 - Requisitos Técnicos e Funcionais

Para gerir os dados dos fornecedores externos será necessário definir duas tabelas na BD, uma para guardar os dados individuais dos fornecedores externos e outra para guardar os dados dos movimentos contabilísticos mensais. Ambas as tabelas serão não ampliáveis.

A tabela onde serão guardados os dados individuais deverá possuir um ecrã de manutenção e uma transação de acesso para permitir a sua manutenção pela UCG. Esta tabela terá como nome **ZTMM_FEX_<Iniciais>** e possuirá a seguinte estrutura:

Nome do Campo	Elemento de Dados	Descrição
MANDT*	MANDT	Mandante
FEXNR*	ZMM_FEXNR_<Iniciais>	Fornecedor Externo. Será um campo CHAR de tamanho 10. Terá associada uma rotina de conversão de modo a permitir o controlo dos zeros à esquerda.
NOMEF	BAPITDOBNAME	Nome do Fornecedor Externo
FEXTY	ZMM_FEXTY_<Iniciais>	Tipo do Fornecedor Externo. Será um campo CHAR de tamanho 1 e que deverá assumir apenas 1 valor da seguinte lista: • 1 – Fornecedor Ativo • 2 – Fornecedor Antigo • 3 – Fornecedor Eventual • 4 – Fornecedor Auxiliar • 5 – Outro
STCEG	STCEG	NIF
WAERS	WAERS	Moeda
STRAS	STRAS_GP	Rua e Nº
ORT01	ORT01_GP	Localidade
PSTLZ	PSTLZ	Código Postal
LAND1	LAND1_GP	País

* Campos-chave

A transação de acesso ao ecrã de manutenção desta tabela terá como nome **ZMM_FEX_<Iniciais>**.

A tabela onde serão guardados os dados dos movimentos contabilísticos mensais terá como nome **ZTMM_MOVFEX_<Iniciais>** e possuirá a seguinte estrutura:

Nome do Campo	Elemento de Dados	Descrição
MANDT*	MANDT	Mandante
WERKS*	WERKS_D	Centro
LGORT*	LGORT_D	Depósito
FEXNR*	ZMM_FEXNR_<Iniciais>	Fornecedor Externo. Será um campo CHAR de tamanho 10. Terá associada uma rotina de conversão de modo a permitir o controlo dos zeros à esquerda.

GJAHR*	GJAHR	Exercício
MONAT*	MONAT	Período
WRBTR	WRBTR	Montante
GSWRT	GSWRT	Valor Total
ANZLI	MC_ANZLI	Nº de fornecimentos

* Campos-chave

Deverá ser definida a ajuda de pesquisa **ZSHMM_FEXNR_<Iniciais>** para permitir a pesquisa de fornecedores externos. Esta ajuda de pesquisa usará a tabela ZTMM_FEX_<Iniciais> como método de seleção e será disponibilizada em todos os ecrãs que tenham o Fornecedor Externo como campo de seleção.

Será desenvolvido um programa de reporte com vista à consulta dos dados dos movimentos contabilísticos mensais. Este programa terá como nome **ZRMM_MOVMESFEX_<Iniciais>** e terá os seguintes elementos no seu ecrã de seleção:

- Centro – Parâmetro de seleção simples, obrigatório
- Depósito – Parâmetro de seleção simples, obrigatório
- Fornecedor Externo – Parâmetro de seleção múltipla
- Período Atual – Parâmetro *radiobutton,* selecionado por omissão
- Ano Atual – Parâmetro *radiobutton*
- Seleção Livre – Parâmetro *radiobutton*
- Exercício – Parâmetro de seleção múltipla
- Período – Parâmetro de seleção múltipla

Para executar o programa, o utilizador deverá preencher os campos “Centro” e “Depósito” e indicar, através dos parâmetros *radiobutton*, qual o período temporal que deseja consultar. Os campos “Exercício” e “Período” deverão estar ocultos exceto quando o utilizador selecionar a opção “Seleção Livre”, situação na qual estarão disponíveis.

No final da execução do programa será apresentada uma listagem de saída, em formato ALV Grid que possuirá a seguinte estrutura:

Nome do Campo	**Elemento de Dados**	**Descrição**
GJAHR	GJAHR	Exercício
MONAT	MONAT	Período
FEXNR	ZMM_FEXNR_<Iniciais>	Fornecedor Externo
FEXTY	ZMM_FEXTY_<Iniciais>	Tipo do Fornecedor Externo
NOMEF	BAPITDOBNAME	Nome do Fornecedor Externo
WRBTR	WRBTR	Montante
GSWRT	GSWRT	Valor Total
PERVT	PRZOPKUM	Percentagem do Valor Total (WRBTR / GSWRT)
ANZLI	MC_ANZLI	Nº de fornecimentos
VALMF	BINV_VALUE	Valor Médio por Fornecimento (WRBTR / ANZLI)
WAERS	WAERS	Moeda

Todas as entradas cuja Percentagem do Valor Total tenha um valor entre 50% e 80% deverão ser pintadas de cor amarela; todas as entradas cuja Percentagem do Valor Total seja superior a 80% deverão ser pintadas de cor vermelha.

A listagem de saída ALV deverá estar ordenada por Exercício, Período e Fornecedor Externo sendo que estes campos também deverão estar assinalados como campos-chave.

Deverá ser apresentado, no final da listagem, o somatório do total dos campos Montante, Valor Total e Nº de Fornecimentos.

