An approach to variability management in enterprise architecture

Rurua, N.

Award date:
2015

Department of Mathematics and Computer Science
Department of Industrial Engineering & Innovation Sciences

An Approach to Variability Management in Enterprise Architecture

Nino Rurua

In partial fulfillment of the requirements for the degree of
Master of Science in Business Information Systems.

Supervisors: dr. ir. H. Eshuis (TU/e)
dr. M. Razavian (TU/e)
dr. D. Fahland (TU/e)
Alphons Spierings (Philips)

Eindhoven, October 2015

ABSTRACT

Increasing tendency of organizations operating on an international scale results in higher variation of company's business models and operations. Royal Philips N.V is one of the multi-national companies facing the challenge of managing differences in the processes caused by country-specific fiscal regulations. The differences in regulations drive variability not only in the process design, but also in supporting application and technology architecture. Variability management on the architecture design level has been extensively discussed in the Software Product Line Engineering and Business Process Management fields. However, only limited research has been done on handling variability in enterprise architecture. The goal of the research paper is to address the challenges caused by variation in the requirements that impact architecture design of the enterprise. In this paper, we propose an approach to identify, document and model variability in a structured way and suggest an extension to enhance the core enterprise architecture metamodel with variability management concepts.

Keywords: enterprise architecture, variability management, variability modeling.

ACKNOWLEDGEMENTS

The report represents the result of my graduation project which completes my Master's degree in Business Information Systems at Eindhoven University of Technology (TU/e). I would like to express my gratitude towards the people who supported me throughout the project execution.

First of all, I would like to thank my supervisor dr. ir. H. Eshuis for his continuous support, constructive feedback and guidance from the very beginning to the end of the project. His dedication and professionalism, as a professor and as a supervisor, has always inspired me.

I would also like to thank my second and third supervisors dr. M. Razavian and dr. D. Fahland for their interesting insights and recommendations about the research process and methodology.

Next, I am thankful to my Philips supervisors: Massimo D' Attoma, who gave me the opportunity to carry out the project at the IT Architecture, Strategy and Performance Department at Philips, and provided valuable feedback on my research and presentation skills; and, Alphons Spierings, who has been continuously supportive and extremely helpful throughout my internship. His remarkable structured way of thinking and expertise in the field was always motivating. Working with him was a truly great experience for me.

I would also like to thank my friends who were there for encouragement and suggestions during my master program.

Finally, I am thankful to my parents for their care and support, and to my sister who has been the most patient, encouraging and helpful person in my life, and especially, during these last few months.

Nino Rurua

October 2015

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ADM	Architecture Development Method
AE	Application Engineering
APQC	American Productivity & Quality Center
B2B	Business-to-Business
B2C	Business-to-Customer
B2G	Business-to-Government
BPM	Business Process Management
BPMN	Business Process Model and Notation
C-EPC	Configurable Event-driven Process Chains
DE	Domain Engineering
EA	Enterprise Architecture
EDI	Electronic Data Interchange
E-invoicing	Electronic Invoicing
ERP	Enterprise Resource Planning
FODA	Feature-Oriented Domain Analysis
IS	Information Systems
ISO	International Organization for Standardization
IT	Information Technology
OVM	Orthogonal Variability Model
PESOA	Process Family Engineering in Service-Oriented Architecture
Provop	Process Variants by Options
SAP	Systems, Applications, Products – German Software Company
SP	Service Provider
SPLE	Software Product Line Engineering
TOGAF	The Open Group Architecture Framework
UML	Unified Modeling Language

1. INTRODUCTION

Today's dynamic business environment brings both benefits and challenges to organizations. Technologies enable easier access to new markets and resources, but internationalization also adds complexity to business processes of multi-national companies. New industry standards, government regulations and compliance requirements cause differences in data, processes or software systems used within organizations. However, identifying and handling such variations enables more effective support to organizational processes in the earlier stages of information systems development. Thus, the variability causes and consequences have to be taken into account during the design and modeling of enterprise architecture.

One of the new developments in fiscal legislations worldwide is the adoption of electronic form of invoicing. Digitalization of invoices and billing documents helps companies save costs by providing more control and automation of the process, and by eliminating the inefficiency caused by manual processing of paper-based invoices [1]. However, switching to new ways of invoicing is related to significant changes in the organizational processes, and requires alignment with not only internal systems but also third-party solution providers.

The trend of mandating electronic invoicing (e-invoicing) and fiscal reporting has been increasing in the countries in South America in the past few years [2]. Starting with the government of Brazil, the procedure has been regulated in Argentina, Mexico, and Chile amongst others. Even though the intent of having an effective means for tax collection is the same for each authority, the requirements still differ in multiple aspects that affect the generic process flow or e-invoicing [2].

E-invoicing can benefit both parties: governments in terms of a better tax compliance control and businesses in terms of an automated process handling. However, large international companies with existing legacy systems face challenges in finding and implementing efficient solutions because of the strict time pressure from the tax authorities [1].

1.1 MOTIVATION

The influence of recent developments in business environment is reflected in the way organizations operate today. Namely, the increasing tendency of outsourcing and globalization results in higher complexity of business transactions, growing regulatory framework forces companies to fully comply with regulations and standards, and growing dependency on

information technologies requires full understanding of the latest technology solutions as well as technology-related risks.

Royal Philips N.V is one of the multi-national companies facing the challenge of incorporating fiscal regulations in its processes. Operating worldwide, Philips provides products and solutions in the areas of Healthcare, Consumer Lifestyle and Lighting. In order to stay competitive and keep up with dynamic global markets, the company has been going through transformation to "seek local relevance while leveraging global scale" for the past few years [3]. One of the main objectives of this organizational transformation is to optimize business processes by standardizing them at the global level, while enabling local innovation and differentiation at the market level.

The variation in the processes also increases due to the constantly changing legislations on tax compliance. One of the processes affected by fiscal regulations is e-invoicing: generating billing documents, transmitting to the customer and reflecting the changes in the accounting and financial documents. High localization and lack of homogeneity in adoption of e-invoicing standards increase the number of process variants. This makes it more challenging to dynamically manage the variations and to maintain stable process models for different countries and organizational units. Complex requirements also influence the choice of IT solutions. In order to reap the benefits of electronic invoicing, Philips needs a well-defined approach to manage variations in its business and IT landscape.

The company uses enterprise architecture (EA) as an instrument to drive organizational changes and to keep its business and IT aligned (detailed definition of the term is given in Chapter 2). Enterprise architecture with well-documented business processes brings value to Philips in multiple ways. Firstly, the models represent a formal way of describing the company's strategic goals and operations, which helps the management to establish a clear vision of achieving each business objective through a consistent way of working. Secondly, enterprise architecture enables more effective selection and implementation of IT solutions.

One way to address the challenges caused by variation is through managing variability in the artifacts of the enterprise architecture. Capturing knowledge about existing variations in the enterprise allows companies to identify common issues and to develop reusable solutions which address similar cases. Some of the popular approaches of variability management developed for Software Product Line Engineering and Business Process Management use the concepts of variation points or configurable regions to locate variable parts in the models, and dependencies to show the linkages between variants [4] [5] [6] [7] [8]. A structured documentation of variability in terms of the concepts can improve management and traceability of variation between architecture artifacts [6].

1.2 Problem Statement

Variability management covers activities such as representation, modeling and resolving differences in given artifacts. The challenge with variability management for the company case discussed above is related to the initial stages of representation and modeling of variation which starts from the requirements affecting e-invoicing process. One of the root causes of the problem is that the information is usually kept in the minds of experts and is rarely documented explicitly. The absence of structured information and clear guidelines leads to inefficiency in the process modeling, resulting in higher maintenance efforts. In addition to that, representing variability in enterprise architecture also requires explicit links between variants across different layers of architecture. Poor visibility of options and alternatives of artifacts and lack of dependencies between them in the architecture design also complicates decision-making for domain architects and process owners.

1.3 Research Objective

Variability in the organizations can be handled by exploiting commonalities and implementing differences of various products or artifacts [9]. Variability management has been extensively studied in the literature in the Software Product Line Engineering (SPLE) [10] and Business Process Management (BPM) domains [8] [11]. Variations in SPLE have been addressed in terms of features, assets or decisions, and in BPM the research has been mainly focused on creating configurable processes, or reusable software components [12]. Some recent studies address variability handling in service-oriented architecture, but there is little research conducted on how variability can be handled in enterprise systems [13]. Artifacts in enterprise architecture go through more complex lifecycle supporting multiple business goals and involving multiple stakeholders than, for example, in product line projects.

The research objective of the study is to improve variability management and modeling in enterprise architecture based on the case of electronic invoicing process at Philips. The case was selected because of the challenges related to capturing and modeling the differences in the fiscal requirements of several countries. Fiscal requirements are categorized as a subgroup of compliance requirements at Philips. The compliance requirements can be related to quality, security, legal and other internal or international standards and controls the company adheres to, which are considered during the process design. For this reason, we use the term *compliance* requirements interchangeably with *fiscal* requirements throughout the report.

Considering available techniques for variability management in literature, our goal is to develop an approach that enables analysis of variability location, rationale and representation in the enterprise architecture.

The scope of the project covers identification, documentation and modeling of variable artifacts in enterprise architecture; modeling of the instances of executable variants are addressed during implementation/run-time of IS development and are not discussed in this report.

The main research question addressed in the paper is ***how to manage variability caused by differences in requirements in enterprise architecture?***

To understand the problem, to analyze its cause and to identify a feasible solution, the following questions will be investigated in more details:

RQ1. What are the techniques for variability management in enterprise architecture in literature?

RQ2. What are the current challenges for managing variability in enterprise architecture at Philips?

RQ3. How can variability management be supported by the enterprise architecture metamodel?

RQ4. How do differences in compliance requirements drive variation in the e-invoicing case?

RQ5. Is the defined approach for variability management feasible in the context of enterprise architecture and applicable to other variability cases?

Given the case study on electronic invoicing process, we analyze and model the effect of variation in the artifacts in architecture. In order to show how variability management can be supported by the enterprise architecture developed at Philips, we use a metamodel of the EA. The metamodel gives a high-level overview of the architecture building blocks, variability concepts and relationships between them across enterprise architecture layers.

1.4 Research Methodology

The project development follows the Design Science [14] approach which proposes a systematic way to build an artifact in order to accomplish effective analysis and the design of information systems. The methodology for the research process is the Information Systems Research Framework, originally proposed by Hevner et al. [14]. The design science approach was selected because of its relevance to the domain of information systems (IS) and the applicability of the approach to the problem. What differentiates design science from other disciplines, such as behavioral science, is that the research goal is to develop a new artifact rather than explain or justify existing phenomena. Design consists of both a process and a product intended to solve the organizational problem [14].

Following the design science approach, the *problem* addressed in this project (Section 1.2) is expressed by the means of the following artifacts: *constructs*, such as variation points, dependencies and variation drivers, *models* – EA metamodel, business and application architectures, and finally, the feasibility of the proposed method is demonstrated through evaluation sessions with the experts**.** The artifact produced as a result of this project is an approach to provide guidance on how to capture and manage variability caused by differences in the compliance requirements in enterprise architecture.

The research process follows an iterative cycle of developing and evaluating the artifact. The complete framework of the IS research is given in Figure 1.

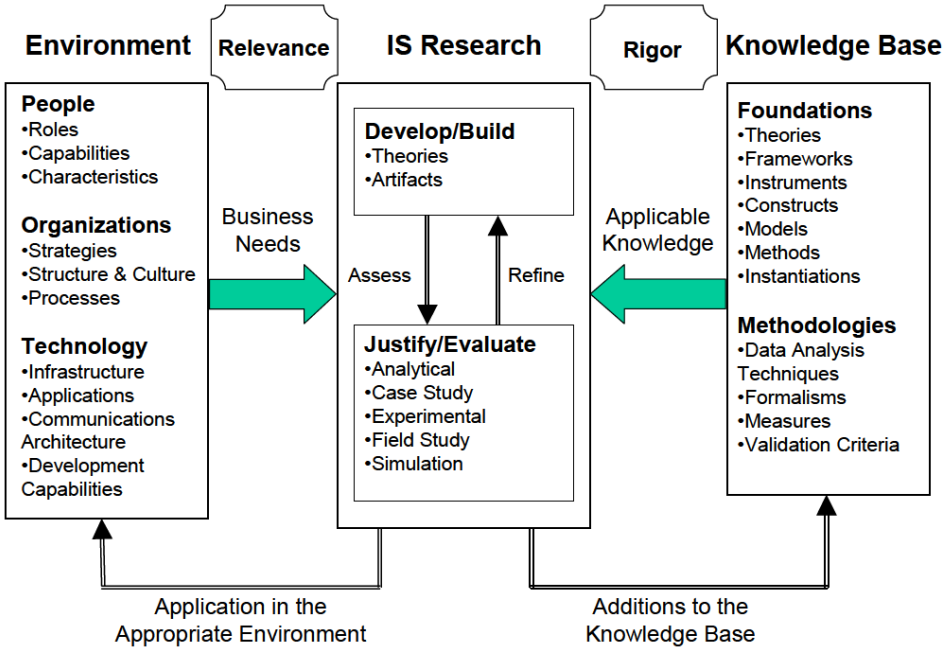


FIGURE 1. INFORMATION SYSTEMS RESEARCH FRAMEWORK [14]

There are two main sources of information used during the research. The first is elicited from the company environment, while the second one is based on the knowledge of scientific theories, frameworks and methodologies from literature.

1.4.1. ENVIRONMENT

The problem investigated in this project takes origin from the company environment. In the IS field, the environment consists of people, organizations and technologies. Using these resources of Philips and the area of interest, we identify the company goals, business needs and related problems in the area of interest, which is enterprise architecture. The problem is considered within the context of company capabilities, structure and technology infrastructure.

The project is carried out in the department of IT Architecture, Strategy and Performance. The main function of the department is to support Philips transformation by defining and designing the enterprise architecture of the company processes and IT systems. Identifying variability in the compliance requirements and its impact on enterprise architecture layers are in the interests of business and IT architects, as well as process experts at the company.

In order to identify the needs of the stakeholders, we have conducted interviews and discussions with them. We also got familiar with the tools (ARIS), modeling languages (BPMN, ArchiMate) and methodologies/frameworks (TOGAF, APQC) used for building enterprise architecture to make sure that the proposed solution is applicable and realizable in the current environment.

1.4.2. Knowledge Base

In addition to the organizational resources, we researched relevant methods, frameworks and theories to explore variability in the scientific literature. We used Kitchenham's [15] guidelines to conduct literature study on existing variability management techniques. The materials helped us define which of the available approaches could resolve the issues the company faces.

We have also studied established methods about how carry out a case study for a research problem [16], how to elicit and document requirements [6], how to document architecture-related design solution [17] [18], and how to evaluate the proposed approach [19].

1.4.3. Research and Design

The research and design process can be divided into three major activities: to *investigate and analyze* existing problem, environment and available resources, to *Develop/Build* the possible solution and to *Justify/Evaluate* the relevance and applicability of the proposed solution. This approach is illustrated in Figure 2, the square rectangles represent activities, the rounded rectangles show the respective resources and knowledge base used during the process, and the document-shaped forms include information about the produced knowledge and artifacts.

The *investigation* phase aims to study an existing issue in information systems (RQ1). At this stage, we study scientific literature to identify a gap in the area of interest – enterprise architecture. After identifying the lack of research on variability management, we investigate available techniques in related fields to find the appropriate solution for enterprise architecture. During the initial phase, we also study the company environment and methodologies, and communicate the research subject with the stakeholders to find a feasible case for exploratory purposes.

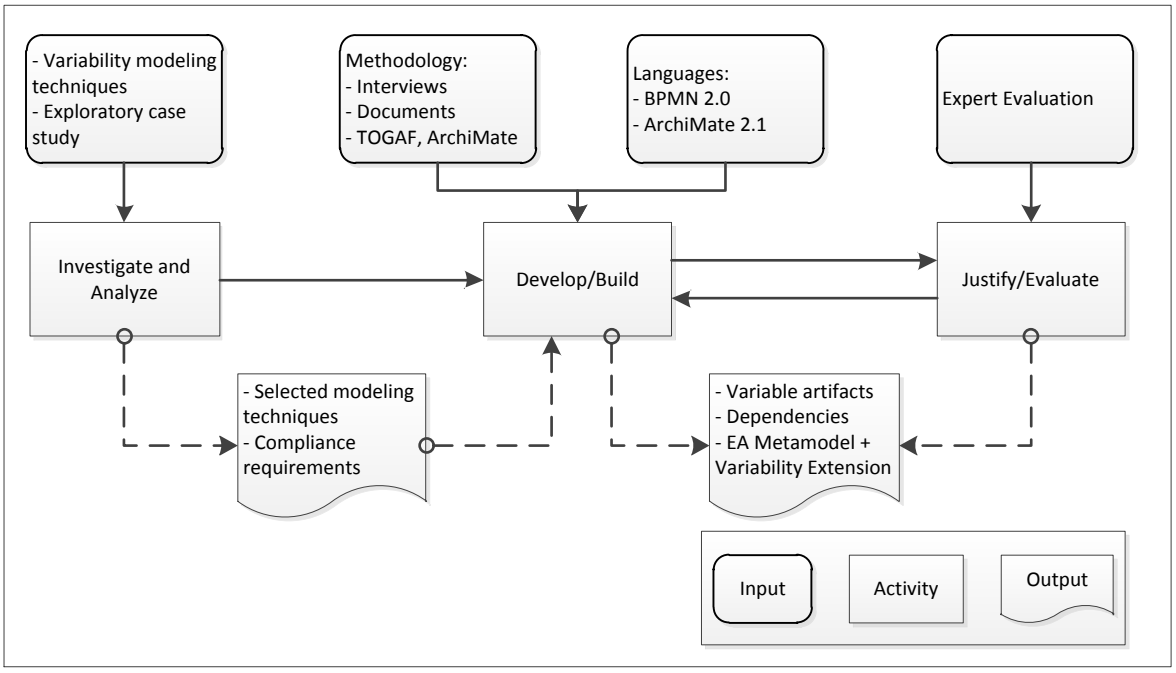


FIGURE 2. SOLUTION DESIGN METHODOLOGY

During the *Analysis*, we study and analyze the problem in detail (RQ2). More specifically, we study the current company approach and the challenges related to handling variability in the design of architecture.

The company case is used to identify and explore variation in the requirements that affect decision during process design (RQ3). In addition to that, we derive solution design by defining variability-related concepts, such as variation drivers, variation points and dependencies.

Structured documentation of variability helps to distinguish impacted components in the enterprise architecture and to represent them on the EA metamodel currently used at Philips (RQ4). One of the important aspects of design science is to ensure that the proposed artifact has utility in the given context. In order to achieve this, Hevner et al. suggest that the research has to show that the artifact works for the stakeholder community that's going to use it.

Having this in mind, we regularly communicate the design process with the stakeholders. The solution is assessed and refined during its development with the field experts, IT and business architects. The design is an iterative process and follows two activities of generating design alternatives and testing them against requirements, as presented in Figure 3.

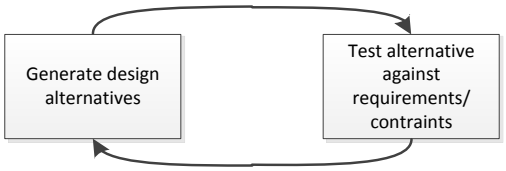


FIGURE 3. SOLUTION GENERATE/TEST CYCLE [14]

Eventually, we evaluate the solution using expert opinions (RQ5). During the *Evaluation* phase, we check feasibility of the solution for managing the differences in the compliance requirements during enterprise architecture modeling. Since our research area addresses a particular organizational problem, the solution is evaluated with respect to its utility within the practical context [14]. The utility of the proposed approach is assessed based on its relevance and applicability to stakeholder goals which were identified during the company case study.

1.5 REPORT OUTLINE

The report outline follows the research process conducted during the project execution. Figure 4 shows the activities carried out during the research with corresponding research questions in the left column, and the right column shows breakdown of the project into report chapters.

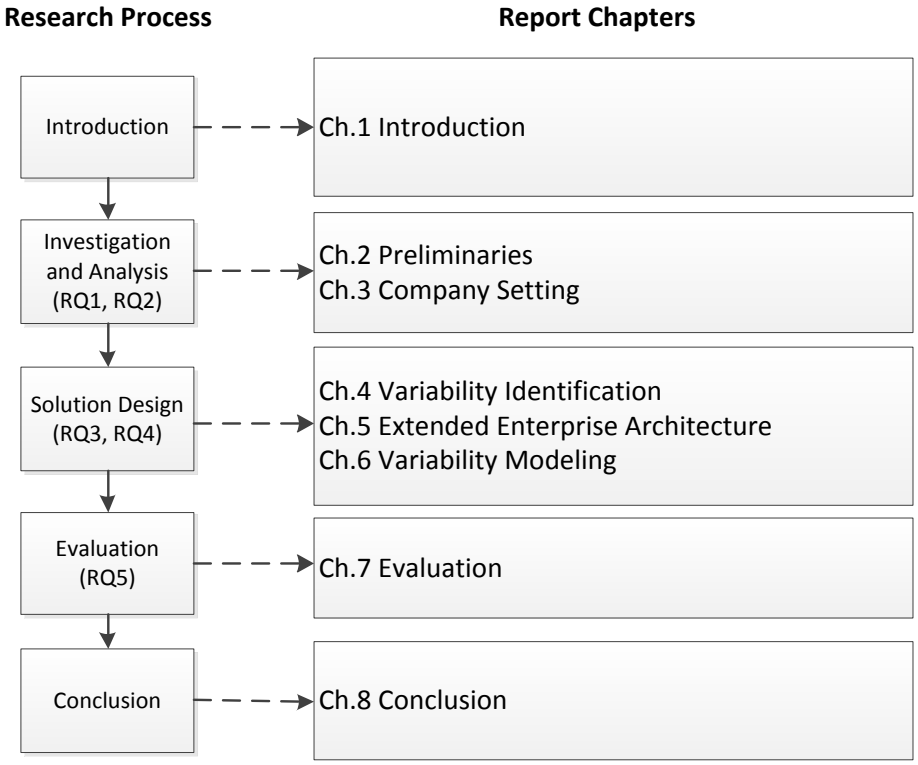


FIGURE 4. REPORT OUTLINE

Firstly, we introduce and define the problem area (Ch.1). The introductory part also includes preliminaries to describe enterprise architecture methodology used during the research, and the findings of the literature study on variability management techniques (Ch.2). Ch.3 describes the case study conducted to analyze the company's current approach for variability managements, the company's current approach to modeling processes and concerns related to variability modeling. This section is followed by the chapters describing the solution design (Ch.4, 5, 6,) and the evaluation of the approach using expert evaluation overview (Ch.7). The report is concluded with discussion of feasibility of the proposed solution, the limitations and ideas for future work (Ch.8).

2. PRELIMINARIES

This chapter describes the main concepts related to the presented research topic. Some of the terms used in the report are relatively new and have more than one possible definition in the literature. In order to avoid ambiguities related to the meaning or context of terms like *enterprise architecture* and *variability management,* we define the most relevant ones below. In addition to that, the chapter presents a summary of the literature analysis conducted to investigate current approaches of variability management in software and enterprise architecture.

2.1 ENTERPRISE ARCHITECTURE

Enterprise architecture (EA) is one of the core concepts discussed in the project. Even though the term is frequently used nowadays, there is no one standard definition of it in the literature. EA design frameworks may range from being specific to a particular domain to being too generic and applicable to different types of organizations. EA design may also focus on structuring models and artifacts in complex environment [20] or on describing a method for organizational change [21]. There are also different ways of developing guiding enterprise architecture development in the organizations. One of the most common approaches for adopting enterprise architecture is the TOGAF methodology. Since Philips also follows TOGAF to go through the organizational change, we describe the methodology in Section 2.1.2.

2.1.1. EA DEFINITION

One of the first works by Zachman [22] suggests a framework of information systems architecture. Even though at the time the work was published, the term "enterprise architecture" was not in use, his framework describes an organizational structure from various stakeholder perspectives and provides support for business, information systems (IS) and information technology (IT) alignment.

The Zachman Framework represents ontology for information systems architecture and not a methodology for strategic planning [22]. In other words, there is no guidance on the sequence or implementation of enterprise architecture. The framework focuses on capturing all the critical models of the enterprise by describing the scope, business model, system model, technology model, components and working system with respect to the six questions: what, how, where,

who, when and why. The latest version from 2011 of Zachman Framework for Enterprise Architecture is given in Appendix A.

Enterprise architecture may also be considered not only as a structural framework, but as a method for ensuring company-wide integration of business and IT. According to the definition of EA suggested by Lankhorst [23], it is necessary to have both architecture models and a respective tool with well-defined modeling language to support integration of different models in EA. EA defined by Lankhorst is as follows:

> Enterprise architecture: a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems, and infrastructure [23].

The definition of "architecture" accepted by the ISO (International Organization for Standardization) is in line with the latter approach. According to ISO/IEC 42010: 2007 terminology, an architecture framework also includes the "principles governing its [architecture's] design and evolution" [21]. The Open Group Architecture Framework (TOGAF) partially adheres to ISO principles, and provides two definitions of the "architecture" depending on the context [21]:

> 1. A formal description of a system, or a detailed plan of the system at component level to guide its implementation.
>
> 2. The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.

Since Philips use enterprise architecture as a means to achieve lean and structured transformation of the organization and follows TOGAF 9.1 version, EA will be referred to not only as an ontological framework, but also as a methodology to derive reorganization to a new structure in the report.

2.1.2. EA Methodology

Building enterprise architecture is a complex process which involves collecting and organizing information based on the input from various stakeholders to create models, roadmaps or viewpoints in order to achieve organization's current and future goals. TOGAF represents a structured methodology to support systematic development of related artifacts, their storage and reuse.

Philips has selected TOGAF to ensure more efficient business operations by clarifying the strategic context and business capabilities of the organization. Using EA, the company also aims to achieve effective realization of its business goals by better alignment of business and IT through architecture.

The Architecture Development Method (ADM), which is the core concept of TOGAF, provides a method for developing and using the enterprise architecture. The ADM covers the following layers: Business, Information Systems (which includes Data and Application layers) and Technology layers. The ADM represents an iterative cycle of the design and execution across the different phases. The complete development cycle of enterprise architecture by TOGAF is given in Figure 5. The figure also shows mapping of modeling languages supporting each layer of EA. Currently, Philips uses ArchiMate (2.1) notation to design the IS and Technology layers, and BPMN (2.0) to model processes in Business architecture layer. More details about each element TOGAF 9.1, ArchiMate 2.1 and BPMN 2.0 are given in Appendix B.

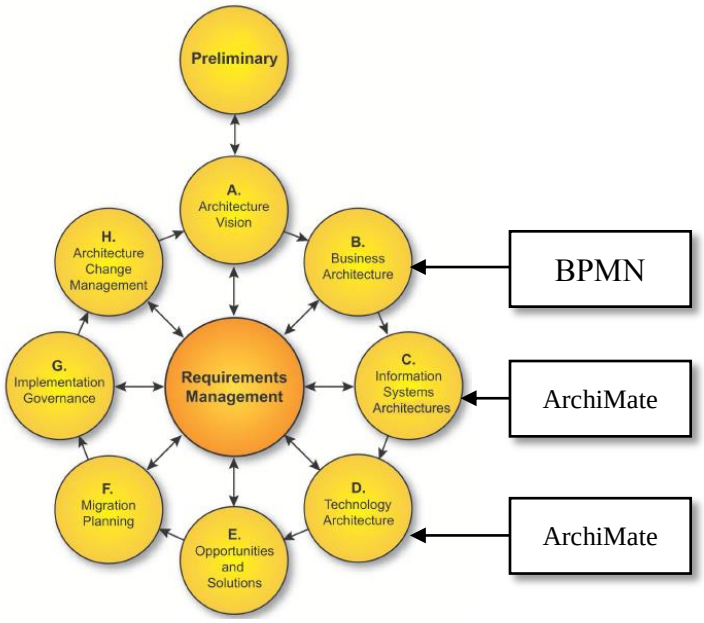


FIGURE 5. THE ARCHITECTURE DEVELOPMENT METHOD (TOGAF 9.1)

In TOGAF the modeling language that supports architecture design is ArchiMate. ArchiMate has been developed to provide a uniform representation for diagrams that describes all layers of the enterprise architecture [18]. One of the major advantages of ArchiMate is that it provides notations for relationships between concepts on different layers. These relationships make it easier to show and trace dependencies between objects across enterprise architecture. All the relevant concepts and relationships are organized in an enterprise architecture *metamodel* which defines the underlying language of a framework or methodology.

An architecture metamodel can be used to describe concepts at different levels of specialization. Figure 1 illustrates three levels of specificity in a metamodel: the bottom layer describing more specialized concepts and moving up to the top layer covering more general terms [18]. The enterprise architecture language, defined by the ArchiMate metamodel, is designed to strike a

balance in between. It provides specialized layers for different domains but is also generic enough to be used for most enterprise architecture modeling tasks.

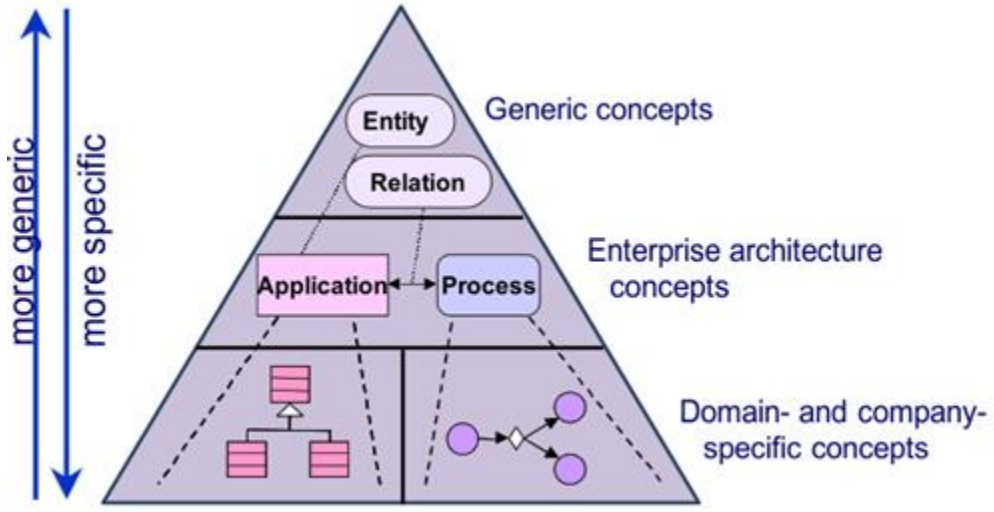


FIGURE 6. METAMODEL AT DIFFERENT LEVELS OF SPECIFICITY

In addition to the three basic layers of enterprise architecture defined by TOGAF, the Philips EA metamodel is also extended with a *Motivation* layer for its strategy development. The purpose of extending the core content concepts and relationships of a given language notation is to support domain, application or generally, company specific needs for analysis and visualization The layer includes motivational elements driving the organization's architecture design. The motivation extension is supported by ArchiMate 2.1 version and includes motivational concepts, such as intentions (*goals, principles, requirements, constraints, etc.)* and sources of these intentions (*stakeholders, drivers and assessments).*

The Motivation extension is a built-in mechanism in ArchiMate, which allows customizing metamodel without actually changing it [24]. Similarly, a built-in mechanism is UML profiling using stereotypes. Profiling is a technique developed to enhance existing metamodel with constructs that are specific to a particular domain, platform, or method [25].

2.2 VARIABILITY MANAGEMENT

One of the goals of having enterprise architecture in place in an organization is to manage the complexity of enterprise-wide systems with respect to business goals. The complexity in design

and architecture of enterprise systems is often due to variation in the organization's business requirements, products or services, as well as implementation options. The problem of handling variations in the information or software systems artifacts is common in information systems field and is often referred to as *variability management* in literature [7] [12].

Several studies [12] [26] [27] [28] differentiate between handling *variability* and *adaptability or modifiability*. They are all referred to as quality attributes of architecture design [29], but while *adaptability* or *modifiability* address unexpected changes or new circumstances to be accounted for during the design, *variability* is considered to be caused by known differences in the artifacts.

2.2.1. VARIABILITY CONCEPTS

Variability conveys and specifies information about differences and commonalities between parameters, features or choices about information systems artifacts [30]. Capturing variability in the models allows representing possible variations and guides deriving an individualized model based on given requirements. The earliest works address variability in features, originating from product lines (families) [31] [32] and software product line engineering (SPLE) [12]. Variability in SPLE is defined as an ability of a system or an artifact to be extended, changed, customized or configured depending on the specific context [10]. It refers to the existing or expected differences in the product or software artifacts and thus, can be taken into account during design or run-time phase of product/software development. Similar approaches have been adopted to deal with large process model repositories of related process variants in business process management field [33].

Variability in information and software systems is managed over the full lifecycle of the artifact development (for example, in SPLE, during *Requirements, Domain and Application engineering* phases [6]). The lifecycle of variability management can be divided in two main phases [13]: *description* or *specification,* which includes identifying, modeling and constraining variability, and *resolution,* which covers instantiation and implementation stages.

The works in literature focus on different stages of variability management; part of the studies deal with design and modeling techniques [34] [31] [8] [35], another part with resolution and execution ways [9] [30], while others cover full lifecycle of variability handling [7] [11] [36] [13]. Some of the differences in classifying the stages of variability management from the literature are presented in Table 5, Appendix C.

In SPLE, several works map the development lifecycle stages to the realization abstraction levels of the artifact [6] [10] [13]. The variability abstraction levels in SPLE are presented in Figure 7, in the first and second columns from the left. The third column shows corresponding layers of enterprise architecture as defined by TOGAF and ArchiMate.

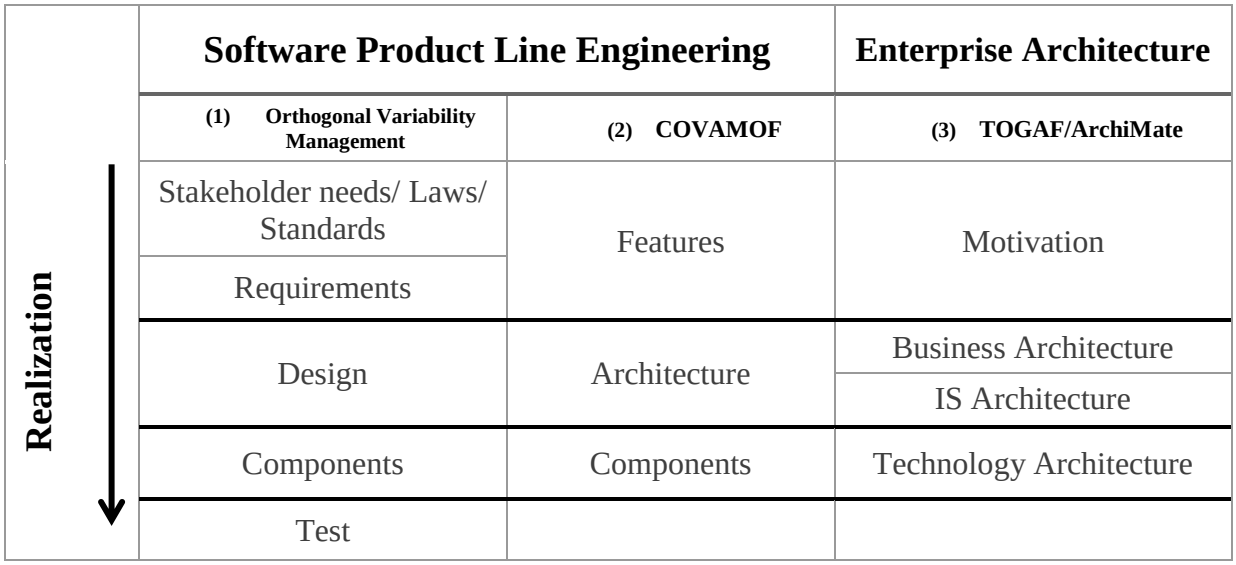


FIGURE 7. REALIZATION ABSTRACTION LAYERS.

The initial steps in variability management are to elicit and identify the causes of variation, as well as differences and commonalities between possible variants. The later stages of variability management support variability realization and implementation during runtime. The scope of the project, however, is limited to the first stages of *identifying* and *modeling* variability.

Understanding and explicitly documenting variability improves decision-making, communication between stakeholders and traceability between variation causes and effects. In order to adequately describe and document variability, Pohl et al. [6] suggest answering four questions about software families. These questions relate to the common terms used to express variability in literature [5] [7] [9] [33]. However, the list proposed by Pohl et al. [6] does not include a question regarding dependencies or relationships between variation points and variants. These elements play an important role in allowing traceability of variation [7] [37] in SPLE and processes, and they also represent one of the biggest concerns for enterprise architecture. So, in order to better understand linkages between variable artifacts across different layers, we also included a question "How can it [variable artifacts] be traced?" in the list:

What does vary?

A variability subject is often referred to as a *variation point*, meaning the location, time or the "point" in a generic artifact where variation may occur. In the business process models, variation points are also called *adjustment points* [11] or *configurable regions.* Variation points are introduced to provide alternative implementations of functional or non-functional artifacts, including features, processes, etc.

One of the common ways to represent variability in software systems is to describe it in terms of variation points and dependencies between variation points and other elements [38] [7] [37] [36].

Variation points can be introduced at any abstraction level of architecture and can have impact on the design and execution of the elements at the lower level [6]. Identifying variation points helps in systematic documentation of variability, as well as provides support for forward- and backward-traceability by defining dependencies and relations between components.

Why does it vary?

A factor or a parameter causing variation in the processes or in system artifacts is described as a *variation driver.* Understanding variation drivers helps to elicit and conceptualize collections of variants (e.g., in the processes). Variation drivers are categorized into two main groups: *business variation drivers* and *syntactic drivers*. The first group refers to the any kind of business reason that produces separate process variants, and can be related to markets, product/services or operational part. And, the second group differentiates the variation in the execution of the process with similar outcomes.

How does it vary?

Artifacts or variability objects affected by differentiating factors are referred to as v*ariants* or *variation options. Variants* express the options or alternatives which resolve a variation point. In SPLE, variants are usually defined during the design phase, selected for a specific choice model and instantiated during run-time of an application. The variants can be *optional*, *mandatory* or *alternative* [39] with regards to resolving the given variation point.

How can it be traced?

The relationships between variation points, variants and other objects across layers are managed using *variability dependencies* and *variability constraints.* These elements express the rationale behind the variation point - variant relationship, and allow traceability between possible choices.

Variability *constraints* restrict dependencies by imposing limitations on variation point resolution. There are two types of variability constraints that influence relationships between artifacts: in one case selecting an artifact *requires* selection of another artifact to be functional, or the opposite, it *excludes* certain options. A constraint describes a relationship between variation points, between a variation point and a variant or between variants [6].

La Rosa et al. [30] treat dependencies as *order dependencies* in their work on questionnaire-based variability modeling for system configuration. *Order dependencies* capture restrictions on the sequence of variation point/variant selection, which can be *partially* or *fully dependent* on the other. However, the sequence is relevant to configuration or instantiation process, while our focus is on capturing and representing variability in the design.

For who is it documented?

Pohl et al. [6] distinguish *external* and *internal* variability based on whether the visibility of the documentation is intended for a customer (an external stakeholder), or if it's intended for developers and engineers within the organization. Thus, the v*isibility of variability* can also be interpreted as the view on variability from a particular stakeholder perspective. *External view* is usually useful when a customer is involved in the product design and selection of features, but in the case of enterprise architecture, the view serves internal purposes for business and IT landscape design.

2.2.2. VARIABILITY MANAGEMENT TECHNIQUES

Variability management has been mainly researched in two areas in information systems field, in software product line engineering [7] [9] [10] and in business process management [8] [30] [11] [12]. While in SPLE approaches mostly deal with modeling features, their combinations or implementation ways, in BPM variability management addresses configurable parts of the process and possible variant instantiations.

Variability modeling in SPLE is usually addressed during domain engineering (DE). Domain engineering is one of the two main processes distinguished in SPLE and includes domain analysis, domain design, domain realization and domain testing phases [6]. Domain knowledge is used to identify and model common and variable parts of software products, and to represent them in a form of architecture. The reusable assets defined during DE are used as an input for application engineering (AE), which is the other process in SPLE. While DE is mainly concerned with identification, customization and design of variability, the instantiation and realization of variability happens during AE process (See Table 5, Appendix C).

Sinnema et al. [10] study modeling techniques applied during domain design phase of SPL development. *Modeling* concerns the ways in which the information about variability is presented in terms of choices, abstractions, formal constraints or as a support for incompleteness or imprecision. The authors identify similarities and differences between six common modeling approaches. The majority of the selected techniques (ConIPF, CBFM, Koalish, Pure::Variants) use feature diagrams and hierarchical structures as main principle behind modeling, so we will only discuss the original FODA (Feature-Oriented Domain Analysis) technique further.

A smaller number of variability modeling techniques in SPLE provide support for decision models (Variability Specification Language (VSL) and COVAMOF). Decision models are also referred as configurable models in the literature [30], which will be discussed in relation to business process modeling later in this section. The complete table classifying the variability modeling techniques is given in Figure 26, Appendix C.

Variability in SPLE is also investigated from realization and implementation perspective [9] [4]. Svahnberg et al. study qualities of variability in software product families and provide detailed taxonomy of variability realization techniques [9]. Based on their earlier works, Svahnberg et al. claim that one of the main drivers behind introducing variability in software development is the possibility to delay design decisions. Since the variability realization happens during application engineering phase and thus, is not within the scope of this report, the techniques will not be analyzed further (The list of all the techniques with corresponding descriptions from is given Figure 27, Appendix C).

Chen and Babar [40] conduct a systematic review of general variability management techniques in literature. Their research identifies 91 different variability management approaches from different aspects and during different development phases. But most of the studied literature deals with variability in SPLE, and in particular, with modeling phase of variability. The approaches are categorized into 13 classes, where two largest groups address variability modeling in terms of features (33 studies) and UML-based techniques (25 studies). Eight studies fall in the third largest group where variability is used as part of a technique to model architecture of the system.

Overall, there can be two major categories distinguished in variability management techniques from the modeling perspective. The first group treats common and variable parts as a single model, while the second one manages variability in a separate model. The first group is referred to as a *single-artifact (model)* variability modeling technique, while the latter is called a *multi-artifact* technique [33].

SINGLE ARTIFACT APPROACH

Techniques using single model approach represent variability by enriching the original model with information on variable artifacts [33]. Single artifact approach is common in business process modeling, expressed as conditional branching, configurable nodes, annotations of BPMN, labels for EPC or as extensions of UML [11] [33].

Configurable Models

Configurable models are designed to represent a common part of the processes and related process variants as a single model. According on the business/customer-specific needs, a relevant business process is derived or configured. The combinations of configuration alternatives are limited by the requirements and constraints, identified during domain or application design. Configurable constraints can be represented as textual annotations on the model.

Configurable Event-driven Process Chains (C-EPCs) is one of the popular approaches using configurable nodes and textual annotations to embed variability in a single model [8]. The EPCs are directed graphs, consisting of events, functions and connectors and visualizing a control flow.

Rosemann et al. [8] extended the original EPCs with a formalized modeling language to capture both variability and commonality in a process model at build-time.

In C-EPC, configurable nodes are denoted by thick circles for configurable connectors and thick rectangles for configurable functions. Functions can be configured as (ON) enabled, (OFF) disabled and (OPT) stands for optional, meaning the function can be skipped. C-EPC also includes configuration requirements (constraints) and configuration guidelines, which bind configurable connectors and functions using sets of logical expressions. On the model, they are represented as tags to the associated configurable nodes.

Domain/application context conditions are presented in a separate *questionnaire model*. The questionnaire model is used to guide configuration process by presenting domain-specific questions to the experts with possible variants or *domain facts* to select from [30].

Stereotypes

Another example of single-model approach is using notations or stereotypes common in UML (Unified Modeling language) to express variations of an artifact in one model. In UML, a stereotype defines "how an existing metaclass may be extended, and enables the use of platform or domain specific terminology or notation in place of, or in addition to, the ones used for the extended metaclass" [25]. The stereotypes represent keywords between angle quotes (<< >>) and are used together with the object labels in order to distinguish between the subclasses of the main class or metamodel base class. Standard stereotypes in UML, such as <<Specification>> and <<Realization>> may apply to system components to model with distinct specification and realization definitions.

One of the common applications of stereotypes is used in PESOA (Process Family Engineering in Service-Oriented Architecture) project [41]. According to this approach, the stereotypes used to mark a place in the process model where variation takes place is <<VarPoint>> and the one used to capture possible alternatives <<Variant>>. In addition to that, PESOA defines more options for stereotypes, such as <<Null>> for optional activity, <<Default>> for default variant, etc. The PESOA approach has been integrated to handle variability for UML activities and BPMN tasks. In order to distinguish variation point resolution options (alternative or optional) in UML, Razavian and Khosravi [42] suggest embedding the indication to decision support inside the stereotypes, such as assigning <<alt_vp>> to variation points where decision has to be made between two alternative variants and <<opt_vp>> where the variants are optional.

Examples of single-artifact approaches modeled using C-EPCs and stereotypes (PESOA) are illustrated in Appendix D.

MULTI-ARTIFACT APPROACH

In multi-model approach variability is handled using several artifacts, such as feature diagrams, or Orthogonal Variability Models (OVM) to represent the variable part of system separately.

Feature models

Feature modeling is one of the common techniques to describe software product lines and to express variability using feature diagrams in SPLE. A feature is defined as "a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems" which can be used to capture commonalities and differences among software artifacts [34].

Kang et al. [34] present separate concepts of feature models and feature diagrams. Feature models provide description of the main areas of the domain using sets of feature values, while feature diagrams provide a way to visually depict relationships between given features using graphical and/or hierarchy structure. The feature diagram is represented as a tree with high-level features on top, which are decomposed into subfeatures to the leaf-nodes. The following relationships are allowed among features: *optional*, *mandatory* and *alternative* features. Figure 8 shows the basic notations used in feature models and a sample model, where *G*, *H*, and *I* represent alternative options for fulfilling *r*, and *r* itself is a mandatory component for *e*.

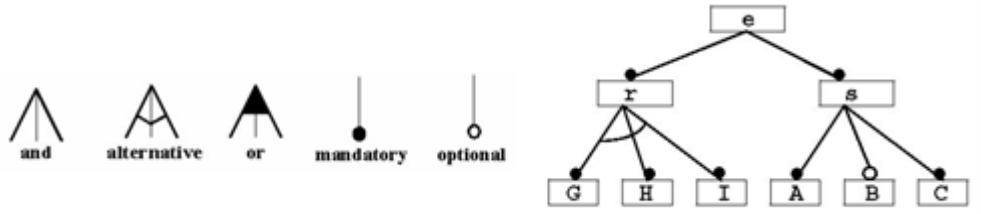


FIGURE 8. FEATURE DIAGRAM NOTATIONS AND AN EXAMPLE OF A FEATURE DIAGRAM

Additional information for features can be expressed in terms of range, frequency and binding times [34]. Binding time relates to the phase when the feature introduction has to impact the architecture. According to the FODA method, feature variability can be represented at *compile-time* for efficiency reason, at *load-time* by providing values before the start of execution, and at *run-time* when feature can be changed during execution interactively or automatically.

Feature diagrams provide clear overview of commonalities and differences of components or systems but lack the support for dependency relations. In addition to that, feature trees assume that variability-related elements can be expressed in a hierarchical structure, which is not always true in other fields (i.e. business process management).

Questionnaire-model

La Rosa et al. [30] propose a framework which represents variability using configuration models composed of questions and facts or possible answers. Configuration from the generic model is derived using domain-related questions and facts, which are selected upon variant customization.

The model also supports control over the order the questions appear by using dependencies over the facts. The suggested order of questions is based on the selected answers and is defined through the domain constraints. This way the consistency of the derived variant with the requirements and constraint is ensured from the beginning.

As opposed to C-EPC process models, the questionnaire-model approach captures the variability and commonality of system or software artifacts separately. In other words, the generic model only depicts the common or shared part of the process, while each variant configuration is presented as a separate model.

Orthogonal Variability Model

Similar to feature and questionnaire models, Orthogonal Variability Model (OVM) approach also separates variability concerns from a system model. Pohl et al. [6] suggest a cross-sectional view of variability from across different system artifacts into a separate model (Figure 9).