No topo a listagem de saída deverá ser apresentado um cabeçalho ALV contendo três informações:

o Centro indicado no ecrã de seleção e a sua descrição;

o Depósito indicado no ecrã de seleção e a sua descrição;

o Período temporal de pesquisa indicado no ecrã de seleção:

Mês e Ano, caso se tenha selecionado o Período Atual;

Ano, caso se tenha selecionado o Ano Atual;

Seleção Livre, caso se tenha selecionado uma seleção livre qualquer.

A transação para acesso ao programa ZRMM_MOVMESFEX_<Iniciais> será **ZMM_MOVMESFEX_<Iniciais>**.

Na barra de botões da listagem de saída em formato ALV Grid deverá ser acrescentado um botão novo "**Exportar para PDF**" que terá associada a funcionalidade de exportar a listagem de saída para um formulário Smartform e guardá-lo num ficheiro local em formato PDF.

Deverá ser apresentada ao utilizador uma janela *popup* para este indicar a localização e o nome do ficheiro a gravar no seu computador. Deverá ser pré-preenchido o nome do ficheiro que será:

<CENTRO>_<DEPÓSITO>_<DATA_ACTUAL>_<HORA_ACTUAL>.PDF

O formulário deverá possuir a seguinte estrutura:

o logótipo da ROFF no canto superior esquerdo;

o Centro e o Depósito e as respetivas descrições no topo;

o Período temporal de pesquisa no canto superior direito:

Mês e Ano, caso se tenha selecionado o Período Atual;

Ano, caso se tenha selecionado o Ano Atual;

Seleção Livre, caso se tenha selecionado uma seleção livre qualquer;

uma tabela com os dados da listagem de saída ALV Grid com os respetivos cabeçalhos listando os campos Exercício, Período, Fornecedor Externo, Tipo de Fornecedor Externo, Montante, Valor Total, Nº Fornecimentos e Moeda e um rodapé com o total dos campos Montante, Valor Total e Nº Fornecimentos;

a indicação da página / total de páginas, do utilizador que gerou o ficheiro e a data de geração do ficheiro no canto inferior direito.

Na barra de botões da listagem de saída em formato ALV Grid deverá ser acrescentado um botão novo "**Exportar para XML**" que terá associada a funcionalidade de exportar a listagem de saída para um ficheiro local em formato XML com a seguinte estrutura:

<MOVMESFEX>

<HEAD>

<WERKS>…</WERKS> * o Centro e a respetiva descrição

<LGORT>…</LGORT> * o Depósito e a respetiva descrição

<PERIO>…</PERIO> * o Período temporal de pesquisa

<WRBTR>…</WRBTR> * o total do Montante

```
        <GSWRT>…</GSWRT>                    * o total do Valor Total
        <ANZLI>…</ANZLI>                    * o total do Nº Fornecimentos
    </HEAD>
    <ITEM>                                  * conteúdo de cada linha da listagem
        <GJAHR>…</GJAHR>
        <MONAT>…</MONAT>
        <FEXNR>…</FEXNR>
        <FEXTY>…</FEXTY>
        <NOMEF>…</NOMEF>
        <WRBTR>…</WRBTR>
        <GSWRT>…</GSWRT>
        <PERVT>…</PERVT>
        <ANZLI>…</ANZLI>
        <VALMF>…</VALMF>
        <WAERS>…</WAERS>
    </ITEM>
    …
</MOVMESFEX>
```

Deverá ser apresentada ao utilizador uma janela *popup* para este indicar a localização e o nome do ficheiro a gravar no seu computador. Deverá ser pré-preenchido o nome do ficheiro que será:

MOVMESFEX_<DATA_ACTUAL>_<HORA_ACTUAL>.XML

De modo a centralizar os acessos às transações atuais (e eventuais futuras) desenvolvidas no âmbito deste módulo de gestão dos fornecedores externos deverá ser desenvolvido um *cockpit* que permita aceder a elas a partir de botões indicativos e ilustrativos. Estes botões deverão estar no ecrã de entrada do *cockpit*. Adicionalmente, deverá ser definido um terceiro botão que permitirá o acesso a um segundo ecrã onde o utilizador pode inserir manualmente registos na tabela ZTMM_MOVFEX_<Iniciais>. Como tal, este ecrã deverá possuir campos de entrada que permitam o preenchimento da informação presente na tabela e especificada acima. Deverão ser implementadas as funcionalidades de gravação dos dados, limpeza do conteúdo dos campos e demais que possam ser úteis.

Este cockpit será implementado através do Pool de Módulos **ZMM_GESTAO_FEX_<Iniciais>** e será acedido via a transação **ZMM_GFEX_<Iniciais>**.

4 - Requisitos de Segurança

N/A

5 - Diagrama de Processos

N/A

6 - Desenho da Arquitetura Aplicacional

N/A

7 - Identificação dos Interfaces

Para além das transações ZMM_FEX_<Iniciais>, ZMM_MOVMESFEX_<Iniciais> e ZMM_GFEX_<Iniciais> não será necessário desenvolver qualquer outra interface ou acesso.

8 - Identificação dos Perfis de Acesso

As transações a definir no âmbito deste pedido serão de acesso livre, sem quaisquer restrições nos perfis de acesso dos utilizadores da UCG.

9 - Necessidades de Migração de Dados

O processo de alimentação da tabela onde serão guardados os dados dos movimentos contabilísticos será definido e programado pela UCG através de carregamentos diários.

10 - Formação

N/A

11 - Documentos Anexos

N/A