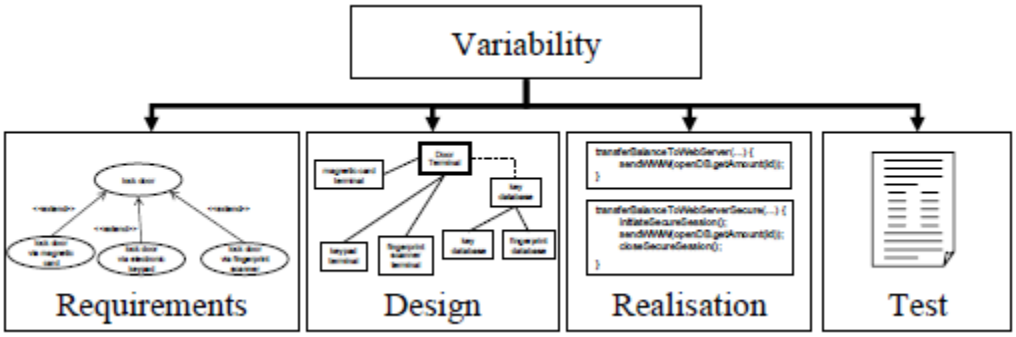


FIGURE 9. OVM APPROACH FOR SPLE

The authors also define special graphical notation for the OVM to express variability-related concepts, such as variation points (*optional*, *mandatory*, *alternative*), variants, dependencies (*requires*, *excludes*) and relationships between them. The complete metamodel of the graphical notation is given in Appendix E (a).

The captured information on the model is used to support variability management in various software artifacts, such as requirements, product models, etc. These software development phases are supported by different models, including use case diagrams, feature models, sequence or class diagrams. The special notation for variability is used to address the issue of traceability between variants spread across the different artifacts.

COVAMOF is another variability management framework which uniformly models the variability in all abstraction layers of software product family [38]. As in OVM, the approach

identifies two types of dependencies requires and *excludes*, but distinguishes five types of variation point, including:

- An *optional* variation point is the choice of selecting zero or one from the one or more associated variants.
- An *alternative* variation point is the choice between one of the one or more associated variants.
- An *optional variant* variation point is the selection (zero or more) from the one or more associated variants.
- A *variant* variation point is the selection (one or more) from the one or more associated variants.
- A *value variation* point is a value that can be chosen in a predefined range.

In addition to that, COVAMOF two views two manage variability, one to show variation point-variant relationships and the other to handle main dependency interactions. The authors [38] also suggest a specific graphical notation for modeling variability based on their previous project. The two development views and a general framework of COVAMOF are given in Figure 10.

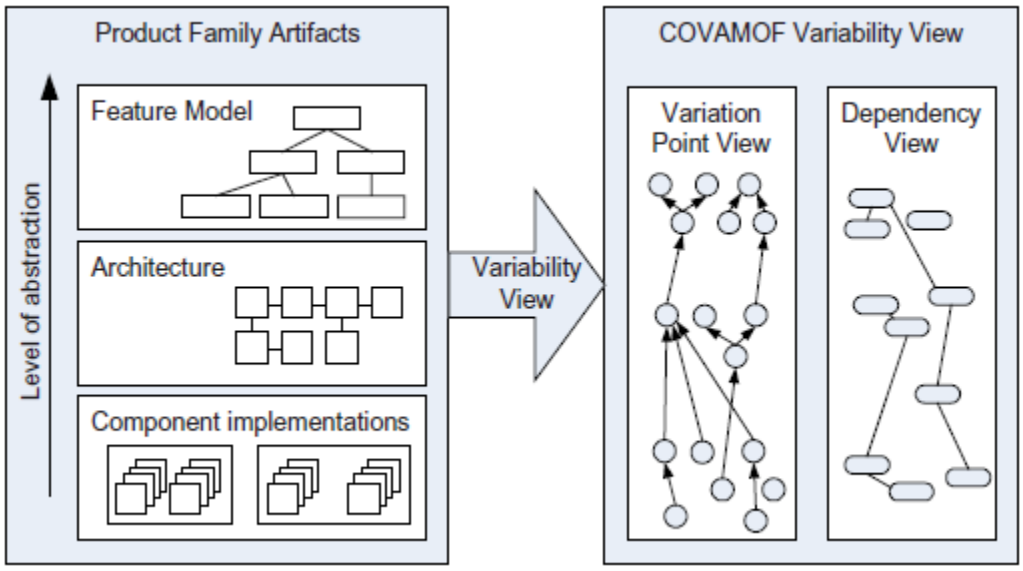


FIGURE 10. COVAMOF VARIABILITY VIEW

Provop

PROcess Variants by Options or Provop [11] is another alternative in modeling configurable processes. Provop differs from the previously described approaches in that it does not follow strictly single-model or multi-model structure, and also it support process variability during the process life cycle.

The business process life cycle as suggested by Hallerbach et al. [11] can be divided in three phases: *Process modeling, configuration* and *execution* phases. The base model with its variants or options grouped together to avoid unnecessary complexity are presented in the Provop model during the *modeling phase.* The base model is built using one of the suggested policies, depending whether it represents a standard process, the most frequently used process, or is one with the minimal average distance to other variant models, or represents superset or intersection of all process variants.

In order to derive configuration variants, the base model in Provop is adjusted using a set of high-level change operations (INSERT, DELETE, MOVE and MODIFY). In comparison with C-EPC or questionnaire-based methods, in Provop it is allowed to move or add model elements or adapt element attributed during a variant configuration. The general framework of the approach is given Appendix E: Multi-artifact Approaches (b).

The flexibility provided in the Provop approach in building the base model, is restricted by the relationships between options to some extent. The options are produced in the modeling phase are associated to context rules. The context rules bind a particular variant to its process family and other constraints, in order to assure correctness in the selection of the configurable variant.

2.3 Concluding Remarks

The benefits of effective variability handling using modeling can be two-fold. Firstly, variability modeling provides clear understanding of the domain and helps to resolve complexities caused by variations dependencies between them. Also, modeling variability using software artifacts provides better understanding of the variability and development of configurable models and architecture.

Variability management approaches can be classified as single-model or multi-model approach. The single-model approach requires less effort in terms of maintainability as it allows representation of common and variable parts in a single model. In addition to that, having all the elements and dependencies in the same model gives a better visibility of existing variations. On the other hand, the model may become too large and difficult to manage in case of large number of objects and complex dependencies [43].

By treating variability as a first-class citizen in the models, the multi-model approach is more effective in case of complex dependencies. Representing variable parts of the model as a separate artifact also allows better traceability between components and flexibility for change management [6].

Based on the selected literature studies, there were two main areas identified with variability modeling approaches. The earliest ways of variability management originates from product line engineering and is represented as a hierarchical tree structure of bill-of-materials. Similar approach using feature diagrams has been widely adopted in software product line engineering,

also initially based on hierarchical representation of feature variants, but later extended with additional attributes. Another major development on configurable models has been used extensively in business processes to resolve process complexities caused by variation.

Variability management is not as deeply explored in the area of enterprise architecture. Enterprise architecture combines multiple sources of information and serves the purpose of the whole organization, rather than a single product or a process. For this reason, artifacts in enterprise-scale systems go through more complex life cycle involving multiple stakeholders and business goals which create more complex dependencies between its layers. Having defined the enterprise architecture and studied the variability management techniques in literature, next we explore the company environment, current architecture design principles and the e-invoicing case study.

3. COMPANY SETTING

The company problem discussed in this project addresses the variability caused by differences in fiscal requirements of several countries. We use a case study approach to investigate the country-specific regulations. Since the goal is to manage variability by exploiting the commonalities and differences between them, we treat this group of countries as a single case. The case study includes Latin American countries (Argentina, Brazil, Chile, Colombia, Mexico, Peru, and Uruguay) and is explored to design analyze the problem and design the solution.

In this chapter, first we present the current enterprise architecture of the company, its metamodel and the way of managing variability, then describe the case study approach to investigate the requirements of electronic invoicing and related process design principles at the company.

3.1 ENTERPRISE ARCHITECTURE DESIGN (AS-IS)

The core content metamodel of enterprise architecture defined at Philips currently uses ArchiMate 2.1 [18] concepts and TOGAF methodology [21] as described in the previous chapter. The metamodel defined at Philips uses only a subset of concepts from ArchiMate. The metamodel presented in Figure 11 is adjusted to omit some of the elements and relationships of strategic importance to the company. The complete metamodel of the company represents their intellectual property and is used only for internal purposes. For the same reason several objects in the motivation layer are abstracted to general ArchiMate terms. However, this does not create a problem for understanding and depicting the relationship with the variability concepts added later to the core metamodel.

The core content metamodel contains four layers, *Motivation, Business, Application* and *Technology* layers. The model has a top-down structure showing the realization levels of business goals starting from the Motivation layer, followed by supporting business processes and corresponding application/technologies. The elements of the metamodel and the key relationships between them are briefly described below. An overview of complete ArchiMate set is given in Appendix B: Enterprise Architecture.

Motivation Layer

Motivation layer describes the highest level of organization goals. Using ArchiMate objects, such as the *goal, driver,* and *requirements*, this part shows the underlying motivation for design or change of enterprise architecture components. The motivation elements described below influence, guide and constrain the design choices of the lower level realization layers.

- **Goal:** A goal is defined as an end state that a stakeholder intends to achieve.
- **Driver:** A driver is defined as something that creates, motivates, and fuels the change in an organization.
- **Assessment:** An assessment is defined as the outcome of some analysis of some driver.
- **Requirement:** A requirement is defined as a statement of need that must be realized by a system.

Business Layer

Business layer illustrates the process framework and design of the company. The processes describe the behavior of the organization in the way it provides products or services to its customers. Following APQC process classification framework, the processes are categorized into seven levels according to granularity. The highest level L1 Category describes the area of the domain (i.e. Finance), L2 describes a Process Group (i.e. order-to-fulfillment), L3 – Process illustrate a standalone unique process (i.e. Manage Sales Order), L4 – Activity describe reusable pieces of flow in the processes, L5 – Tasks describe activities done by a single person at a time, L6 – Steps are even more granular breakdown of tasks and L7 – Work instructions provide guidelines to accomplish a particular process activity.

A process is executed by people or by software systems, which is illustrated using a *Business Role* element. Other related terms in business architecture are described below:

- **Business Role:** A business role is defined as the responsibility for performing specific behavior, to which an actor can be assigned.
- **Business Process:** A business process is defined as a behavior element that groups behavior based on an ordering of activities. It is intended to produce a defined set of products or business services.
- **Business Event:** A business event is defined as something that happens (internally or externally) and influences behavior.
- **Business objects (Input/output):** A business object is defined as a passive element that has relevance from a business perspective.

Application Layer

The application support of business processes is described using an Application layer. The main component of the layer is application component which can be used to model any structural entity, such as software application, sub-application or information systems. Application architecture also defines interfaces and information flow channels between different components. The rest of the elements are defined as follows:

- **Application Component:** An application component is defined as a modular, deployable, and replaceable part of a software system that encapsulates its behavior and data and exposes these through a set of interfaces.
- **Information Flow (Application Interface):** An application interface is defined as a point of access where an application service is made available to a user or another application component.
- **Application Function:** An application function is defined as a behavior element that groups automated behavior that can be performed by an application component.
- **Data Object:** A data object is defined as a passive element suitable for automated processing.

Technology Layer

The lowest realization layer in enterprise architecture metamodel is the technology layers, which is used to define and map application components to supporting software systems. For the same application component there can be different software available, which can be distinguished using the *Location* element.

- **System Software:** System software represents a software environment for specific types of components and objects that are deployed on it in the form of artifacts.
- **Location:** A location is defined as a conceptual point or extent in space (region, market, etc.).

The key relationships between the concepts are summarized as follows:

- **Process should normally be used to describe flow - P**rocesses describe the flow of execution for a function and therefore the deployment of a process is through the function it supports; i.e., an application implements a function that has a process, not an application implements a process.
- **Function describes units of business capability at all levels of granularity -** The term ‘‘function’’ or “application function” is used to describe a unit of business capability at all levels of granularity, encapsulating terms such as value chain, process area, capability, business function, etc. Any bounded unit of business function should be described as a function.
- **Application functions are deployed onto application components –** Application functions that are supported by IT are deployed onto application components. Application components can be hierarchically decomposed and may support one or more business services.
- **Application components are deployed onto system software (technology components) -** An application component is implemented by a suite of technology components, for example, application server software, application services and hardware.

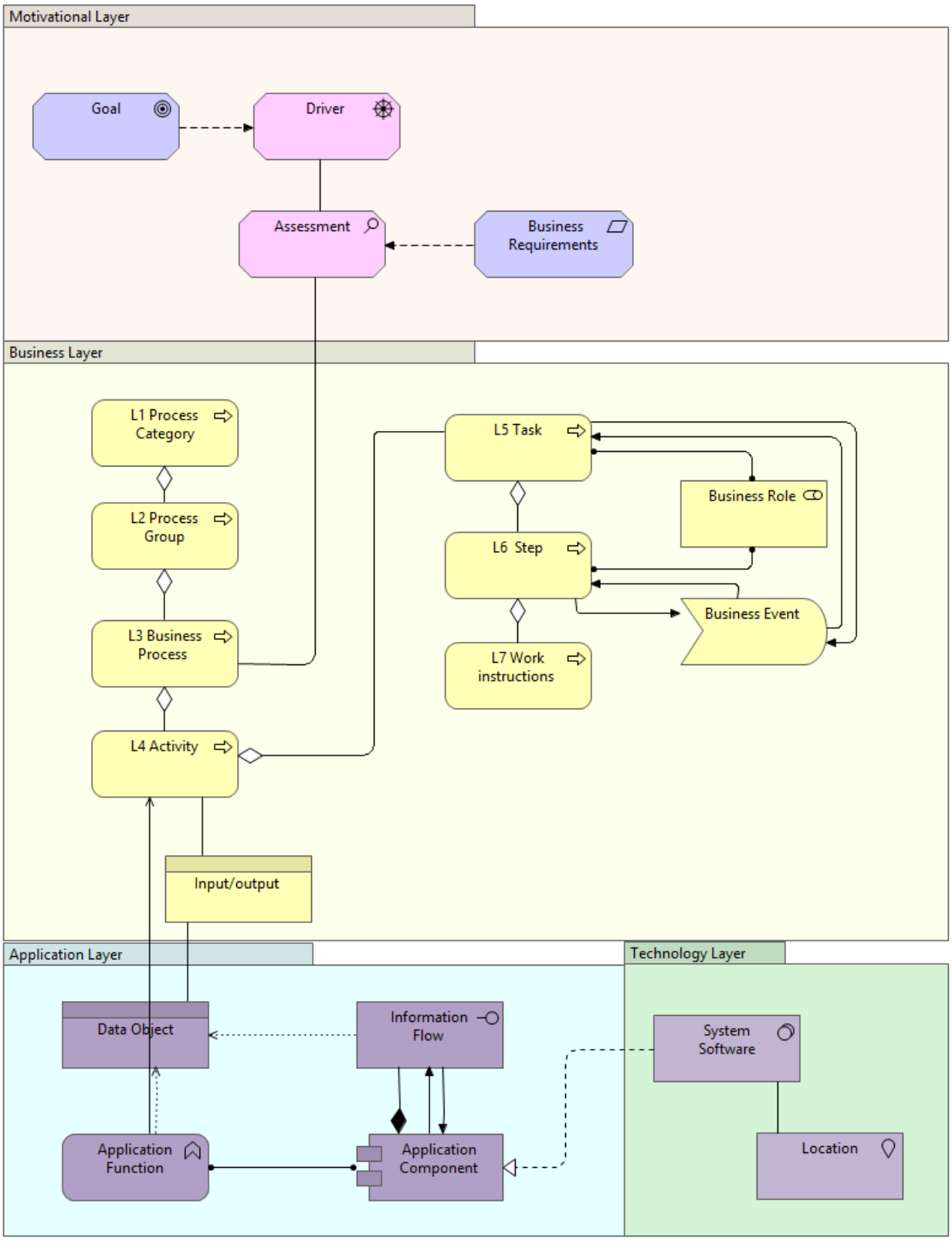


FIGURE 11. ENTERPRISE ARCHITECTURE METAMODEL (ADJUSTED)

3.2 Process of E-invoicing

In order to study the effect of differences in the country-specific fiscal requirements on electronic invoicing process at Philips, we explore the requirements related to the fiscal regulations and design constraints that influence the e-invoicing domain and process level (L3-L4). We collected the necessary data for the case study from various sources using various means. The process of case study research follows the steps defined by Runeson and Höst [16] which is explained in more details below.

3.2.1. Case Study: Methodology

Runeson and Höst [16] define a case study as an empirical method aimed at "investigating contemporary phenomena in their context" by making use of multiple sources of evidence (i.e. people, groups, organizations). One of the purposes of a case study is the exploration of the current problem to gain new insights and ideas for the research.

3.2.1.1. Design

The initial planning of the case study considers the research objectives, the ways how the necessary data will be collected and how it will help to answer research questions or to validate the proposed solution. The objective of the case study is to investigate the fiscal requirements for e-invoicing process and variability management issues related to the process design. The first objective addresses the first research question to collect the information about the fiscal requirements specific to each country in Latin America. The second objective relates to the second research question, for which we investigate current process design principles and constraints as well as related variability issues.

3.2.1.2. Preparation for data collection

In order to identify relevant sources and knowledgeable stakeholders about this matter, we organized several preliminary meetings with colleagues who either work in the related area, such as enterprise architecture, finance and compliance, or could provide references to the right persons in the organization. To understand the e-invoicing process tasks and related process design requirements, we held several meeting with the global business process owner in Finance and the regional business process expert. The outcome of the discussions is embedded in the process models. On the other hand, to investigate existing variability management approaches from the architecture design perspective we interviewed the business process analyst in compliance, two business architects and two IT architects. In addition, we asked each of the interviewees for a reference to relevant sources to have a better understanding of the context of the problem.

3.2.1.3. COLLECTING EVIDENCE

The data for the case studies can be collected in three ways: via direct or indirect methods or via independent analysis [16]. The first method is realized through direct interaction with the stakeholder or the primary sources of information. The indirect method is used to collect data without actual interaction with the subjects (e.g. where usage of a tool is automatically monitored), and independent analysis is accomplished through individual work to analyze the artifacts available about the case.

For the company case studies we collected data through direct method and independent analysis. The information from the stakeholders was obtained through individual semi-structured interviews and during multi-disciplinary team meetings and discussions. The idea behind the semi-structured interviews is to plan open questions ahead [16]. This type of interview set-up is used to allow more improvisation with the questions during the interview. Since the case study is of an exploratory purpose, more flexibility during the interviews is necessary to get a deeper understanding of the studied matter. The interview questions are more structured for the evaluation with a set of defined open-ended ones. The purpose of the evaluation is to obtain a qualitative assessment of the proposed solution from the interviewees, which enables us to validate the project results.

Part of the content discussed during the interviews and the meetings are only for internal use as they contain confidential information of the company. However, this information did not have influence on the choices of the project solution design and thus, can be left out from the scope of the report. The main findings about variability and its management are summarized in Appendix F: Interviews.

The main objective of the interviews was to understand variability modeling and management. The topics discussed during these interviews can be formulated as follows:

- **Q1.** How do you define variability in business processes (fiscal requirements, IT solutions)?
- **Q2.** How is the variability represented/modeled in the current architecture at Philips?
- **Q3.** What are the constraints considered to be considered when modeling variability?
- **Q4.** What are the issues related to the current way of managing variability?
- **Q5.** How can the current variability management approach be improved?

The part of the information about existing processes and the methodology of the enterprise architecture modeling were collected mostly independently based on the existing documentation available at the IT Architecture, Strategy and Platforms department. The studied documents included the training and presentation materials on the methodology and modeling guidelines of EA.

In addition to that, the new laws about the fiscal regulations were studied using online resources. The information about regulations can be found on the official government websites and in the white papers of the companies providing consulting or tax/e-invoicing solutions[1] in compliance with country-specific requirements. However, the information on the government websites is given only in country-specific native language, and because of the language barrier these sources were not used. On the other hand, the white papers provided by the third party service providers are in English, as are the webinars organized regularly around this subject. These webinars were a good opportunity to interact with the experts in the area and receive more country-specific details. The obtained information from the online sources was checked for relevance and accuracy with the local process expert of Philips in the region.

3.2.1.4. REQUIREMENTS ELICITATION AND DOCUMENTATION

The In order to coherently document the country-specific requirements effecting electronic invoicing process at Philips, we follow the common steps of domain requirements engineering for software product lines [6].

Elicitation

The elicitation of requirements took place mostly during the multi-disciplinary team meetings attended by the Business Process Owner, Business Process Expert, IT architect, and Business Analyst for Integration. The main focus of the team meetings was to analyze business and IT requirements for implementing e-invoicing process in Peru. But the stakeholders also discussed the current and coming requirements for e-invoicing process in other Latin American countries[2] (more specifically, the countries Philips conducts its business with: Argentina, Brazil, Chile, Colombia, Mexico, Peru, and Uruguay). Currently, the government-regulated invoicing procedure for Philips is in place in Argentina, Brazil, Chile and Mexico. In Colombia and Uruguay full requirements are still unknown; however, the government has announced that the process will become regulated for organizations in the coming two years.

A significant part of the process in these countries, except Brazil, is manual. The goal of the stakeholders is not only to design future processes in the models, but also to optimize and standardize the process across countries as much as possible.

Since the regulations on e-invoicing process in not in place, we collected additional information through individual interviews, online resources and webinars on the topic.

[1] www.invoicewareint.com; www.edicomgroup.com; www.trustweaver.com; www.einvoicingbasics.co.uk

[2] "Latin America" or "Latin American countries" in this report only refers to the countries where Philips operates, namely: to Argentina, Brazil, Chile, Colombia, Mexico, Peru, and Uruguay.

Documentation

The requirements were documented and refined in a textual format after each interaction with the stakeholders. No special format was required/used to represent the requirements initially. We only classified the requirements based on the generic classes they relate to. The fiscal authorities mostly impose regulations for the following subjects: authenticity and integrity of the invoicing process and documents, logistics, and archiving.

Negotiation

The negotiations usually take place between parties before the agreement is achieved. The governments publish a list of the companies or formally inform them when and how legislation on electronic invoicing effects their business operations [2]. Compliance to fiscal regulations is non-negotiable. On the other hand, trade agreement conditions can be discussed with customers and/or vendors. Such agreements are internally handled by the company, and are outside the scope of this project.

Validation and verification

The list of requirements has been iteratively checked for accuracy and clearness with the business process owner and local business process expert. However, the list of requirements may not be complete for several reasons: firstly, some of the countries still have not official published laws mandating e-invoicing (Colombia, Uruguay) and some still have ambiguities in the laws (Peru); secondly, we have not inquired full details of invoicing process in each country as our scope does not cover the lowest level of processes representing steps of the operations or application realization details; thirdly, since we cover *Invoice Customer* process where Philips is the supplier or invoice issuer, we do not consider regulations which address the buyer/invoice receiver side (e.g. reporting).

3.2.2. Case Study: Results

The main input to architecture design decisions comes from the requirements related to business goals, processes and IT support defined by key stakeholders. The requirements specify what the organization's objectives are and how they can be achieved. These requirements can originate from different sources, including but not limited to specifications related to organizational matters, governance, processes and application/technology/infrastructure. In addition to that, there are certain international compliance standards every organization has to adhere to, such as International Financial Reporting Standards (IFRS), Food and Drugs Administration standards (FDA), etc., or customer/vendor-specific trade agreements, such as International Commercial Terms (Incoterms) or country-specific legislations, e.g. on tax reporting and collection.

In the following sections first we define the meaning of electronic invoicing procedure, then describe the requirements by local authorities impacting the process and finally, explain how does e-invoicing process fit into the current design approach of Philips to modeling L3-L4 level processes.

3.2.2.1. ELECTRONIC INVOICE DEFINITION

Electronic invoice (e-invoice) is an alternative to the paper invoices and represents a fiscal proof of delivery of a product or services. The difference to the traditional invoicing method is that the e-invoicing is issued, transmitted and received in an electronic format. In order to be considered valid, electronic invoice should comply with country-specific legal requirements and business requirements on agreed sending and receiving parties.

Electronic invoice can be generated in a structured or non-structured format. Scanned invoices, PDF files sent by e-mail are types of non-structured invoices, while a structured format requires invoice data to be issued and exchanged in standard internet-based web forms via EDI or XML formats. According to the official definition by the European Union legislation, "an 'electronic invoice' is defined as an invoice that has been issued, transmitted and received in a structured electronic format which allows for its automatic and electronic processing" [44].

The process of generating and transmitting e-invoice is referred to as e-invoicing or e-billing, depending whether it's from the perspective of a selling or a buying organization. Even though in the case study used in this report, Philips is the selling organization, the process will be referred to as e-invoicing since it is the term most often used for the Business-to-Business (B2B) and Business-to-Government (B2G) segment.

Most common regulations on e-invoicing require Electronic Data Interchange (EDI) format to transfer the billing data, digital signature to ensure authenticity and integrity of the document and to authorize a service provider to communicate with the authorities. Thus, in this report we refer to the exchange of invoice in a structured electronic format with the involvement of local authorities as e-invoicing or regulated invoicing process.

3.2.2.2. E-INVOICING PROCESS REQUIREMENTS

It is important to keep an accurate and structured way of documenting and analyzing the requirements effecting the process design, as non-compliance may result in high fines or low operational efficiency [1].

The general requirements about electronic invoicing present in these countries altogether are presented in the Table 1. These requirements are not mandatory to each and every country, and some of them require different value per-country basis (e.g. invoice content). The possible

options whether a requirement is mandatory or not, or whether it represents a range of values is shown in column three *Options*.

TABLE 1. FISCAL REGULATIONS EFFECTING E-INVOICING PROCESS IN LATIN AMERICAN COUNTRIES

Requirement Class	Requirements	Options
Authenticity and integrity	FR1: Government requires invoice approval/validation	Alternative: {Yes, No}
	FR2: Government prescribes invoice delivery method	Value: {Paper, PDF, EDI}
	FR3: Invoice issuer needs to be certified	Alternative: {Yes, No}
	FR4: Invoice needs to be digitally signed	Alternative: {Yes, No}
Logistics	FR5: Goods must be accompanied with a document containing invoice authorization number	Alternative: {Yes, No}
Invoice structure and content	FR6: Invoice must follow government prescribed format	Alternative: {Yes, No}
	FR7: Invoice must include pre-defined number range to be present provided by the government	Alternative: {Yes, No}
	FR8: Invoice must be in government-specified language	Value: {Portuguese, Spanish}
Customer Invoice	FR9: Invoice sent to the customer must include gov. authorization number[3]	Alternative: {Yes, No}
Archiving	FR10: Invoices must be available for downloading online (government-specified number of months/years)	Alternative: {Yes, No}

3.2.2.3. L3-L4 PROCESS DESIGN (AS-IS)

The process affected by the fiscal legislation and considered in this project is the *Invoice customer* process. Invoicing is a global process for sending billing documents to customer for ordered products or services, and for reflecting the effect of transaction in terms of taxes, receivables and revenues in accounting documents. The process of issuing invoice is a part of all the business models in all markets for Philips. Business model is the highest level where the variation starts. From the defined four business models, the base model of *Invoice Customer* and the case of regulated invoicing concerns the *Product model.*

Invoice customer (L3) is a sub process of *Perform Revenue Accounting* process group (L2). The latter belongs to the *Finance* process category (L1), which is one of the enabling functions

[3] Please, note that the *authorization number* assigned by the government in FR9 is the same as the *protocol number* used in the process models.

defined at Philips (Please refer to the Philips Enterprise Architecture Metamodel for a high-level overview of process classification in Figure 11). There are multiple L1 and L2 process groups defined at Philips, but we will not list the variants of each class since they are irrelevant for the case and represent the company's internal information.

In order to make it easier to follow which process represents which level, we give each process a multi-level numbering. Since we deal with only one process group, we assume that the *Finance* process category is the first variant (Please, note that the numbering does not correspond to Philips process framework). The processes are classified according to their granularity and start with the lowest L1- Process category up to and including the L4 – Activity. The process levels describing tasks (L5) and steps (L6) are not included in the numbering range because they represent the operational part; thus, they include a large number of objects which are frequently modified during the process design. The detailed example of *Invoice Customer* process is given in Table 2. Process Level Classification

TABLE 2. PROCESS LEVEL CLASSIFICATION

Process Level Classification	Example of *Invoice Customer* process
L1: Process category	1. Finance
L2: Process group	1.1 Perform Revenue Accounting
L3: Process	1.1.1 Invoice Customer
L4: Activity	1.1.1.1 Generate customer billing data 1.1.1.2 Transmit billing data to customer 1.1.1.3 Post account receivables 1.1.1.4 Recognize revenue

The generic or base process of *Invoice Customer,* as already defined at Philips, consists of four main activities. Since the variation due to fiscal requirements only impacts the *Transmit billing documents to customer* activity, we give more details about the process. Short descriptions for all of them are given below:

1.1.1.1 Generate customer billing data – the activity is triggered automatically in the system when the payment is due, the billing run is scheduled or the invoice transmission is required. The process also covers generating billing documents, determining when to create accounting documents, and triggering resolution actions in case of an error in the generated documents.

1.1.1.2 Transmit billing documents to customer – this process follows the steps to determine what he preferred method of transmitting billing documents to the customer is and how they can be transferred. The activity consists of the following L5 tasks:

- *L5: Determine invoice distribution method*
- *L5: Print invoice at shipping location*
- *L5: Send EDI invoice*

- *L5: Send PDF invoice with email*
- *L5: Send printed invoice*
- *L5: Trigger resolution of incorrect invoice transmission*

1.1.1.3 *Post receivable entries* – the process deals with reflecting billing transactions into accounting documents, such as debit and credit notes, as well as resolving issues in case of blocks during posting.

1.1.1.4 *Recognize revenue* – the activity describes a procedure for calculating, settling and posting costs and revenues based on product sales.

A simplified model of the standard process of *Invoice Customer* is given in Figure 12. The model only shows the control flow using the complex objects using BPMN 2.0 notation. The process itself is a reusable component and

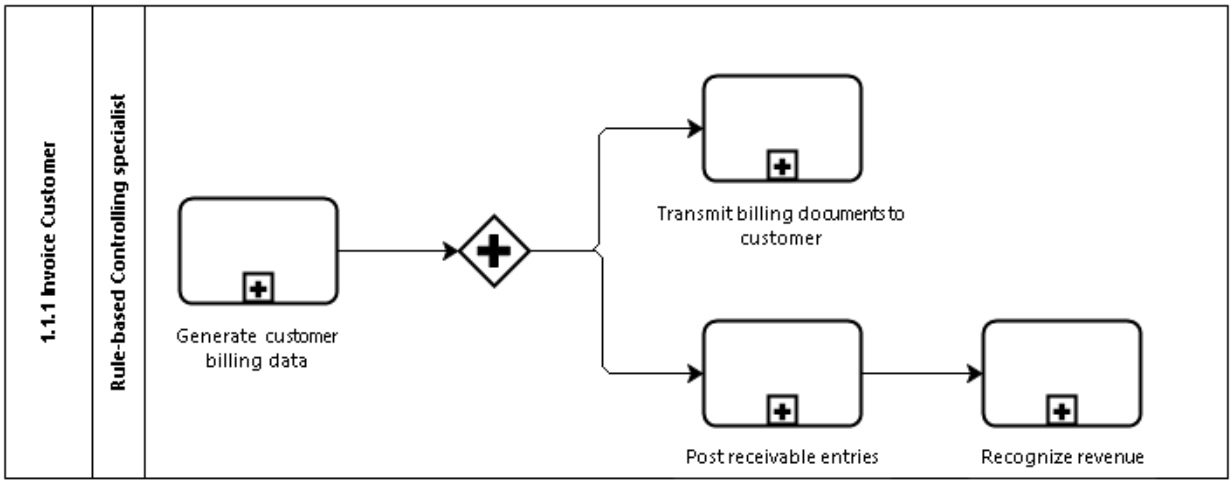


FIGURE 12. GLOBAL INVOICE CUSTOMER PROCESS (WITH ACTIVITIES ONLY)

The analysis of the country-specific requirements helped us to identify 2 variants of *Invoice Customer process* on L3 and 3 variants of *Transmit Billing Document to Customer* on level four. The high-level overview of variant breakdown is as follows:

L3: 1.1.1 Invoice customer

1.1.1a Invoice customer (standard)

1.1.1b Invoice customer (regulated)

L4: 1.1.1.2 Transmit billing data to customer

1.1.1.2a Transmit billing data to customer (standard)

1.1.1.2b Transmit billing data to customer (regulated)

- Shipment authorization
- Invoice approval
- Document presentment

Because of the variety of implications, the goals and the needs of the stakeholders interviewed were also diverse. Based on the interviews and meetings throughout the project, the following stakeholder concerns with respect to management of variability in compliance requirements and modeling on L3-L4 processes were identified:

- There is no clear guidance how to capture and document variation in the compliance requirements;
- There is no clear approach for modeling variants on L3 and L4 processes;
- There is no clear way to represent the impact of differences across layers of enterprise architecture.

3.3 Concluding Remarks

Investigation of the company environment and the case study showed there is no well-defined approach used at the company to handle the variability caused by the differences in the requirements. In order to describe and document the information about the variations from the case study in a structured manner, we will use the concepts related to variability causes, subjects, objects, and relations to build the conceptual model in the following chapters.

4. VARIABILITY IDENTIFICATION

This chapter describes the analysis of concepts related to variability management and their application to e-invoicing case. Structured documentation of variability improves communication about the variation in information systems artifacts and helps in decision-making by explicitly showing variable parts of the system or architecture [6]. In addition to that, documenting variability enables better traceability between the sources of variation and the corresponding variable artifacts. Existence of such linkages is useful for managing changes in the requirements. Architects need to be able to trace variability defined in the models to other architecture artifacts, such as textual documents, catalogues, other models, etc.

The analysis of the concepts from literature will be used as an input to create the conceptual model of variability management extension presented in the following Chapter 5. And, the result of its application to e-invoicing case will guide the modeling process explained in Chapter 6.

4.1 VARIATION DRIVERS

One of the first steps in variability management is to identify and describe variability. Variation in the artifacts can be expressed on different layers of architecture. For example, in SPLE the variation starts during the requirements analysis, which is represented on the top layer of variability management architecture [6]. Requirements analysis helps us to study variability-related causes and identify impacted objects in the processes and IT landscape of the organization.

Variation in process or enterprise architecture design occurs due to various reasons. Capturing variation in the architecture increases the complexity of the design, and requires more support for maintenance [11]. For this reason, it is important to minimize the variation and the modeling efforts during the design. One way to achieve this is to understand the causes of variability or *variation drivers* [43]. Understanding and categorizing process variants according to variation drivers can support in design decisions in architecture, which is addressed later in the report.

One of the factors driving variation in the processes can be a business-related reason, such as environment, resource or market in which the business operates. Besides variation within the processes, process variants can also be derived by the differences in the way they produce outcomes. The first class of factors playing role in process variation is referred to as *business drivers*, while the latter is called *syntactic drivers* [43].

Business drivers can be further categorized based on the specific context in which they are used in the organization. The framework presented in Figure 13 shows the classification of variation drivers [43]. Process variants can be derived not only because of the differences in the operational flow, but also depending on *what* type of product/services the company produces, to *who* it is targeted *where* and *when* it is going to be delivered to the customer. Categorizing variation drivers into fewer classes makes it simpler to generalize the causes of variation and to reuse the method for different cases.

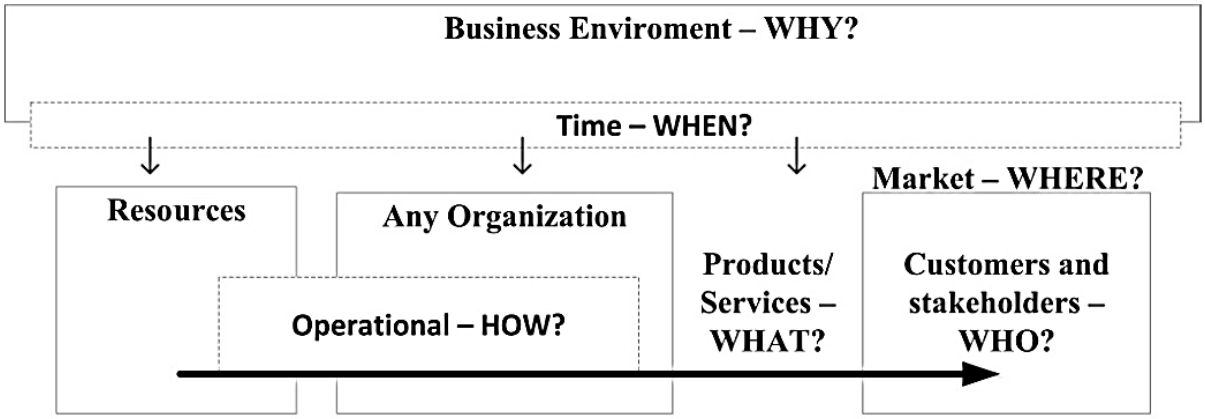


FIGURE 13. FRAMEWORK FOR CLASSIFICATION OF (BUSINESS) VARIATION DRIVER [43]

With the involvement of experts, the requirements were classified according to the compliance domain. These classes of requirements were then matched with the variation drivers, based on the context and type of the driver as presented in the framework on Figure 13. For the e-invoicing process, the differences in the requirements originate from fiscal regulations. Thus, one of the variation causes is identified as market-related. As some requirements affect the process execution and implementation, another variation cause is related to the operational aspect.

4.2 VARIATION POINTS

In order to extract and analyze variability information from the listed requirements and constraints, we need to understand the variation points or the location in the artifacts, like process or application components, where the difference occurs [7] [33] [13].

Variability in enterprise architecture can be present in all levels of architecture. Respectively, the variation points are relevant to business, application and technology layers. As the levels of architecture represent realization layers of the enterprise architecture, also variation points on one level indicate how the choice of one variant in business layer may affect its possible realization options in other layers.

During the meetings stakeholders, including the Business Process Owner, Business Process Expert and IT architect, we discussed and identified points in architecture where there was more than one option or alternative for an object. The discussions to analyze variation options started with requirements and business needs. Later meetings to discuss business and IS aspects were held in parallel to ensure that the information from both sides were consistent and aligned with each other. For the reporting purposes, we follow the enterprise architecture realization levels from top to bottom and start with the description of differences in the business layer followed by the implementation possibilities.

4.2.1. Business Architecture Layer

Given the base process, the fiscal requirements and the variability matrix, we identified the branching points in the generic process and added new tasks to reflect government-involvement in the process. Identifying branching points is one way to define variation points in the process [5]. Milani et al. [5] distinguish two types of branching points in the processes: *variation* and *decision points.* A branching point is v*ariation points* if it splits the process into process variants, and it is a *decision point*, otherwise.

Together with the business process owner and expert, we identified three variants that occur in the sub process of *Transmit billing documents to customer*. Depending on the fiscal regulations, an invoice issuer can be obliged to get the invoice approved before shipping the goods, and/or can be required to send only the approved invoice to the customer. These variants are caused because of the differences in the control flow of the process. Other type of variation occurs due to differences in the business object exchanged or the even triggering the start of the process.

4.2.2. Application/Technology Architecture Layer

The variants in the business processes trigger variation in the application layer because of the specific requirements or options for their realization. In Application architecture variability refers to different ways of supporting the activities and different versions of the technologies that implement the process.

The application layer includes the following objects that support business functionalities:

Information flow – the type of communication channel with the government which can be synchronous or asynchronous;

Application function – represents software functionalities that are necessary to support business activities (L4). For e-invoicing case, together with the IT architect, we identified the following application functions: Manage number ranges, Message creation, Message transformation, Message transfer, Message monitoring, Message encryption, Archiving.

Application Component – a group of reusable functions that is implemented by software or hardware component. The application component supporting invoicing process, including posting accounting documents and recognizing revenue, is Enterprise Resource Planning (ERP) Central Component.

System Software - Software or technology component that implements given application function or a set of application function. Given the government-imposed rules of business processes, available software vendors in the countries the following options for communication with the authorities have been identified:

1. Automated process with full company ERP support;
2. Local 3rd party service provider with a license to communicate with government;
3. Log-in to the government website manually.

The mapping of the identified variation points to the respective business and IT architecture objects in the architecture are presented in Table 3 below.

TABLE 3. VARIATION OPTIONS MAPPED TO CORRESPONDING OBJECTS ACROSS ARCHITECTURE LAYERS

EA Layer	Variation Points	Variation Options
Business Architecture, L3 process	1.1.1 Invoice Customer	18.2.2a Invoice Customer (standard) 18.2.2b Invoice Customer (regulated)
Business Architecture, L4 activity	1.1.1.2 Transmit billing data to customer 1.1.1.2a Shipment authorization is required 1.1.1.2b Invoice approval is required 1.1.1.2c The government requires a copy of invoice	{required, not required}
Business Architecture, Data	L5: Determine invoice distribution method	{printed, PDF, EDI}

Business Architecture, Event	L4: Generate customer billing data	**{**Billing run scheduled, Goods accompanying invoice required}
Application Architecture, Data object	Business object: Input/output	{Gov.-prescribed format, non-gov.-prescribed format} {SPA, POR} Digitally signed
Application Architecture, Information Flow	Information Flow: communication	{Synchronous, asynchronous}
Application Architecture	Application	{Cockpits, fire-and-forget}
Technology Layer	Component	{ERP + in-house dev., outsource)
Technology Layer	Component	{Any certified service provider, Gov. specified service provider, certified supplier}

4.3 Dependencies and Constraints

Variability dependencies are restrictions on the variant selection of one or more variation points and are used to capture the rationale behind variability decisions. Dependencies are caused because of constraints in the business (i.e. stakeholder) requirements, realization, or restrictions on quality attributes [38]. Constraints in information systems represent architectural concerns imposed by stakeholders or environment (technical, regulatory, etc.) [29]. Constraints dictate which variation point or variant depends or is dependent on another element in the model. For example, "the selection of variant *a* at variation point A excludes the binding of variant *b* to variation point B."

Two types of constraints *requires* and *excludes* describe dependency relationships between variation points and variants [6] [37]. Some of the common constraints affecting the design of enterprise systems architecture are maturity of legacy IT systems, integration-related issues with existing or external software, vendor lock-in (when design of architecture is restricted by a

vendor possessing required knowledge in the domain), or the mismatch between software and processes [29].

Given the limited time and available information on existing architecture, software vendors, and legal environment in Latin America, we have identified constraints that limit the e-invoicing process-related flow and the realization of identified variants. The implementation-related constraints for e-invoicing case are mainly influenced by the necessity of integrating the company's information systems with local government systems.

The dependencies between the variation options identified in the processes and possible realization support is summarized below:

1. Invoice approval process *requires* integration with government systems;
2. Invoice issuer must be a government-certified party, which limits the implementation of encrypting and transferring billing data to one of these three options:
    a. Philips gets certified (by going through homologation process) by the local government;
    b. The process is outsourced to one of the local government-specified service providers;
    c. The process is outsourced to one of the certified service providers.
3. Shipment authorization process *requires* real-time communication channel (synchronous information flow) with the government systems;
4. Synchronous information flow *requires* implementation of cockpits for enabling information status monitoring of the invoicing process;
5. Synchronous information flow *excludes* implementation of 3rd party service with online portal for invoice status storing and retrieval (fire-and-forget principle).

4.4 Concluding Remarks

In this chapter we analyzed literature to define variability-related concepts applicable in the enterprise architecture context. In addition to that, we used the requirements related to the case study to identify and structurally represent variability cause by the compliance requirements.

In the following chapters, we create a conceptual model of variability management to illustrate its fit in the architecture. And, we use the identified variation points, variation options and dependencies from the case study to build respective models for business and application/technology layers.

5. EXTENDED ENTERPRISE ARCHITECTURE METAMODEL

Being an international and multi-profile company, Philips has to comply with numerous policies and regulations, such as medical device regulations (FDA), financial reporting (ICS/SOx), privacy regulations, export controls related to embargoed countries, etc. Compliancy to all these regulations is one of the most important requirements that Philips has to satisfy in its business processes, IT systems and tools.

The implications related to adoption of e-invoicing can be related to several aspects, including legal and technological ones. The legal aspect concerns the lack of standards or uncertainty at a regional and country level, while the technological aspect is related to the complexity, and internal and external compatibility of the system requirements [45].

In information systems, an enterprise architecture metamodel is used to define and represent relationships between system artifacts. The current enterprise architecture model used at Philips does not support variability concepts in its architecture design (Section 3.1). In order to enhance architecture metamodel with new concepts we customize its metamodel extension [18] [24]. ArchiMate 2.1 supports a Motivation extension to represent internal and external factors that influence the design of business, application and technology architecture design. Using the existing motivational objects such as *driver, requirement* and *constraint* we defined an extension that supports the variability concepts identified in the previous sections and shows the relationships between design artifacts.

5.1 VARIABILITY MANAGEMENT EXTENSION

From the enterprise architecture perspective, the company has formulated core principles for business and IT architecture design. For business architecture, the design principles defined at Philips suggest to make maximum use of constructed re-usable building blocks, and thus, reduce redundancy caused by modeling related process variants separately. The principle for IT architecture design is to use standard and proven technologies from reliable service and technology providers.

In addition, Philips follows proven practices to organize and model its business processes. The main goal of the business architecture is to reduce the complexity by creating a simplified and generalized version of the model. In order to obtain the desirable outcome, modeling process has to focus on the clarity, and consistency of each model.

We are going to use the core content metamodel which was created at Philips and add an extension for variability management to it. The extension of enterprise architecture for variability management, proposed in the report, is used to describe additional design elements and concepts that handle modeling issues caused by variability in the requirements.

There are several ways to extend a metamodel of architecture, which include the built-in mechanism, metamodel customization and model annotation [24]. Since our objective is to support variability management in the architecture, the most effective way to support separation of concerns is to use built-in extension mechanism and model annotation (see Figure 14).

Extension Mechanism	(1) Enhancement	(2) Augmentation	(1) and (2)
Built-in	- introduction of accidental complexity through meta-model element duplication	+ separation of concerns by separating problem domains of host language and augmentation	+ Pluggable extensions - not standardized across tools and modeling languages
Meta-model Customization	+ Direct change of enhanced meta-model elements	- mix of augmentation domain and host language domain, because of hard wired packages	+ At first glance, very simple - not supported by most state-of the art meta-modeling frameworks - Can break existing toolings - Can cause redeployment
Model Annotation	- introduction of accidental complexity through meta-model element duplication	+ separation of concerns by separating problem domains of host language and augmentation	+ indepenent of tools or modeling languages + Pluggable extensions - Tools need to be customized for good integration

FIGURE 14. PROS AND CONS OF EXTENSION MECHANIMS [24]

Each development phase follows certain specifications concerning business, technical, implementation or design requirements. Managing architecture requirements is a dynamic process which reflects the changing conditions in the environment, as well as the changing objectives of the stakeholders. Thus, the *Requirements Management* needs to be supported during the full EA development cycle.

The output of the *Requirements Management* is used as an input for developing the architecture of Business, IS and Technology layers. They are represented in the *Motivation Layer* of enterprise architecture at Philips which is an extension layer with motivational concepts, provided by ArchiMate to support managing requirements for architecture development in TOGAF [18].

The architecture should reflect how the goals are supported by the requirements and how the requirements are realized by business processes and software applications. Thus, the requirements need to be constantly maintained for changes or updates based on internal or external drivers. TOGAF does not recommend a particular tool or language for requirements management, but the motivational layer by ArchiMate can be used to reflect the impact of fiscal regulations as a driver to variation in processes and applications.

In order to describe the concept of a variation driver, we use ArchiMate 2.1 notation of a *driver*. The *driver* element is a part of an extended motivational layer of ArchiMate and is defined as "something that creates, motivates, and fuels the change in an organization" [18]. Drivers are considered to be essential factors that influence other motivational elements, such as *goals*, *requirements*, and *constraints.* Defining the motivational elements and relationships between them is necessary for understanding the context and reason lying behind the enterprise architecture.

ArchiMate 2.1 will be used to depict the traceability between variation drivers, requirements and architectural elements as the language covers all these layers of enterprise architecture (see Figure 15).

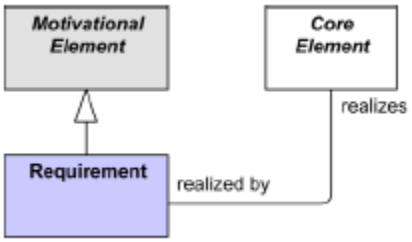


FIGURE 15. RELATIONSHIP BETWEEN REQUIREMENT ELEMENT AND OTHER ARCHITECTURE ELEMENTS

The detailed relationships between variability concepts are presented in Figure 16. The objects in orange color indicate the extended elements related to variability concepts defined earlier in the report. The objects in yellow color are not explicitly shown on the core content metamodel, and they describe the rationale behind the variability management approach.

Variation drivers are categorized according to their business context. The categorization is influences by the company's requirements, including IT-governance, compliance, strategic, etc. The variation drivers help to determine variation points in the models. We reuse a subset of variability related concepts defined by the Orthogonal Variability Model approach [6], such as variation points, variation options and dependencies. However, during the discussions with stakeholders, we dismissed the option of adopting the full approach due to the necessity of a new notation and a separate model. Both of these elements conflict with the architecture principles defined at Philips require maximum standardization of the methodologies and minimum maintainability efforts (partially achieved by less number of models).

Variation points represent locations in the model where a selection or choice between options has to be made. Variation options can be optional, mandatory or alternative [6]. Variability matrix is a document that associates existing alternatives to the corresponding variation points. The relationships between variation points and variants are constrained by dependencies. The dependencies, on their own, are restricted by constraints which may relate to business-related constraints, such as cost/budget during process design, or IT-related constraint, such as integration with vendor's software systems. Given the constraints, the dependencies either may

necessitate selection of a particular variant (*require)* or limit the possible options (*exclude)* [38] [6]. A variation point can have many possible resolution ways, so the mappings between variation point and variants are many-to-many.

In addition to the core variability concepts defined in SPLE, we add *a variation driver* in the conceptual model to illustrate the categorization of causes of variation that effect processes [5]. The conceptual model of variability relationships is given in Figure 16.

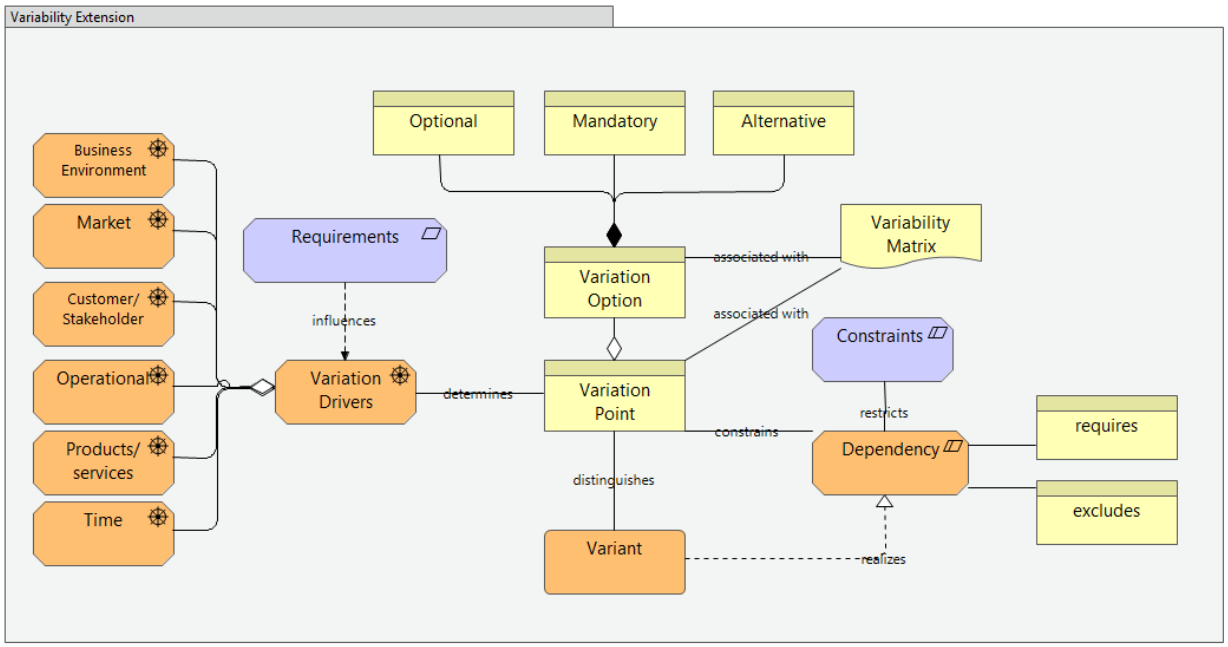


FIGURE 16. VARIABILITY MANAGEMENT CONCEPTUAL MODEL

In order to adequately document the extension of the metamodel, we discuss the purpose and the required changes to the core content metamodel in details [21]:

Purpose

The variability management extension is intended to guide identification of causes, rationale and documentation of variations in the enterprise architecture.

The scope of this extension is to demonstrate:

- The ability to describe root causes of variation or variation drivers;
- The ability to show links and relationships between variability concepts and other architectural components across EA layers.

The extension should be used in the following situations:

- When the differences in the requirements or other variation drivers impacts the design of processes, applications or technologies.
- When there is a need for optimization of possible variants and of modeling efforts.
- When there is a need for tracking the links between selected choices across EA layers.

The benefits of using this extension are as follows:

- The causes of variability are elicited and documented in a structured way.
- It enables reusability of architecture components.
- It provides a better traceability between variants across architecture layers.

5.2 REQUIRED CHANGES TO METAMODEL

Changes to the metamodel entities and relationships are as follows:

- New entities are added: *variation drivers, dependencies, variant.*
- New relationships are added: Constraints *restrict* dependencies; Variation drivers *distinguish* variants through derived relationship via variation points.
- Added a *constraint* to the Motivation layer. A constraint is defined as a restriction on the way in which a system is realized.
- Stereotypes:
    - <<Variation Point>> - location of variation;
    - <<Alt_varpoint>> - alternative variants;
    - <<Mand_varpoint>> - mandatory variants;
    - <<Opt_varpoint>> - optional variants;
    - <<Variant>> - process, application or system software variant.

Additional diagrams to be created are as follows:

- Variability matrix to allow mapping of possible variation options with respective variation drivers/variants

The architecture extension integrated with the core content metamodel is illustrated in Figure 17. The elements in orange color indicate the objects which are added as the architecture extension. *Variation points* and *Variation Options* given in the conceptual model (Figure 16) are embedded using stereotypes [41] within business, application, and technology layers. And the relationships *requires* and *excludes* are represented as annotations to the linking lines.

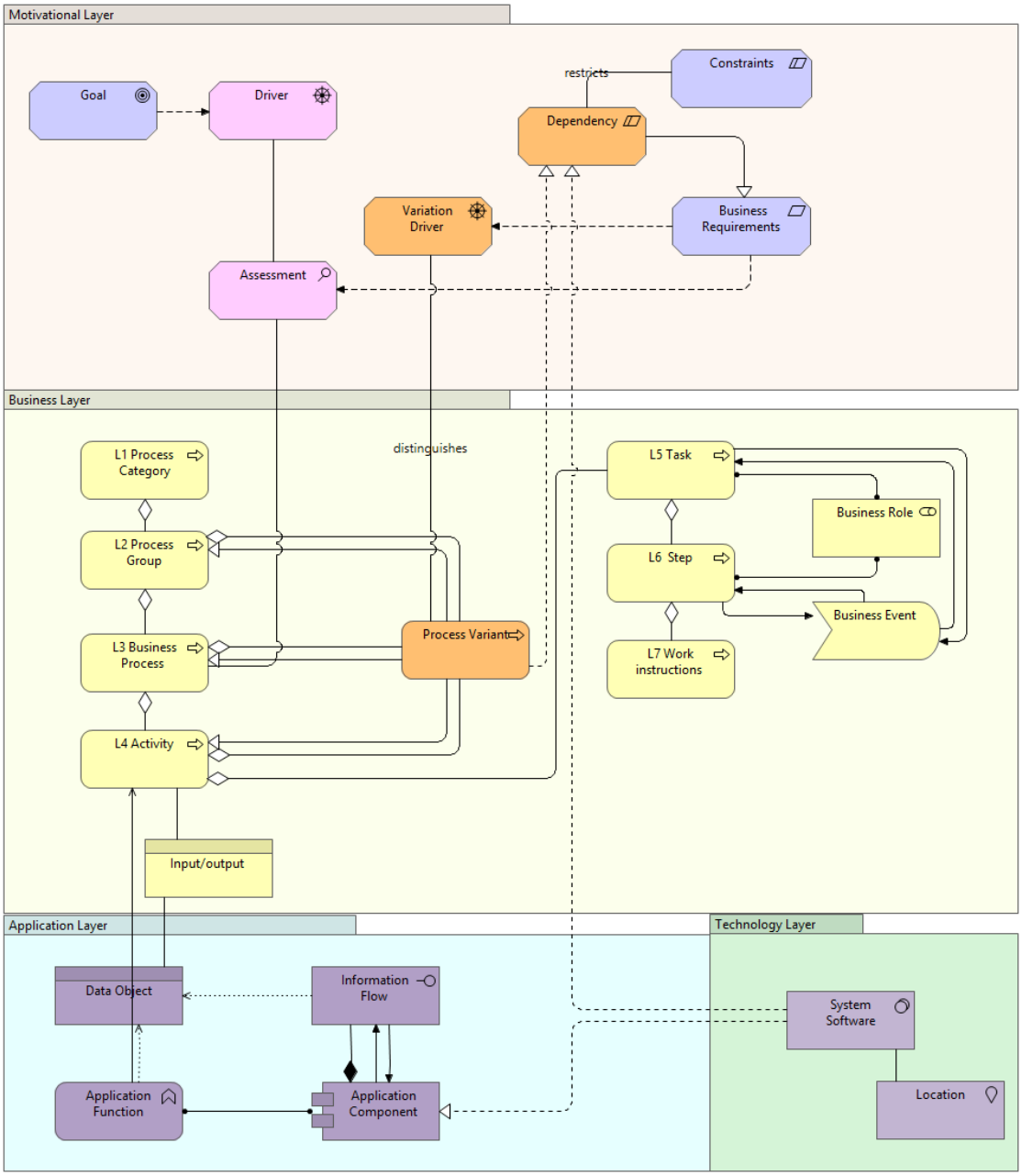


FIGURE 17. EXTENDED ENTERPRISE ARCHITECTURE METAMODEL

5.3 CONCLUDING REMARKS

In order to structure the variability concepts described in the previous chapter, we propose a conceptual model. The conceptual model illustrated the relationships and influences of the variability concepts and the architecture elements. Using the conceptual design, we have extended the enterprise architecture metamodel with the objects concerning variability management.

6. VARIABILITY MODELING

In the previous chapters we have defined how to identify and structure variability and how to incorporate it as an enterprise architecture extension. In this chapter, we model the e-invoicing case using the variability concepts identified in Chapter 4. As described earlier, the E-invoicing is the L3 level process with three variants identified in L4 activities. Following the enterprise architecture methodology, we design models specific to the case study with extended variability concepts. We present three views to show the process design and supporting applications used at Philips. The first view is the L3 and L4 process design modeled in BPMN, second view represents the matrix used to map L4 activities to corresponding application functions, and the last view is the Function Allocation Diagram showing the L3 business process with related requirements and supporting technologies.

6.1 MODELING PROCESS

In order to describe the variability in the enterprise architecture for the e-invoicing case we use three viewpoints: Business process models covering L3, L4 and L5 levels, Application function/L4 Activity matrix, and Function Allocation Diagram. Choices for modeling techniques for each view were determined separately. The guidelines on how to incorporate variability management during the architecture design is given in Appendix G: Variability Management Method.

The process of modeling architecture consists of several steps. When creating model in enterprise architecture, Lankhorst [23] suggests to first establish the purpose, scope and focus of the models, then to structure the content of the models and finally to visualize them using selected views.

Establishing the purpose, scope and focus

The purpose of modeling process of electronic invoicing is to get insight into processes, IT infrastructure, and their alignment.

- The scope of the models is L3 – invoice customer process, with corresponding application and technology support.
- The focus is to show variation caused by country-specific fiscal requirements on the processes and implementation options.

Creating and structuring the model

Based on the elicited requirements, we model the structure in order to make the essence easy to recognize and understand. In the case of e-invoicing, we first structured the requirements based on the country-specific scenarios, in order to identify commonalities and differences between the variants.

6.2 Structuring Requirements

The requirements regarding fiscal regulations have been elicited from global and local process experts in finance. The requirements are presented in the form of a matrix with corresponding countries as columns in Table 4. Each row cell is filled in with either Yes/No, indicating the existence of the requirement per country, or with a certain value, such as language or data type. The far left column lists the same fiscal regulations presented in Table 1. The cells in light green color indicate the assumptions of the possible regulation to be introduced in the respective countries. The cells in red, on the other hand, indicate the differences or deviations from the general pattern. The color-coding was used simply for better visibility purposes.

Because of either the prolonged process of eliciting fiscal regulations via indirect communication with stakeholders, or still lack of official and clear information from respective governments, there is no definite information concerning some requirements. In certain countries, like Columbia and Uruguay, it is expected that the legislation on mandating e-invoicing will be effective in the following years. In order to avoid completely ignoring the missing requirements, based on the consultation with the domain architect, we assumed “the most likely” cases (based on the information on current practice and expected legislations).

TABLE 4. COUNTRY-SPECIFIC REQUIREMENTS MATRIX

Fiscal Requirements	Argentina	Brazil	Chile	Colombia	Mexico	Peru	Uruguay
FR1: Invoice must be approved by the gov.	Yes	Yes	No	No	Yes	Yes	Yes
FR2: Government prescribes invoice delivery method	EDI	EDI	EDI	EDI	EDI	EDI	EDI
FR3: Invoice issuer must to be certified	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FR4: Invoice must to be digitally signed	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FR5: Goods must be accompanied with a doc. containing invoice authorization number	No	Yes	No	No	No	No	No
FR6: Invoice must follow government prescribed format	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FR7: Invoice must include pre-defined number range to be present provided by the government	Yes	No	Yes	No	Yes	No	Yes
FR8: Invoice must be in government-specified language	SPA	POR	SPA	SPA	SPA	SPA	SPA
FR9: Invoice sent to the customer must include protocol number	Yes	Yes	No	Yes	Yes	No	Yes
FR10: Invoices must be available online for download (government-specified number of months/years)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

6.3 VISUALIZING THE MODELS

Based on the Philips architecture methodology, principles and constraints, we visualize the process models in BPMN as L3 and L4 processes and show the impact of variation on the Application and Technology layers of EA using high-level relationships.

Developing and modeling enterprise architecture requires considering the different constraints imposed by the organizational structure as well as external factors. Some of the constraints that affect the design of enterprise systems architecture include existing legacy architecture and systems, system integration, vendor lock-in, interactions with external parties (government), quality attributes as drivers (e.g. variability) [21].

6.3.1. BUSINESS ARCHITECTURE

The variation driver as identified in Section 4.1 affects the operational part of the invoicing process. Thus, the process levels which are impacted by the differences include Invoice Customer (L3) and include its sub processes - L4 activities, L5 tasks, and L6 steps. Given the detailed analysis of the fiscal regulations and the variability matrix for country-specific requirements Table 4), we located branching points in the generic process of *Invoice Customer*.

The variants in the business processes were selected on two levels:

L3: 1.1.1 Invoice customer

1.1.1a Invoice customer (standard)

1.1.1b Invoice customer (regulated)

L4: 1.1.1.2 Transmit billing data to customer

1.1.1.2a Transmit billing data to customer (standard)

1.1.1.2b Transmit billing data to customer (regulated)

- Shipment authorization
- Invoice approval
- Document presentment

The new process model with two variants of *Transmit Billing data to customer* is presented in Figure 18. Please, note that the model is L3 process model containing L4 activities. Thus, all the tasks on the model are complex tasks and include sub processes modeled separately. The sub process of *1.1.1.2b Transmit billing data to customer (regulated)* is described later in the section with corresponding model.

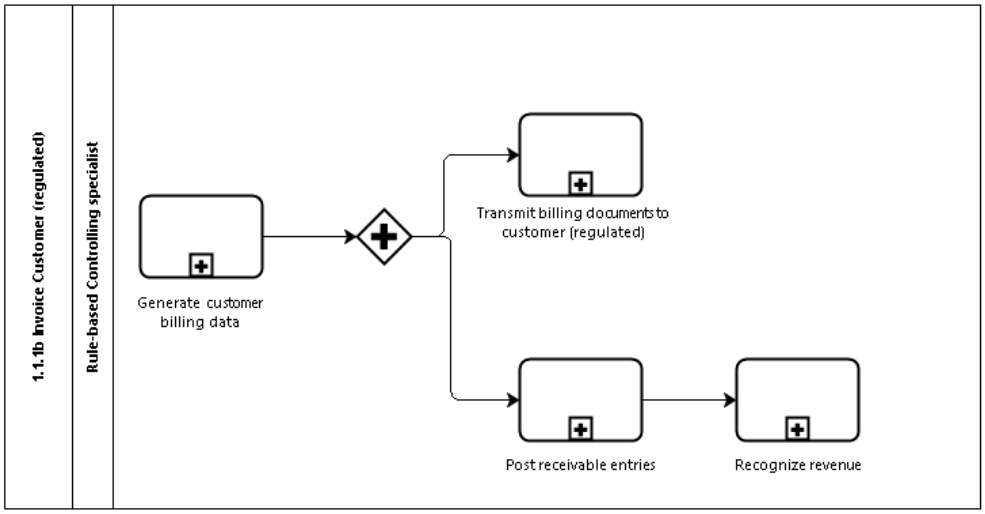


FIGURE 18. INVOICE CUSTOMER PROCESS MODEL

Having considered the existing modeling constraints at the company, and pros and cons of single artifact vs. multi-artifact variability modeling techniques, we decided:

- To use multi-artifact approach to model L3 process variants as separate models because of the clear visibility of business needs for the markets as well as the possibility for reusability of the artifact; and to distinguish between L4 - Transmit Billing data to customer standard and regulated, since there is a difference between application functions supporting the process. For this reason, we decided on two separate objects needed for mapping in the matrix shown in Figure 21.
- To use single-model approach when documenting variation for L4 processes. The key considerations behind the decision are the following: the variants are syntactically similar or have in common a significant part of the process; the activity can be reused in scenarios which are not market-dependent; the stakeholders wanted to see the complexity of the variants in one process to see the possibility of configuring the single ERP system that supports the process;
- To use UML stereotypes in order to show the variants of the business object transmitted during the process (gov.-specified invoice format vs. standard format);
- Other variation points which do not affect the variant flow are depicted as branching point.
- Please, note that we do not address variation in the roles (business actors) in these models. The differences in the roles, according to Philips process classification framework, are addressed at L5-L6 process levels.

The single model combining all three variants of L4 - Transmit billing data to customer (regulated) is shown in Figure 19 and Figure 20. The variants modeled separately are given in Appendix H: Process Model Variants.

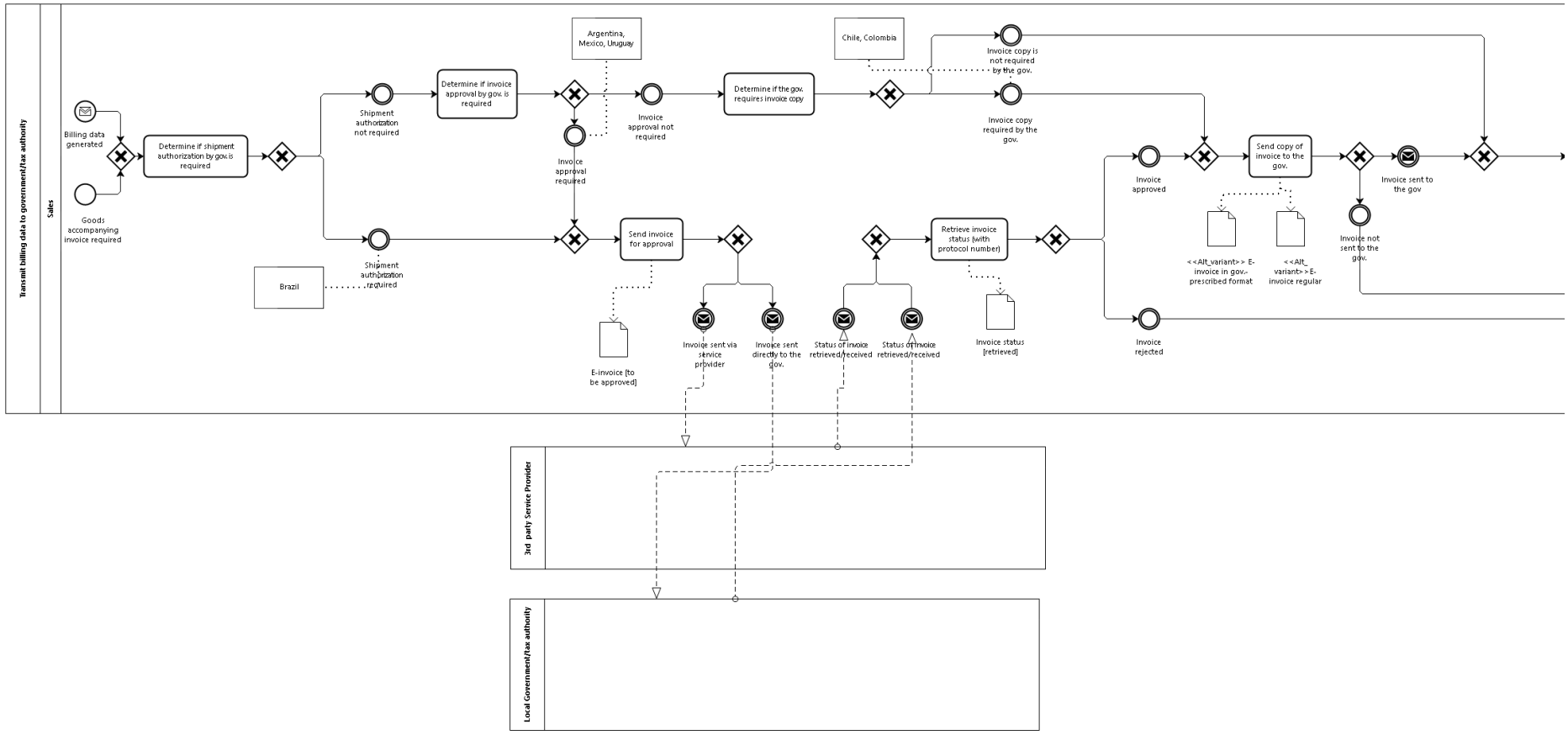


FIGURE 19. L4 - TRANSMIT BILLING DATA TO CUSTOMER (REGULATED) (PART 1)

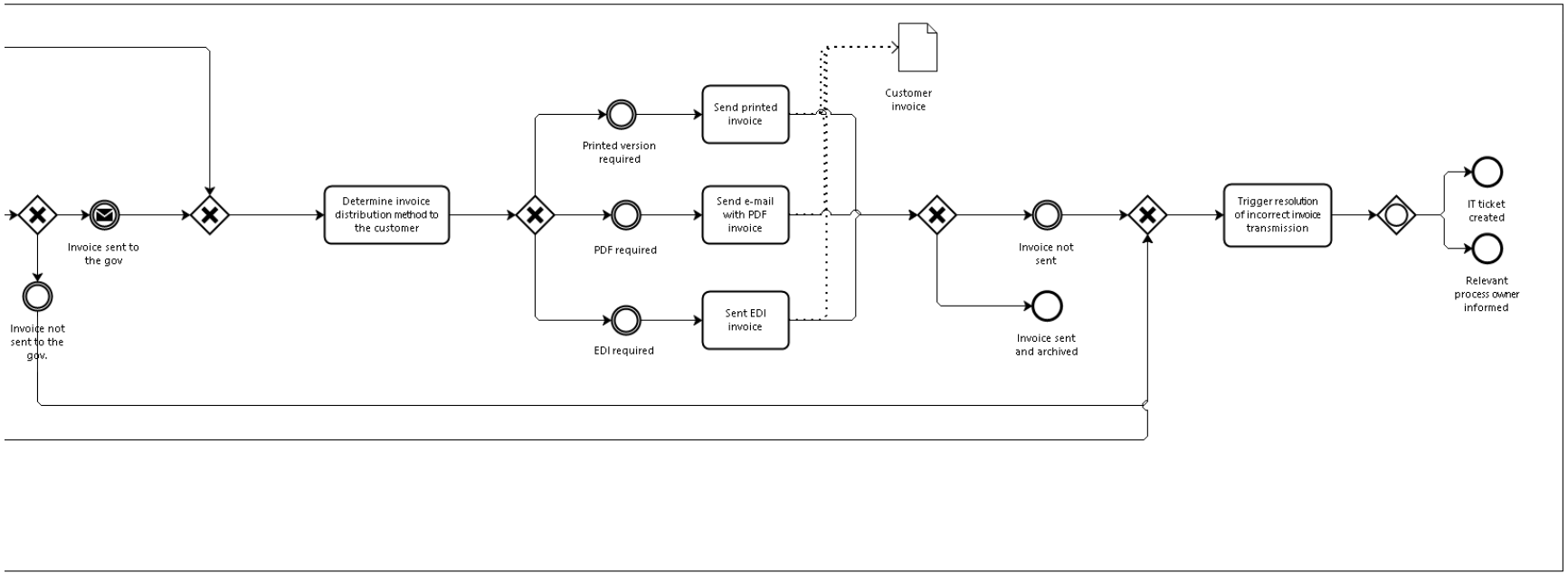


FIGURE 20. TRANSMIT BILLING DATA TO CUSTOMER (REGULATED) (PART 2)

6.3.2. APPLICATION/TECHNOLOGY ARCHITECTURE

The matrix (Figure 21) shows the link between L4 activities and application functions supporting the processes. The variants are displayed separately to match the corresponding application functions.

L4 Activity \ Application Functions	Account customer invoice/receivables	Archiving	Complaint resolution - Accounting	Customer Account Adjustments	Customer e invoicing	Customer invoicing	Customer invoice delivery	Invoice Authorization	Message creation	Message encryption	Message monitoring	Message transformation	Message transfer	Number range management	Payment card processing
1.1.1b Invoice Customer (regulated)															
1.1.1.1 Generate customer billing data				✓		✓			✓					✓	✓
1.1.1.2a Transmit billing data to customer (standard)					✓	✓	✓	✓	✓	✓					
1.1.1.2b Transmit billing data to customer (regulated)					✓	✓	✓	✓	✓	✓	✓	✓	✓		
1.1.1.3 Post receivable entries	✓					✓									✓
1.1.1.4 Recognize revenue			✓			✓									

FIGURE 21. APPLICATION FUNCTION/ L4 ACTIVITY MATRIX

The Function Allocation Diagram view shows the linkage between the variant of the L3 process, requirements, dependencies and supporting technologies in the allocation countries. Because of the limited space of the report, the diagram displays technology solutions and constraints only for two countries (see Figure 22). The variants are represented using stereotypes in angle brackets, indicating variation options with “variation option” label, optional variants with “Opt_” prefix and alternative with “Alt_” prefix.

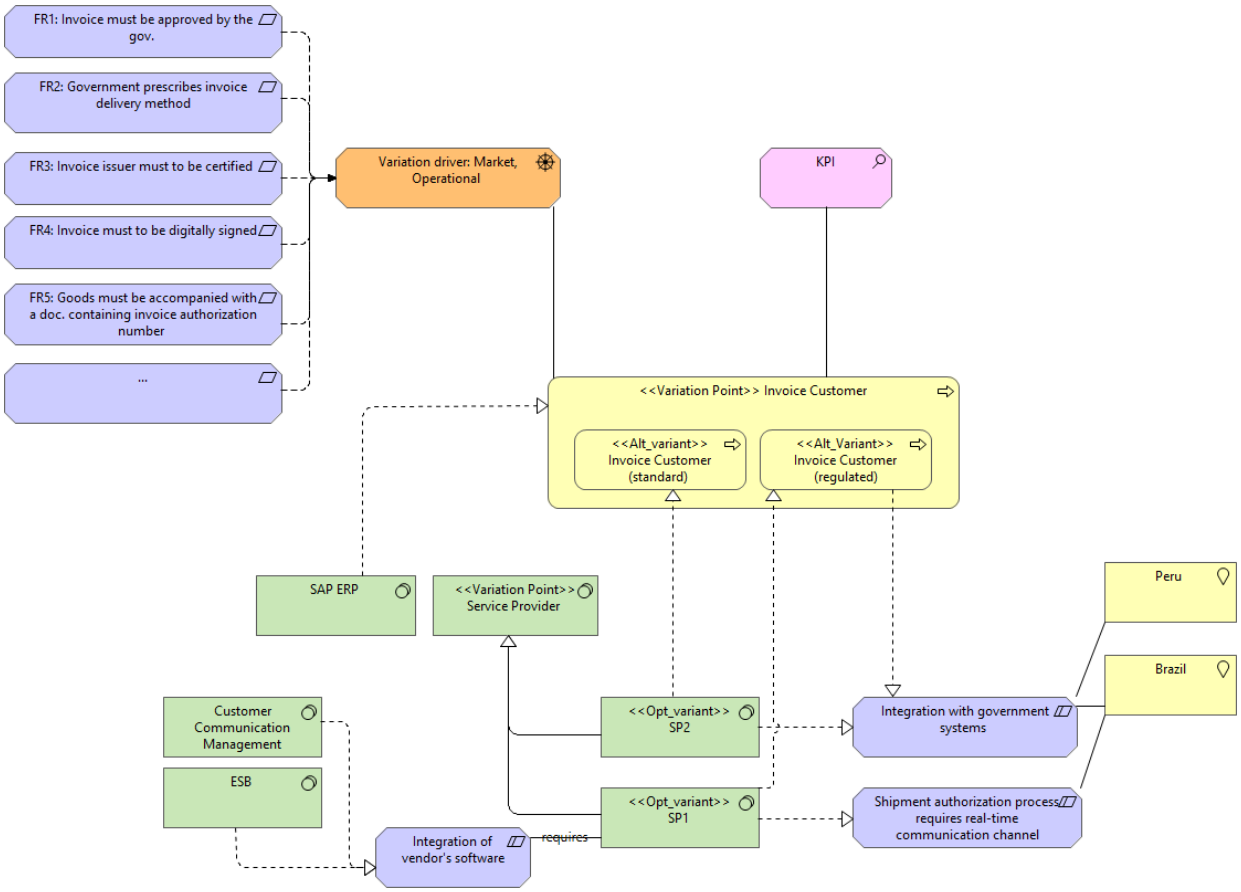


FIGURE 22. FUNCTION ALLOCATION DIAGRAM

6.4 Concluding Remarks

In this chapter we have illustrated how variability can be embedded in the models of enterprise architecture. The models were built using the concepts and approach to structuring variability described before. The models presented in this chapter are specific to the e-invoicing case. We discuss and evaluate whether the approach can be applied to other cases in the next chapter.

7. EVALUATION

The chapter discusses evaluation results of the proposed approach. Evaluation is an important aspect of design science as it demonstrations whether the designed artifact is relevant to the business and IT environment, and if it satisfies the requirements and constraints of the given problem. Evaluation of IS artifacts can be conducted in several ways. We have engaged the primary stakeholders during the solution design for iterative feedback which was incorporated in the design choices described in the earlier chapters. For the final evaluation, we decided to use *analytical method* [14] and explore the applicability/fit of the artifact into company architecture with architecture designers who were not involved during the design process.

7.1 EVALUATION QUESTIONNAIRE SETUP

The problem addressed in the research is related to a new way of handling modeling issue in the enterprise architecture. And, the proposed artifact is intended to be used and utilized by the stakeholders who are involved in the architecture design. For this reason, it is crucial to understand if the design solution meets their requirements and suggests a better way of managing variability. Also, to address the utility of the artifact, we are interested in the stakeholders' opinion about the consistency of documentation, understandability and applicability of the approach in the current setting (modeling tools and notations) of the company.

The intended evaluation approach was to conduct a focus group, involving 5-6 experts from Philips to discuss and evaluate the feasibility and applicability of the proposed approach. The main benefit of a focus group can be the added value due to the interaction of participants during the session; however, the method also creates threat to validity due to group dynamics or profile of participants. Unfortunately, because of limited availability of the stakeholders at the company and time constraints, we managed to conduct evaluation sessions separately with three experts.

The selected interviewees: the transformation architect (for business), the business architect (domain Order-to-Cash), and the enterprise architect are the key stakeholders in developing and guiding the design of the enterprise architecture at Philips. Out of the three experts, the transformation architect was involved during the design process of the solution. We have contacted the business architect and the enterprise architect mostly during the initial stages of the research to understand the company's approach and modeling standards.

It would be also beneficial to hear the overall evaluation of other stakeholders, such as the process owner and IT architect. But since they were involved during the solution development process, the choice was made on the stakeholders who were less involved in the process and also have influence in defining the overall enterprise architecture design of the company.

The evaluation session consisted of two parts; first half was dedicated to explanation of the approach and demonstration of the case study. In the second half, an open-question discussion session took place to receive feedback from experts on the quality and feasibility of the approach. The questions based on the validation aspects for software architecture documentation proposed by Clemets et al. [19]. The source was selected because of its relevance to our research topic; the focus is on assessing both functional (if the solution meets intended purpose) and quality aspects (if it's consistent, understandable, compliant with company principles and usable by the stakeholders).

The interviews were semi-structured, with same set of open-ended questions. This type of interview setup provides opportunity for the interviewer to ask more detailed questions if necessary and for the stakeholders to elaborate more on the topic. The questions used during the evaluation sessions are given below:

1. Is the approach consistent with the stakeholder community who's going to use it (enterprise, business, IT architects, business process owners and experts)?
2. Is the approach consistent itself?
    a. Are the mappings between different steps, elements of the approach provided?
    b. Are there any ambiguities, redundancy, limitations?
3. Does the documentation adhere to the standards and templates that it claims to follow?
4. Is the approach consistent with the purpose it describes?
5. Will you consider using the proposed variability extension elements in the architecture design?

7.2 Evaluation Results

The overall evaluation of the proposed solution by the experts was positive. All of them emphasized the high relevance of the problem addressed in the research. They were also satisfied with the consistency of the approach and with its feasibility in the current enterprise architecture landscape. One of the common concerns from the stakeholder's side was the lack of case studies or analysis about how applicable the approach can be for other cases of variability.

The short summary of the evaluation feedback from the interviewed stakeholders is provided below:

Expert 1: Transformation Architect

1. The approach is consistent with the stakeholder needs: the problem reflects the company's current needs and is highly relevant to the issues faced in the design of enterprise architecture. The proposed approach is in line with architecture methodology used at Philips.
2. The concepts about variability are clear, the steps for variation identification are easy to follow, and there are no redundancies. The reasoning behind the selection of variability modeling approaches is understandable but not clearly explained.
3. The documentation complies with the Philips modeling standards and is applicable with the tools used for documenting enterprise architecture.
4. The purpose was to understand how to analyze and document variability, and what are the ways to do it, and the proposed approach provides it. However, it gives one example of how it can be applied in practice.
5. The suggested variability concepts are very useful. They give the content and substance to the issues the company faces which was not clearly defined before.

Expert 2: Business Architect (Order-to-Cash)

1. The approach is consistent with the stakeholder community and is a big step further to solve problems the company has in the design, particularly unstructured and dispersed approaches how to manage variations. It shows how the company can adapt its current classification of the processes in the architecture which can be the next stage of the current architecture maturity.
2. The structure and the options are clear, and also the process how to follow variability management from its identification to modeling. It would be helpful to see more generalized overview of the pros and cons of modeling approaches which cover other cases of variability as well.
3. The documentation of the models is consistent and follows the company's modeling standards. It would be easier for the stakeholders to read the proposed process of variability management if modeled using BPMN notation [Author: the process is shown using basic Visio objects, given in Appendix G], since the same language is used for business process modeling.
4. The proposed approach is consistent with the purpose and how it deals with the e-invoicing case. But it will be interesting to have more understanding how the same approach can be applied to more customer/user-specific variability issues which can be more complex in nature than finance processes.
5. The proposed solution guides the designers to make decisions about variations and which steps to take further to embed it in the current modeling techniques. Especially, having the classification of the variation drivers and constraints that have to be considered during the design of business architecture. Classified causes of variation enables more standardized approach and reduces redundancies in the modeling which is common across domains

currently. The aspect which would help the architects more and could be the next step is to define the ways how to execute the identified variants.

Expert 3: Enterprise Architect

1. The approach is consistent with the stakeholders who are going to use it. The metamodel gives a formal way of describing the issue and that should help to easily apply it to the work. But the question is if all the stakeholders are able to recognize and read the metamodel. For example, during communication giving more concrete examples of variation drivers and textual explanation could help the business process experts or owners (who are less familiar with ArchiMate language) to understand the purpose of the elements.
2. The overall approach is useful and consistent. Having variation drivers and variation points seems redundant, as they seem to capture the same context. Instead of variation points, it would also be possible to have variation sub-drivers to capture more specific variability cause/location.
3. The approach adheres to the modeling language (ArchiMate) and the viewpoints used in the enterprise architecture. There are some differences with the current labels of the objects, but that is due to the recent switch to a new tool.
4. The approach for enterprise architecture is consistent overall but the application layer is not covered as extensively as the other layers. However, variability affects application architecture in a different way (configuration templates) and is not that important for this approach.
5. The proposed approach is useful to have a formal way of analyzing and describing variation issues for business processes and technology component support, but the application layer support is weak, so can be left out.

7.3 Concluding Remarks

The goal of design science research is utility and fitness for purpose [14]. For this reason, during the design process we involved primary stakeholders for iterative feedback and improvement of the solution. In addition to that, we conducted final assessment sessions with field experts to evaluate the feasibility of the variability management approach derived from literature study and the case study of the company.

Using a case study methodology in the design process creates an inherent threat to validity [16]. We evaluate the validity of our case based on *internal* and *external* validity aspects.

The *internal* validity aspect concerns the causal relations investigated during the case study. It is important to be aware of factors influencing the design process. The threat to internal validity of our solution was addressed by spending a considerable amount of time getting familiar with the company environment, the tools and standards they use during the design. The factors considered included the overall architecture design principles and guidelines, different modeling techniques used for creating views in the enterprise architecture, and perspectives of different stakeholders on the same process. In addition to that, we tried to consider the fact that the company plans to switch to another modeling tool and related to that, several components of enterprise architecture are going to be removed or renamed. We abstracted several concepts and focused on representing links between the concepts so that the variability documentation and modeling is not intended to be fit only for the given tool or modeling language.

External validity of an artifact is concerned with to what extent it is possible to generalize the findings. Conducting a single case study which affects only certain levels of business architecture makes it hard to validate the generalizability of the approach to other cases. We have explained above that the case scope is limited but addresses to one of the problems in architecture design related to differences in the requirements which is common to a lot of domains. This aspect was also mentioned by the experts during the evaluation sessions. We tried to reduce the threat of external validity by building our solution on established theories and techniques from the existing knowledge base. Even though the case was used for exploratory and design purposes, the conceptual model and general approach was largely inspired by literature studies which gives more scientific value to the designed artifact.

The interviews with the experts also showed that handling variability in enterprise architecture is especially important for them now because the current enterprise architecture maturity level has already achieved the stage when it's almost completely *defined*. The next step to higher maturity level is to become able to *manage* the design process across all domains through more standardized and consistent methodology. And, according to their feedback, the proposed approach gives a solid basis for adopting and standardizing it in the current architecture design.

8. CONCLUSION

In this chapter we present our findings and reflections about the research process and its outcome. The project was carried out in several stages, including the investigation, analysis, solution design and evaluation activities. In the following sections we reflect on the challenges and experiences learned throughout the project execution and discuss the how initial objectives of the research plan are met.

8.1 DISCUSSION

The goal of this research project was to address the literature gap about variability management in enterprise architecture using the electronic invoicing case study carried out at Philips. The problem of handling variability is common in the area of information systems, such as software product line engineering and business process management. The problem is also common at Philips as the company faces the challenge of managing architecture and process design because of its multi-profile business and multi-national scale.

After the initial phases of introduction and investigation of the company environment, we identified the major stakeholder concerns related to modeling variability in architecture. The main research objective was to propose an approach to handle variability in enterprise architecture caused by the differences in the requirements.

In order to develop the solution, firstly we studied available variability management techniques in literature. The literature analysis showed that the research in the field that concerns enterprise architecture is limited. However, there are numerous works which address variability handling in related areas. The study showed that the approaches in SPLE and BPM are highly relevant and can be reused for the enterprise architecture, since EA combines in itself architecture of software and applications as well as business processes. The difference in the approach that EA requires is related to its multi-functional nature and inherent complexity as it covers the whole scope of an organization.

Using the variability-related concepts from literature, we build a conceptual design of variability management approach. The design shows the concepts and their relationships with architecture elements. The conceptual model is not the full extension of the enterprise architecture metamodel. For extending the core content metamodel, we only used the built-in components of ArchiMate which can be are explicitly documented and supported by the tool (ARIS, used at Philips), such as the *variation driver, constraint, dependency* and *process variant.* Variation

points and their resolution options as well as dependency options are embedded (using stereotypes, layering, or branching points) within the models of specific architecture layers.

We used a case of electronic invoicing to explore and develop the solution design for modeling variability. As a result of several meetings and individual interviews with parties involved in defining the process requirements, we identified the country-specific fiscal regulations affecting the process of e-invoicing. Identifying requirements helped us in locating variation points and possible variation options in the processes. In addition to that, we investigated the variability in the application and technology layers. As the case study showed, different process variants may require different types of supporting technology. And, the decision on selecting software systems can be constrained by the requirements the local governments impose on the process execution.

The last part of the solution design covers the modeling of variability using three different views used at Philips. The business process variants are modeled using BPMN and L3, L4 process levels, the mapping of L4 activities and its variants to the application functions are provided using a matrix, and the linkage between the processes and support software systems are illustrated using the Function Allocation Diagram.

There are more than one variability modeling techniques used to model variability in these views due to the structural differences in modeling different layers of enterprise architecture at Philips. Business processes represent active elements of the model and are also used to support rationale behind possible execution variants. While the type of variability handled in the application/technology layers is more related to realization alternatives based on the company resources or vendors. In addition to that, the company uses two distinct modeling languages (only a subset of elements from both of them) for its EA: BPMN to model the business architecture and ArchiMate to model application/technology architecture. And, since these two languages provide different types of support for variability management, we had to consider the language-specific technique to model variability for each view.

Overall, we managed to develop an approach that addresses the main concerns of stakeholders at the company:

- Identification of variability using variation drivers, variation points and dependencies enable more structured representation and documentation of variability caused by differences in requirements;
- Application of common variability management techniques from literature to the current modeling ways, improves the modeling practice. By providing guidance on when to model process variants (L3-L4) separately and when together enables more standardization across domains;
- Extension of the Motivation layer with *constraints* and *dependencies* improves decision-making by indicating rationale between variation options and variants on the models;

- Extension of the enterprise architecture metamodel with concepts related to variability management improves the traceability between variations across layers.

It is important to notice that the proposed extension of variability management takes into account the impact of differences in the compliance requirements to the architecture layer; and, that the possible variations were captured using the architecture views that the company has already defined. Variability management could have been approached differently for other domains (i.e. more customer-specific products or service-oriented markets) where much more flexibility is allowed during the process design, and more visibility of variant instantiations in the models is necessary.

When it comes to the architecture design, it is crucial to consider that there can be multiple ways of modeling the same content each with advantages and disadvantages, but the choices have to be made based on their relevance to the stakeholder community who's going to use it. During the process design of the case of e-invoicing, we had to consider that the processes had to be understandable for the process owners or process experts who are not always familiar with modeling notations. So, we tried to keep the balance between higher efficiency in the process design (more variants in a single model) and better visibility. The smaller the number of models, the less amount of work it requires to maintain changes made to the common parts. On the other hand, multiple branching points and complex nesting is harder to comprehend.

Related to the modeling choices, the idea of using the orthogonal variability modeling approach was also considered since the OVM technique addresses the complexity of managing variant dependencies across different layers of system artifacts. The benefits it provides during modeling are the separation of concerns (domain design and variability in separate models) and uniform representation of variations from different layers. On the other hand, the technique requires a special graphic notation for modeling and tool support. Adopting a new modeling language and creating separate views for variability does not comply with architecture design principles of simplification and standardization of architecture of the company. However, for a particular complex case (could be related to a release for a specific product, or market) it would be reasonable to consider the OVM notation and approach, possibly by adapting it to one of the languages currently used at the company.

8.2 LIMITATIONS

The proposed approach has several limitations mostly due to the limited scope of the case study and time constraints:

- The requirements for the process of e-invoicing cover only the billing procedure for standard products. Philips also has other business models, including services, for which other types of fiscal regulations apply.

- The proposed variability management approach does not fully cover application layer of the enterprise architecture. One of the reasons is that the case study does not demonstrate much variation in the application layer components, and also as discussed with the enterprise architect, the variability in applications is usually configured for a run-time or instantiation of specific variants.
- Another limitation which was already discussed in Ch. 7 relates to how generalizable the solution is. We have not validated the approach for other case studies, which could give useful insights to the benefits and shortcomings of the current one.
- The case study covers only the impact of differences starting from compliance requirements in the enterprise architecture, which is also reflected in the metamodel extension. Variability due to compliance requirements is not desired in the design as it increases complexity but does not really increase the value of the business. However, variability caused by innovation or customer and market specific offers may have to be treated in a different way as the difference in the latter case is of strategic importance to the company.

8.3 Future Work

Variability management is a complex topic and is even more complicated when it is addressed in the enterprise architecture context. Our project proposes one way of approaching the issue based on the available literature and the case study. However, conducting several case studies covering different domains and variability scopes will be necessary to develop a more comprehensive approach.

Further research can be conducted for adopting orthogonal variability management approach in the enterprise architecture. Current project scope and company design constraints limited deeper research and application of the OVM technique. But the insights would be useful to see if the OVM technique can fully cover the issues with variability management on enterprise scale, or if it's possible to adapt the proposed graphic notation to ArchiMate which fully covers the modeling of all enterprise architecture layers.

Our current project studies and resolves the design phases of variability management in enterprise architecture, including the identification, documentation and modeling. Future work can be addressed to the instantiation and implementation phases, where run-time execution of specific variants with supporting information systems can be explored in more details.

REFERENCES

[1] B. Koch. (2014) E-Invoicing / E-Billing - Key stakeholders as game changers. [Online]. http://www.readsoft.com/resources/report-e-invoicing-e-billing

[2] (2014, May) The Economist. [Online]. http://www.economist.com/news/finance-and-economics/21602274-reduce-tax-fraud-governments-encourage-automated-accounts-electronic

[3] M. Mocker, J.W. Ross, and E. van Heck, "Transforming Royal Philips: Seeking Local Relevance While Leveraging Global Scale," Massachusetts Institute of Technology, Case Study 2014. [Online]. https://cisr.mit.edu/blog/documents/2014/02/27/mit_cisr_wp394_philips_mockerrossvanheck-pdf/

[4] J. van Gurp, J. Bosch, and M. Svahnberg, "On the notion of variability in software product lines," in *Working IEEE/IFIP Conference on Software Architecture, 2001. Proceedings.*, Washington, DC, 2001.

[5] F. Milani, M. Dumas, and R. Matulevičius, "Identifying and Classifying Variations in Business Processes," in *Enterprise, Business-Process and Information Systems Modeling*.: Springer Berlin Heidelberg, 2012, vol. 113, ch. `, pp. 136-150.

[6] K. Pohl, G. Böckle, and F.J.v.d.L. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*, 1st ed. New York: Springer-Verlag Berlin Heidelberg, 2005.

[7] K. Schmid and J. Isabel, "A customizable approach to full lifecycle variability management," vol. 53, pp. 259-284, 2004.

[8] M. Rosemann and W.M.P. van der Aalst, "A configurable reference modelling language," *Information Systems*, vol. 23, no. 1, pp. 1-23, 2007.

[9] M. Svahnberg, J. van Gurp, and J. Bosch, "A taxonomy of variability realization techniques: Research Articles," *Software Practice & Experience*, vol. 35, no. 8, pp. 705 - 754, 2005.

[10] M. Sinnema and S. Deelstra, "Classifying variability modeling techniques," *Information and Software Technology*, no. 49, pp. 717–739, 2007.

[11] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing Variability in Business Process Models: The Provop Approach," *Journal of Software Maintenance and Evolution*, vol. 22, no. 6-7, pp.

519-546, October 2010.

[12] L. Chen and M.A. Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology*, vol. 53, no. 4, pp. 344-362, 2011.

[13] M. Galster and P. Avgeriou, "An industrial case study on variability handling in large enterprise software systems," *Information and Software Technology*, vol. 60, pp. 16-31, 2015.

[14] A. Hevner, S. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75-105, 2004. [Online]. http://dl.acm.org/citation.cfm?id=2017217

[15] B.A. Kitchenham, "Guidelines for performing Systematic Literature Reviews in Software Engineering," EBSE-2007-01 2007. [Online]. http://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

[16] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2009.

[17] R.J. Wieringa, *Desgn Science Methodology for Information Systems and Software Engineering*.: Springer-Verlag Berlin Heidelberg, 2014.

[18] The Open Group. (2013, December) The Open Group. [Online]. http://pubs.opengroup.org/architecture/archimate2-doc/

[19] P. Clements, F. Bachmann, and L. Bass, *Documenting software architectures : views and beyond*, The SEI Series in Software Engineering ed.: London: Addison-Wesley, 2003.

[20] P. Grefen, *Business Information Systems Architecture*, 2014th ed. The Netherlands, 2014.

[21] The Open Group. (2011) Open Group Standard. [Online]. http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[22] J.A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 38, no. 2, pp. 454 - 470, 1999. [Online]. http://search.proquest.com/docview/222422073?accountid=27128

[23] M. Lankhorst, *Enterprise Architecture at Work*, 3rd ed.: Springer-Verlag Berlin Heidelberg, 2013.

[24] C. Atkinson, R. Gerbig, and M. Fritzsche, "A multi-level approach to modeling language extension in the Enterprise Systems Domain," *Information Systems*, vol. 54, pp. 289-307,

December 2015.

[25] Object Management Group, *OMG Unified Modeling Language (OMG UML) Version 2.5*., 2015.

[26] R. Lagerström, P. Johnson, and D. Höök, "Architecture analysis of enterprise systems modifiability – Models, analysis," *The Journal of Systems and Software*, no. 83, pp. 1387–1403, 2010.

[27] S. Mahdavi-Hezavehi, M. Galster, and P. Avgeriou, "Variability in quality attributes of service-based software systems: A systematic literature review," *Information and Software Technology*, vol. 55, no. 2, pp. 320–343, February 2013.

[28] R. Kazhamiakin, S. Benbernou, and L. Baresi, "Adaptation of Service-Based Systems," in *Service Research Challenges and Solutions for the Future Internet, Lecture Notes in Computer Science*.: Springer Berlin Heidelberg, 2010, vol. 6500, pp. 117-156.

[29] M. Galster, P. Avgeriou, and D. Tofan, "Constraints for the design of variability-intensive service-oriented reference architectures - an industrial case study," *Information and Software Technology*, vol. 55, pp. 428-441, 2013.

[30] M. La Rosa, W. van der Aalst, M. Dumas, and A.H.M. ter Hofstede, "Questionnaire-based variability modeling for system configuration," *Software & Systems Modeling*, vol. 8, no. 2, pp. 251-274, April 2009.

[31] H.M.H. Hegge and J.C. Wortmann, "Generic bill-of-material: a new product model," *International Journal of Production Economics*, vol. 23, no. 1-3, pp. 117-128, October 1991.

[32] F. Erens and K. Verhulst, "Architectures for product families," *Computers in Industry*, vol. 33, no. 2-3, pp. 165-178, 1997.

[33] C. Ayora, V. Torres, B. Weber, M. Reichert, and V. Pelechano, "VIVACE: A framework for the systematic evaluation of variability support in process-aware information systems," *Information and Software Technology*, vol. 57, pp. 248-276, 2015.

[34] K.C. Kang, S. G. Cohen, J.A. Hess, W.E. Novak, and A.S. Peterson, "Feature-Oriented Domain Analysis (FODA), Feasibility Study," Software Engineering Institute, Pittsburgh, PA, CMU/SEI-90-TR-21, 1990. [Online]. http://www.sei.cmu.edu/reports/90tr021.pdf

[35] G. Gröner, M. Boskovic, F.S. Parreiras, and D. Gasevic, "Modeling and validation of business process families," *Information Systems*, vol. 38, no. 5, pp. 709-726, 2013.

[36] C.-ai Sun, R. Rossing, M. Sinnema, P. Bulanov, and M. Aiello, "Modeling and managing the variability of Web service-based systems," *The Journal of Systems and Software*, vol. 83, pp. 502-516, 2010.

[37] M. Sinnema, S. Deelstra, and P. Hoekstra, "The COVAMOF derivation process," in *InProceedings of the 9th international conference on Reuse of Off-the-Shelf Components (ICSR'06*, 2006, pp. 101-114.

[38] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch, "COVAMOF: A Framework for Modeling Variability in Software Product Families," in *Software Product Lines, Lecture Notes in Computer Science*. Boston: Springer Berlin Heidelberg, 2004, vol. 3154, pp. 197-213.

[39] A.K. Thurimella and B. Bruegge, "Issue-based Variability Management," *Issue-based variability management*, vol. 54, pp. 933-950, March 2012.

[40] L. Chen and M.A. Babar, "A systematic review of evaluation of variability management approaches in software product lines," *Information and Software Technology*, vol. 53, no. 4, pp. 344-362, 2011.

[41] F. Puhlmann, A. Scheiders, J. Weiland, and M. Weske, "Variability Mechanisms for Process Models," BMBF-Project, Technical Report 2006.

[42] M. Razavian and R. Khosravi, "Modeling Variability in Business Process Models Using UML," in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, 2008, pp. 82-87.

[43] F. Milani, M. Dumas, and R. Matulevičius, "Decomposition Driven Consolidation of Process Models," in *Advanced Information Systems Engineering, 25th International Conference, CAiSE 2013*, Valencia, 2013, pp. 193-207.

[44] Official Journal of the European Union. (2014) DIRECTIVE 2014/55/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 16 April 2014 on electronic invoicing in public procurement. [Online]. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014L0055

[45] S. Kreuzer, A. Eckhardt, S. Bernius, and J. Kronung, "A Unified View of Electronic Invoicing Adoption: Developing a Meta-Model on the Governmental Level," in *HICSS '13 Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 1943-1952.

[46] J. A. Zachman. (2009-2011) Zachman International. [Online]. http://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution

[47] M.M. Lankhorst, "Enterprise architecture modelling—the issue of integration," *Advanced Engineering Informatics*, vol. 18, no. 4, pp. 205–216, 2004.

[48] M. von Rosing, H. von Scheel, and A.-W. Scheer, *The Complete Business Process Handbook*.: Elsevier Inc. , 2014.

[49] D. Greefhorst and E. Proper, *Architecture Principles: The Cornerstones of Enterprise Architecture*, 1st ed.: Springer-Verlag Berlin Heidelberg, 2011.

[50] R.J. Wieringa, "Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology," in *DESRIST '09*, 2009.

[51] M. Rosenmüller, N. Siegmund, T. Thüm, and G. Saake, "Multi-Dimensional Variability Modeling," in *VaMoS '11 Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems*, 2011, pp. 11-20.

[52] S. Thiel and A. Hein, "Systematic Integration of Variability into Product Line Architecture Design," in *Proceedings of the Second International Conference on Software Product Lines, SPLC 2*, 2002, pp. 130-153.

[53] "An Electronic Invoicing System," in *Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on*, Graz, 2011, pp. 149-156.

[54] E. Katz. (2012, April) SAP Community Network. [Online]. http://scn.sap.com/docs/DOC-26192

[55] D. Batory, "Feature Models, Grammars, and Propositional Formulas," in *SPLC'05 Proceedings of the 9th international conference on Software Product Lines*, 2005, pp. 7-20.

[56] K.C. Kang et al., "FORM: A feature-oriented reuse method with domain-specific reference architectures," *Annals of Software Engineering*, vol. 5, no. 1, pp. 143-168, 1998. [Online]. http://link.springer.com/content/pdf/10.1023%2FA%3A1018980625587.pdf

[57] M. La Rosa, M. Dumas, A.H.M. ter Hofstede , and J. Mendling, "Configurable multi-perspective business process models," *Information Systems*, vol. 36, no. 2, pp. 313-340, April 2011.

APPENDIX A: ZACHMAN FRAMEWORK

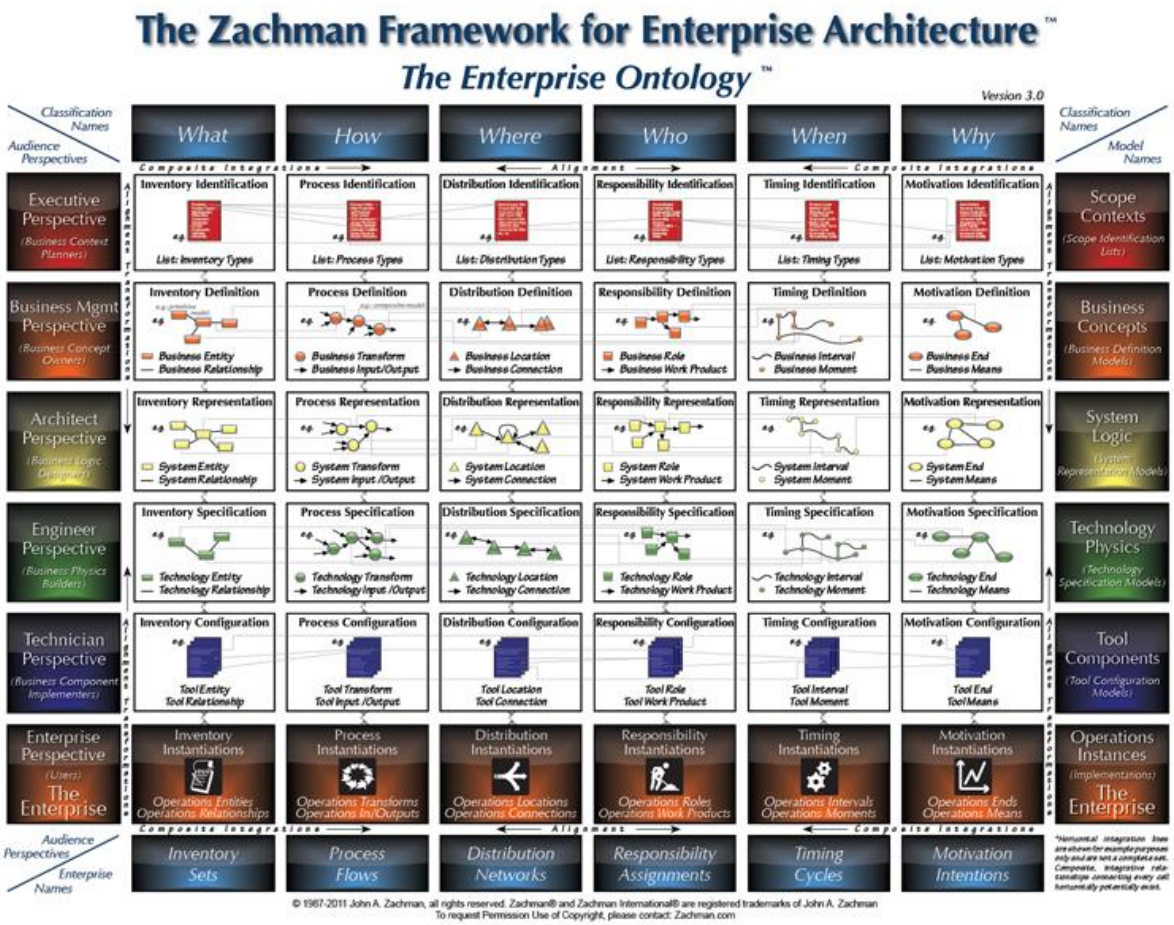


FIGURE 23. THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE. VERSION 3.0 (2011) [46]

APPENDIX B: ENTERPRISE ARCHITECTURE

A. TOGAF

TOGAF is an open standard, developed by the Open Group, which provides a detailed method and a set of supporting tools for developing enterprise architecture in organizations [21].

The enterprise architecture methodology considers not only current requirements, but future needs of the organization, and it addresses the needs of different stakeholders involved in its development. TOGAF reflects the structure and the content of architecture resources and capabilities within the enterprise. The complete framework covers Architecture Development Method, Architecture Content Framework (Figure 24), Enterprise Continuum (a model for structuring virtual repositories) and Tools and Reference models, supported by guidelines and techniques for each component.

One of the important parts of TOGAF methodology is the Enterprise Continuum. The Enterprise Continuum refers to the methodology for defining overall structure of the virtual repository of EA which enables classifying architecture and solution artifacts. The main objective of the Continuum is to support re-using of architecture assets, such as models, patterns, catalogues etc. These assets need to be structured and classified in a way which enables exploitation of commonalities and avoiding unnecessary repetitions.

In order to achieve the re-use, the architecture assets are represented as building blocks in TOGAF. The building blocks may represent the deliverables of each phase, including the reference models from the architecture layers, patterns, etc.

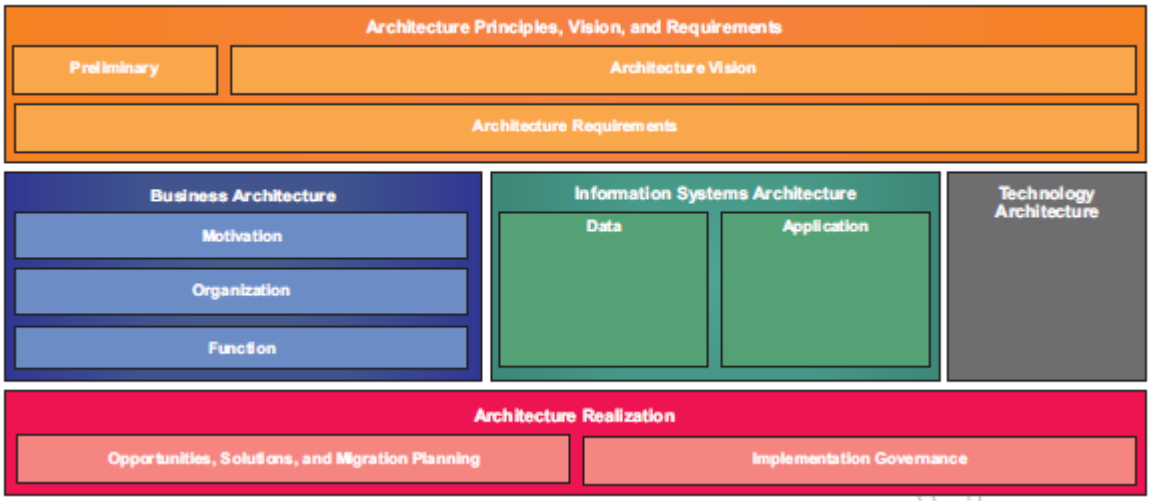


FIGURE 24. CONTENT FRAMEWORK WITH ADM PHASES

B. ARCHIMATE

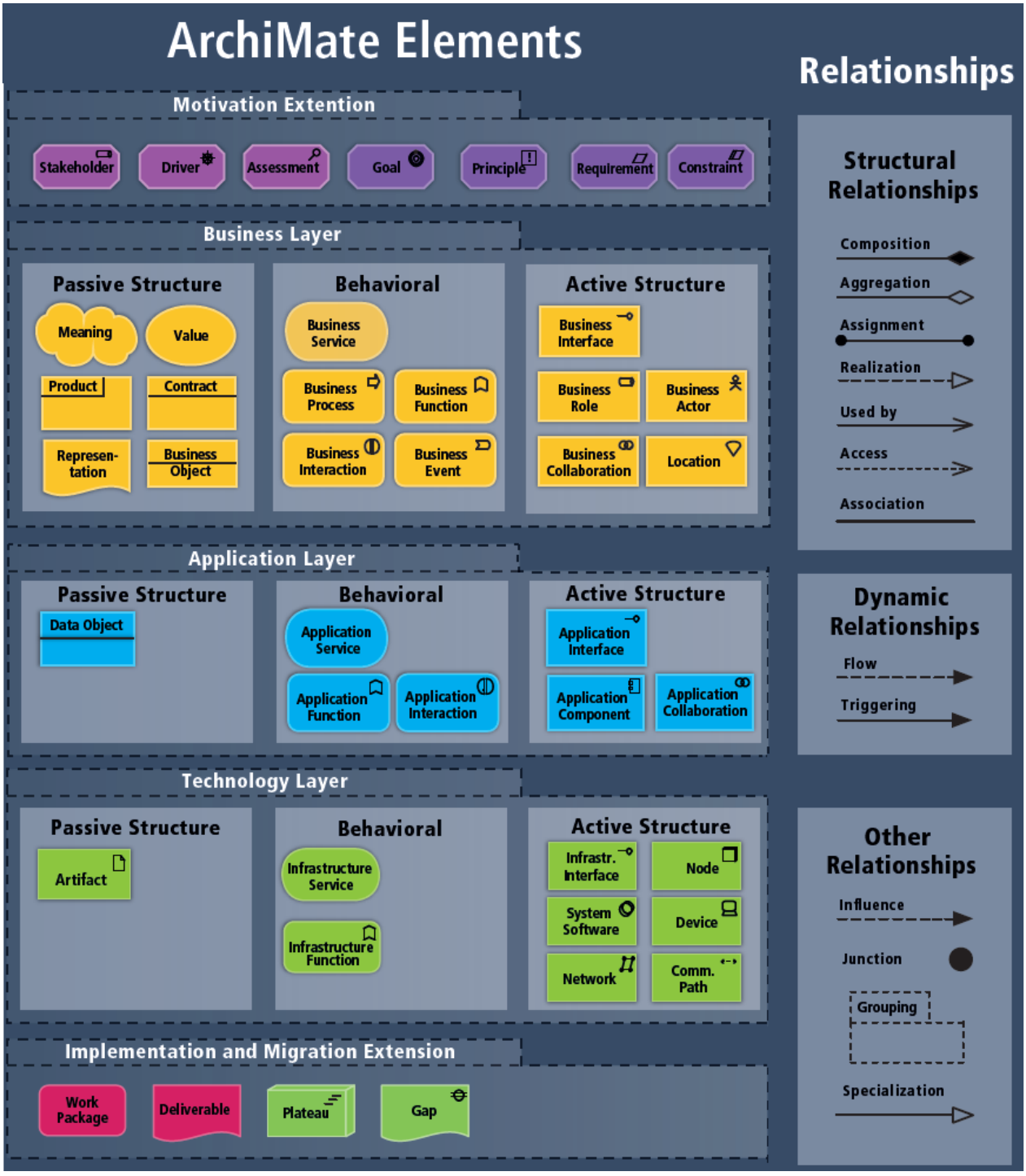
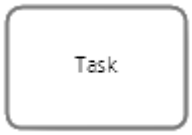
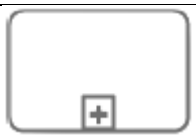
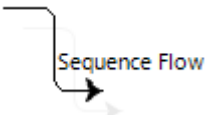


FIGURE 25. ARCHIMATE 2.1 QUICK SHEET

C. BPMN

The business process models presented in the report were created using Bizagi Process Modeler. The software follows BPMN notation for process modeling. A subset of BPMN notation used for modeling is given below:

Element	Description	Notation
Task	Is an atomic Activity within a Process flow. It is used when the work in the Process cannot be broken down to a finer level of detail.	
Sub-process	Is an Activity which internal details have been modeled using activities, gateways, Events, and sequence flows. The elements have a thin border.	
Start Event	Indicates where a particular Process starts. It does not have any particular behavior.	
Message Start Event	Is used when a message arrives from a participant and triggers the start of the Process.	
Timer Start Event	Is used when the start of a Process occurs on a specific date or cycle time (e.g., every Friday)	
Intermediate Event	Indicates where something happens somewhere between the start and end of a Process. It will affect the flow of the Process, but will not start or (directly) terminate the Process.	
Message Event	Indicates that a message can be sent or received. If a Process is waiting for a message and it is caught the Process will continue its flow. A catch Message Event waits for a message to arrive and once the message has been received, the Process will continue. The Event marker in this instance will be filled. A throw Message Event sends a message to an external participant. The unfilled Event marker is allocated to the throw message.	 
Timer Event	Indicates a delay within the Process. This type of Event can be used within the sequential flow indicating a waiting time between activities.	
End Event	Indicates when the Process ends.	
Message End	Indicates that a message is sent when the flow has ended.	

Exclusive Gateway	As Divergence: It is used to create alternative paths within the Process, but only one is chosen. As Convergence: It is used to merge alternative paths.	
Parallel Gateway	As Divergence: is used to create alternative paths without checking any conditions. As Convergence: is used to merge alternative paths, the gateways waits for all incoming flows before it continues.	
Inclusive Gateway	As Divergence: represents a branching point where alternatives are based on conditional expressions. The TRUE evaluation of one condition does not exclude the evaluation of the other conditions. All evaluations of a TRUE condition will be traversed by a token. As Convergence: is used to merge a combination of alternative and parallel paths.	
Data Objects	Provides information about how documents, data and other objects are used and updated during the Process.	
Annotation	Is a mechanism for a modeler to provide additional information for the reader of a BPMN Diagram.	
Pool	A Pool is a container of a single Process (contains the sequence flows between activities). A Process is fully contained within the Pool. There is always at least one Pool.	
Lane	Is a sub-partition within the Process. Lanes are used to differentiate elements as internal roles, position, department, etc. They represent functional areas that may be responsible for tasks.	
Sequence Flow	A Sequence Flow is used to show the order that Activities will be performed in the Process.	
Association	Is used to associate information and Artifacts with Flow Objects. It also shows the activities used to compensate for an activity.	
Message Flow	Is used to show the flow of messages between two entities that are prepared to send and receive them.	

APPENDIX C: VARIABILITY MANAGEMENT TECHNIQUES

a. VARIABILITY MANAGEMENT LIFECYCLE

TABLE 5. VARIABILITY MANAGEMENT LIFECYCLE PHASES IN SPLE

Literature Source	Variability Management Phases					
	Domain Engineering			Application Engineering		
K. Schmid and J. Isabel (2004)	Identify	Model	Store	Resolve	Instantiate	Change
M. Svahnberg, J. v. Gurp and J. Bosch (2005)	Identify	Constrain		Implement		Manage
M. Sinnema and S. Deelstra (2007)	Introduction	Use				Evolution
M. Galster and P. Avgeriou (2015)	Describe			Resolve		

b. VARIABILITY MODELING TECHNIQUES

Characteristics	Approaches					
	VSL	ConIPF	COVAMOF	CBFM	Koalish	Pure::Variants
Model						
Choices	Choice model (Variabilities and Variation Points)	Multiplicity in structure (Feature and Artifact tree)	Choice model (Variation Points)	Multiplicity in structure (Feature tree)	Multiplicity in structure (Koala components)	Multiplicity in structure (Feature and Family model)
Products	Decision Model	Decision Model	Decision Model	Decision Model	Stand-alone entity	Decision Model
Abstraction	Multiple Layers	Hierarchy and Multiple Layers	Hierarchy and Multiple Layers	Hierarchy	Hierarchy	Hierarchy and Multiple Layers
Formal Constraints	Include/exclude (technique specific)	Algebraic expressions (technique specific)	Algebraic expressions (technique specific)	Algebraic expressions (XPath)	Algebraic expressions (technique specific)	Algebraic expressions (Prolog)
Quality Attributes	Not modeled	Mentioned (Context entities, restrictions)	Modeled (Dependency entities)	Mentioned (Feature entities)	Not modeled	Not modeled
Support for Incompleteness and Imprecision	Requires precise specifications	Aimed at completeness, requires precise specifications	Supports both	Requires precise specifications	Requires precise specifications	Aimed at completeness, requires precise specifications
Tooling						
Multiple Views	Not supported by tooling	2 (modeling and configuration)	8 (modeling and configuration)	2 (modeling and configuration)	1 (configuration)	4 (modeling and configuration)
Active Specification	Not supported by tooling	Proactive	Proactive	Reactive specification of constraints	None	Proactive is possible, no consistency checking
Configuration Guidance	Not supported by tooling	Static and dynamic	Static and dynamic	None	None	None
Inference Engine	Not supported by tooling	Consistency, decision making, prevention	Consistency, decision making	Consistency	Consistency, decision making, prevention	Consistency, decision making
Effectuation	Operations on XML documents and Jscript sources	None provided	File-based operations with special support for C, C++, and C#	None provided	Generation of C applications	File-based operations with special support for C and C++

FIGURE 26. CLASSIFICATION OF VARIABILITY MODELING TECHNIQUES [10]

c. VARIABILITY REALIZATION TECHNIQUES

Section	Name	Introduction Time	Open for Adding Variants	Collection of Variants	Binding Times	Functionality for Binding
3.1	Architecture Reorganization	Architecture Design	Architecture Design	Implicit	Product Architecture Derivation	External
3.2	Variant Architecture Component	Architecture Design	Architecture Design Detailed Design	Implicit	Product Architecture Derivation	External
3.3	Optional Architecture Component	Architecture Design	Architecture Design	Implicit	Product Architecture Derivation	External
3.4	Binary Replacement - Linker Directives	Architecture Design	Linking	Implicit or Explicit	Linking	External or Internal
3.5	Binary Replacement - Physical	Architecture Design	After Compilation	Implicit	Before Run-time	External
3.6	Infrastructure-Centered Architecture	Architecture Design	Architecture Design Linking Run-time	Implicit or Explicit	Compilation Run-time	Internal
3.7	Variant Component Specializations	Detailed Design	Detailed Design	Implicit	Product Architecture Derivation	External
3.8	Optional Component Specializations	Detailed Design	Detailed Design	Implicit	Product Architecture Derivation	External
3.9	Run-time Variant Component Specializations	Detailed Design	Detailed Design	Explicit	Run-time	Internal
3.10	Variant Component Implementations	Architecture Design	Detailed Design	Explicit	Run-time	Internal
3.11	Condition on Constant	Implementation	Implementation	Implicit	Compilation	Internal or External
3.12	Condition on Variable	Implementation	Implementation	Implicit or Explicit	Run-time	Internal
3.13	Code Fragment Superimposition	Compilation	Compilation	Implicit	Compilation or Run-time	External

FIGURE 27. VARIABILITY REALIZATION TECHNIQUES [9]

APPENDIX D: SINGLE-ARTIFACT APPROACHES

a. CONFIGURABLE EPC

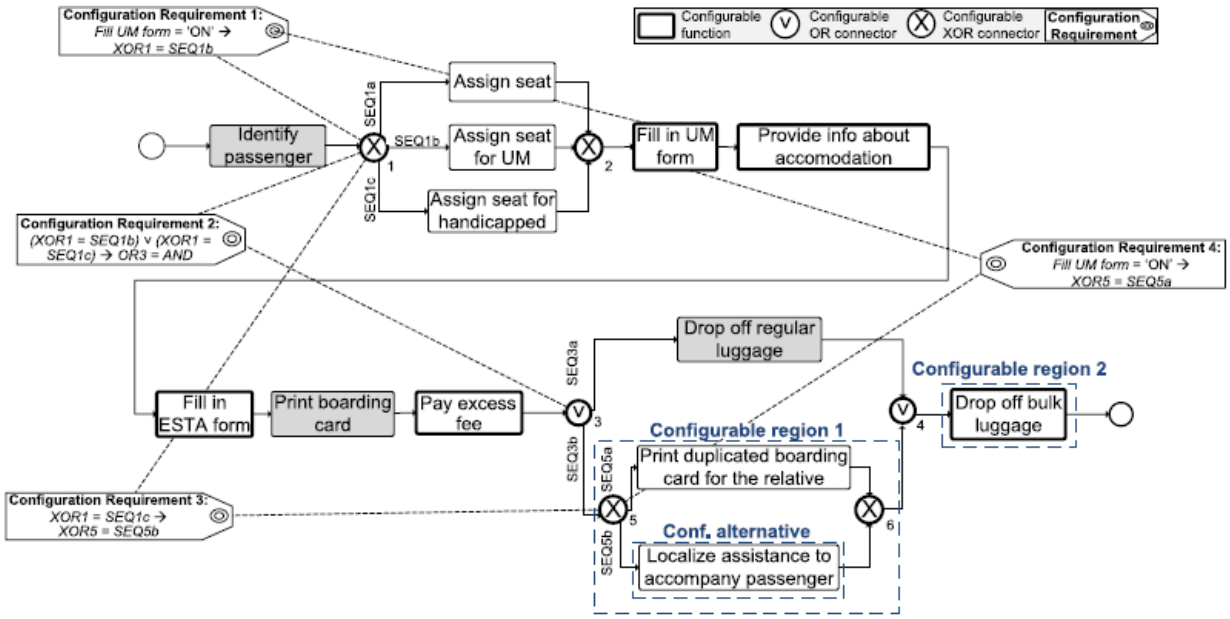


FIGURE 28. AN EXAMPLE OF CONFIGURABLE – EPC [33]

b. PESOA (WITH STEREOTYPES)

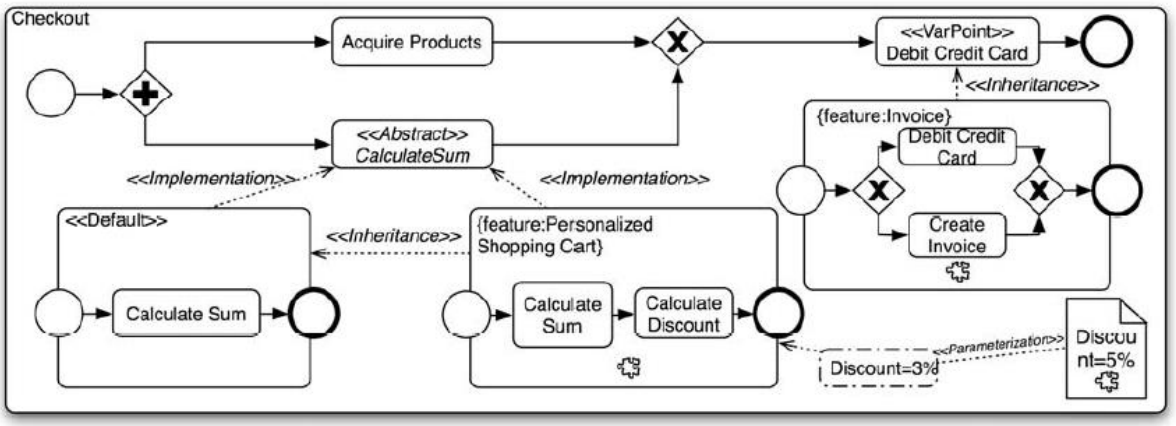


FIGURE 29. AN EXAMPLE OF PROCESS MODELING IN PESOA [41]

APPENDIX E: MULTI-ARTIFACT APPROACHES

a. ORTHOGONAL VARIABILITY MODEL

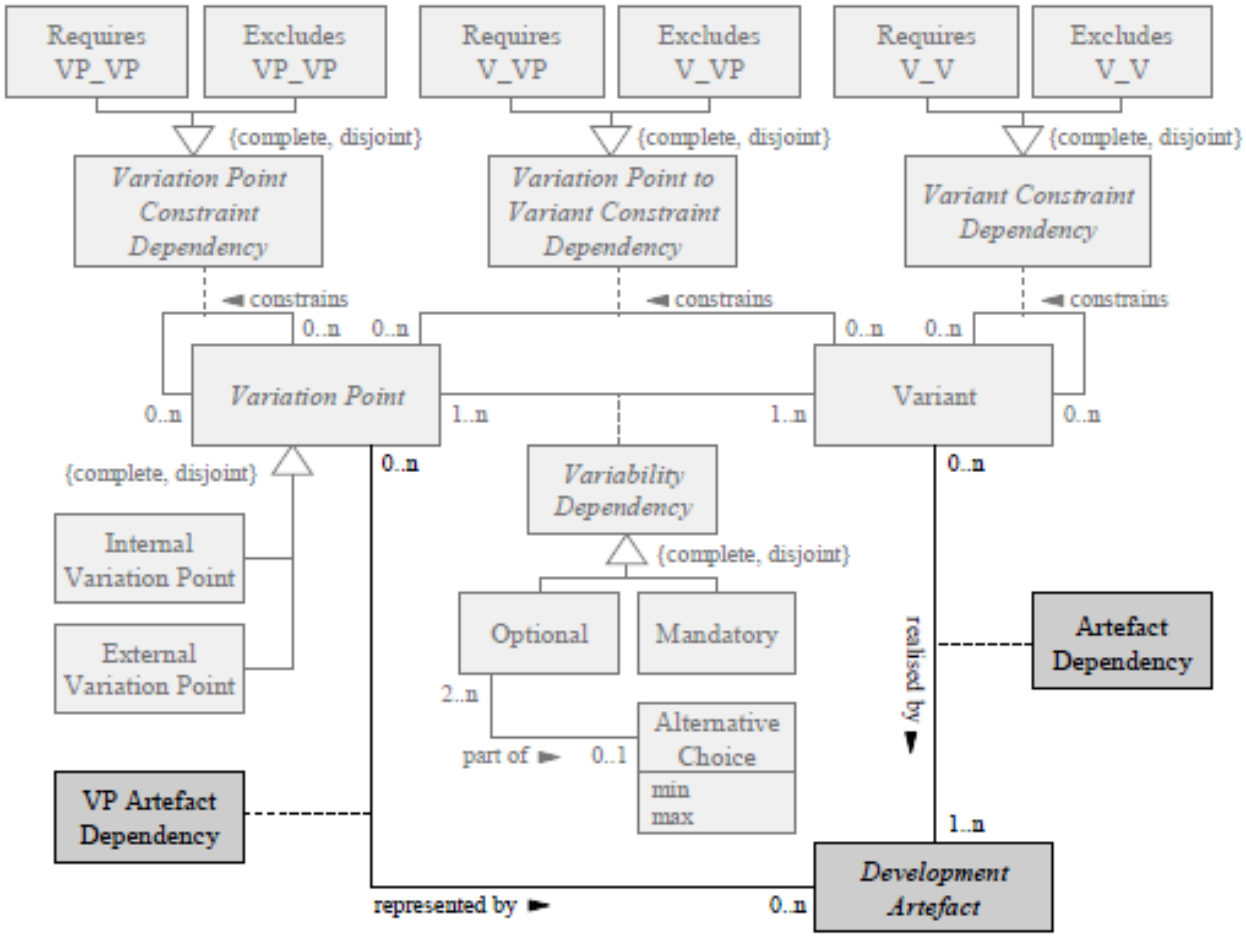


FIGURE 30. OVM GRAPHICAL NOTATION

b. THE PROVOP APPROACH

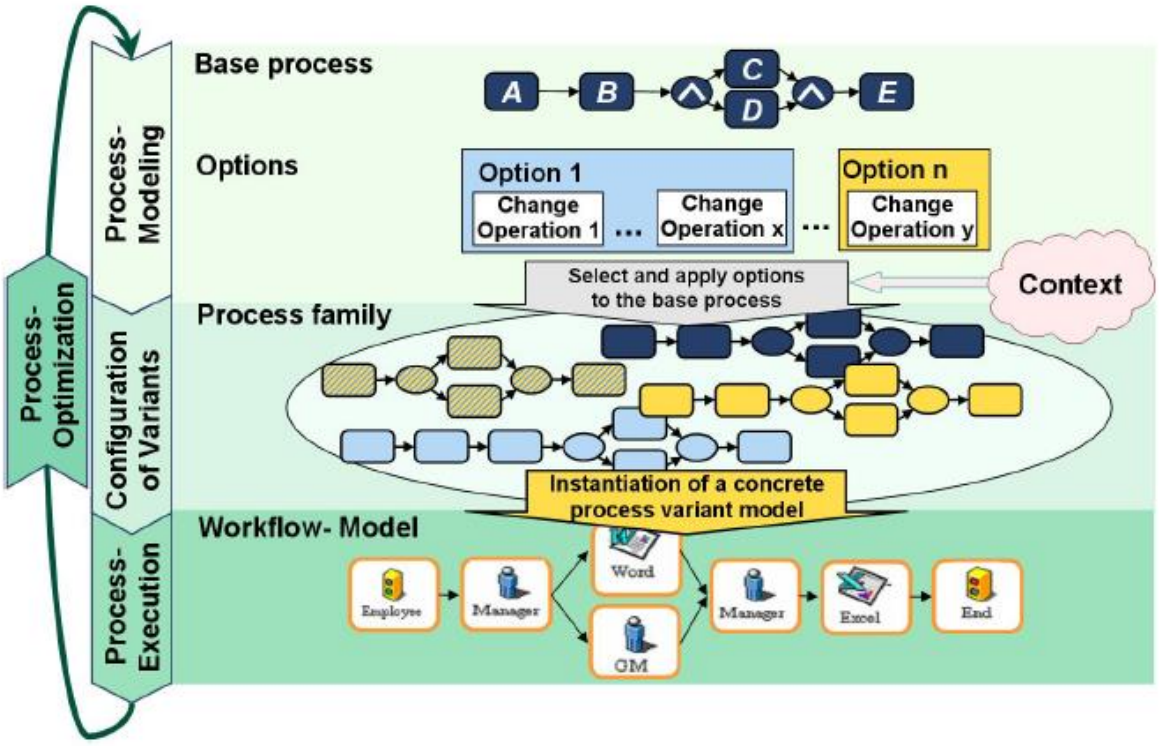


FIGURE 31. THE PROVOP APPROACH

APPENDIX F: INTERVIEWS

The goal of the interview was to identify what is the current approach to variability management at Philips and what are the concerns related to modeling the differences in the artifacts. The interview was setup using open-ended questions, which gave us, the interviewer, and the respondents more flexibility to explore the subject in more details. The questions are presented below:

1. How do you define variability in business processes (compliance requirements, IT solutions)?
2. How is the variability represented/modeled in the architecture?
3. What are the constraints considered to be considered when modeling variability?
4. What are the issues related to the current way of managing variability?
5. How can the current variability management approach be improved?

Interviewee 1: Business Architect (Finance)

1. How do you define variability in business processes?
   Different ways of achieving the same goal through reusable components.
2. How is the variability represented/modeled in the architecture?
   Archetypes - high level processes (Operating models), Scenarios - low level processes (execution variants).
3. What are the constraints considered to be considered when modeling variability?
   Variability modeling preferred on higher levels of business processes, Standardization.
4. What are the issues related to the current way of managing variability?
   No structured way of identifying and storing reusable elements.
5. How can the current variability management approach be improved?
   A way to identify patterns and catalogue them.

Interviewee 2: Business Architect (Order-to-Cash)

1. How do you define variability in business processes?
   Differences in the business models and in the execution of the processes.
2. How is the variability represented/modeled in the architecture?
   Archetypes - high level processes (operating models), Scenarios - low level processes (execution variants).
3. What are the constraints considered to be considered when modeling variability?

Variability modeling preferred on higher levels of business processes, harmonization, technical (tool) limitations.

4. What are the issues related to the current way of managing variability?
   Complexity of the models with high variation, Hard to maintain standard way of modeling across domains.
5. How can the current variability management approach be improved?
   More standardization and enforcement of constraints across domains.

Interviewee 3: IT Architect (Finance)

1. How do you define variability in business processes?
   Variations of the same process in terms of process flows or solutions.
2. How is the variability represented/modeled in the architecture?
   Multi-model approach.
3. What are the constraints considered to be considered when modeling variability?
   Applied architecture modeling techniques, Local IT requirements.
4. What are the issues related to the current way of managing variability?
   Complexity related to analysis and maintenance of current models, Lack of structured documentation.
5. How can the current variability management approach be improved?
   A way of identifying and defining patterns/reusable components for modeling.

Interviewee 4: Process Expert Physical Distribution (Order-to-Cash)

1. How do you define variability in business processes?
   Differences in the local and regional requirements.
2. How is the variability represented/modeled in the architecture?
   Catalogue of compliance requirements.
3. What are the constraints considered to be considered when modeling variability?
   Country-specific requirements, locally defined processes (customs management).
4. What are the issues related to the current way of managing variability?
   Complexity of the defining the processes due to high variation, adapting to frequently changing regulations.
5. How can the current variability management approach be improved?
   Clear documentation of the country-specific requirements and process flows.

Interviewee 5: Senior Business Analyst (IT architecture for Order-to-Cash, Compliance)

1. How do you define variability in business processes?
   Differences in the local and regional requirements.
2. How is the variability represented/modeled in the architecture?
   Catalogue of compliance requirements, mapping the requirement to business and IT architecture.
3. What are the constraints considered to be considered when modeling variability?
   Applied architecture modeling techniques, Local IT requirements.
4. What are the issues related to the current way of managing variability?
   Lack of structured documentation.
5. How can the current variability management approach be improved?
   Clear documentation of the country-specific requirements and process flows.

APPENDIX G: VARIABILITY MANAGEMENT METHOD

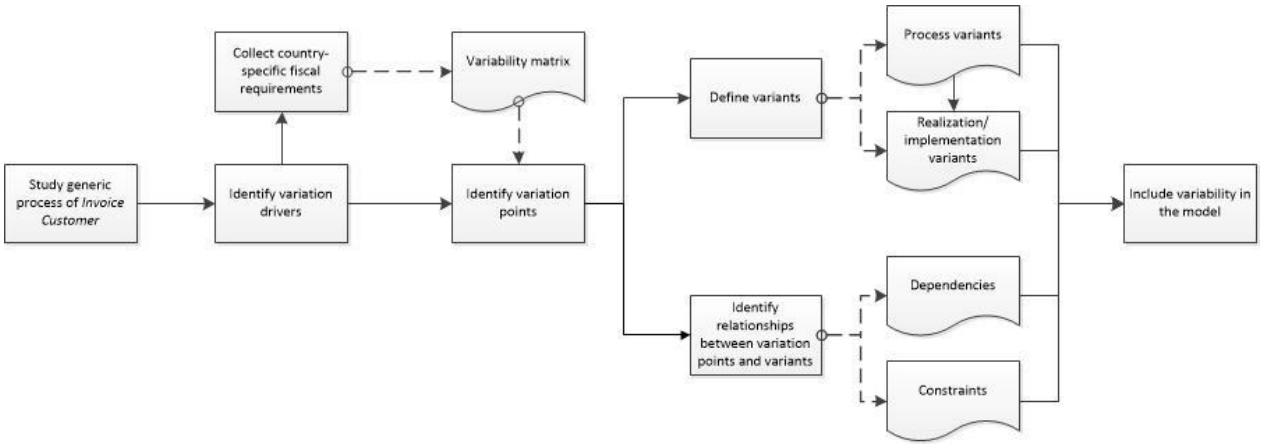


FIGURE 32. STEPS TO IDENTIFY AND INCORPORATE VARIABILITY IN THE ARCHITECTURE DESIGN

The model presented in Figure 32 represents the steps we followed during the design process. The framework combines variability identification, representation and modeling phases and can be used as guidelines to manage variability in enterprise architecture.

A business process or business method is a collection of related, structured activities or tasks that produce a specific service or product for a particular customer or customers. Variability from business process perspective can be classified as following [33]:

- *Functional perspective* specifies the decomposition of business process into atomic and complex activities. The variability in process activities based on their function is handled by modeling them as different layers of abstraction at Philips, as described before.
  Case: "Invoice customer" is a L3 process, while "Standard invoicing" and "Electronic invoicing" are lower L4 processes with more detailed activity flows.

- *Behavioral perspective* describes the control flow of activities and constraints for their execution.
  Case: While the normal procedure is to generate and validate invoices after the goods have been delivered, some tax authorities require a validate invoice accompanying the goods. This regulation requires validation procedure earlier in the process, creating a different variant of the process flow.

- *Organizational perspective* captures different accountabilities and roles responsible for each work unit in the process.
  Case: Depending on the regulation and service implementation choice, the communication with the government for invoice validation can be realized by Philips itself or via third party service provider.

- *Informational perspective* cover the data and data flow required as the process input or output.
  Case: The difference in the data objects for the case studies in this project, is reflected on L4, where the procedure of issuing and validating invoice is different paper-based and different for e-invoice. However, the difference in the e-invoice layout, which requires higher details, can be reflected on the application layer of the enterprise architecture as it affects the implementation and transferring of the data.

- *Temporal perspective* captures the constraints that affect the time of a start or an end of an activity, an event or a business object.
  Case: Time required for invoice approval is essential in cases when the goods cannot be shipped without a valid invoice. For this reason, real-time sending and receiving of an e-invoice status message needs to be reflected on the process design.

- *Operational perspective* covers implementation perspective of the processes and the differences between the services provided by different applications.
  Case: There are a few options for realizing the e-invoice validation process depending on whether Philips implements it or if it's outsourced. In addition to that, there is a difference in the services provided by certified brokers locally.

APPENDIX H: PROCESS MODEL VARIANTS

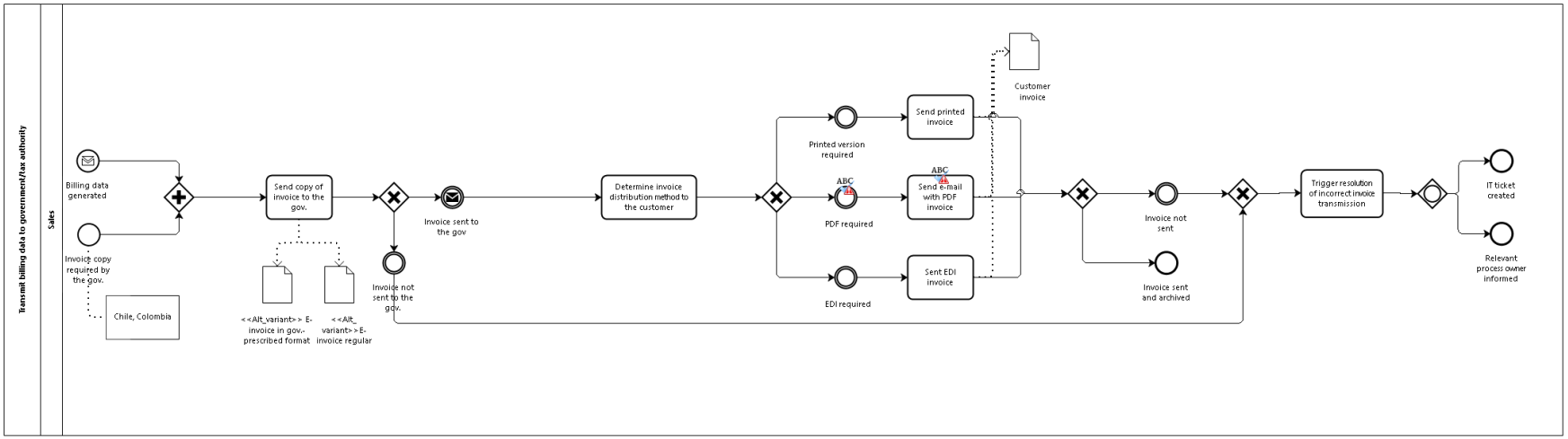


FIGURE 33. TRANSMIT BILLING DATA TO CUSTOMER (DOC. PRESENTMENT)

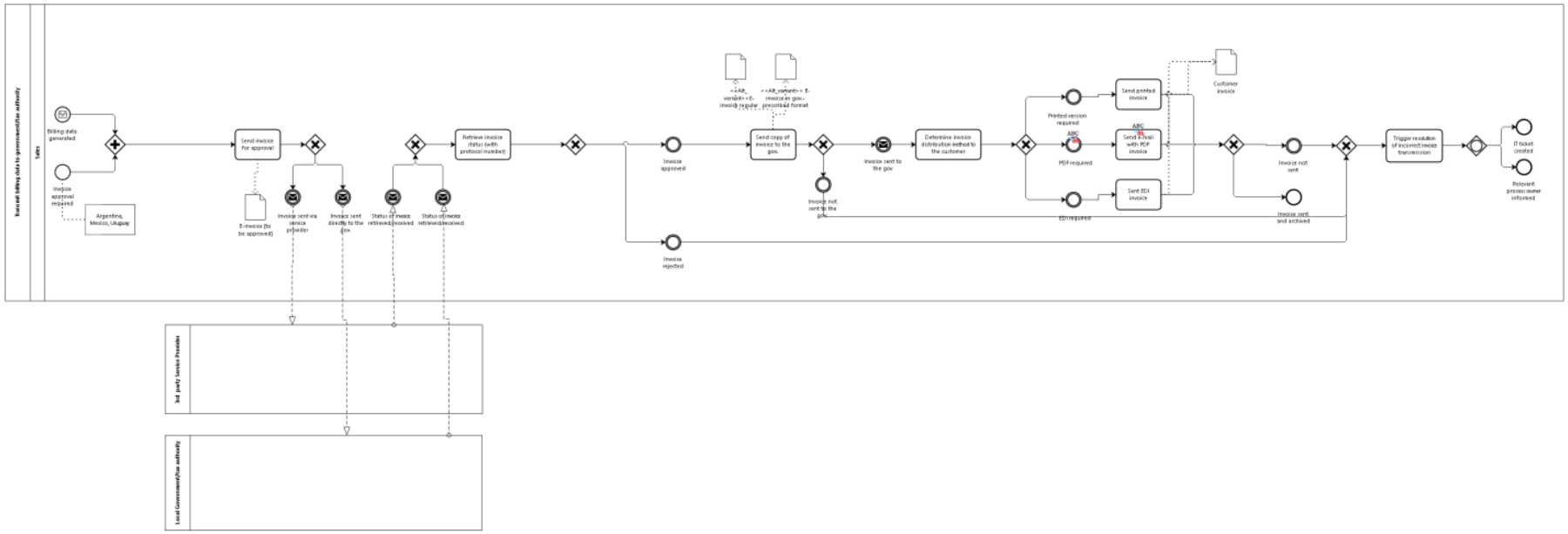


FIGURE 34. TRANSMIT BILLING DATA TO CUSTOMER (INVOICE APPROVAL)

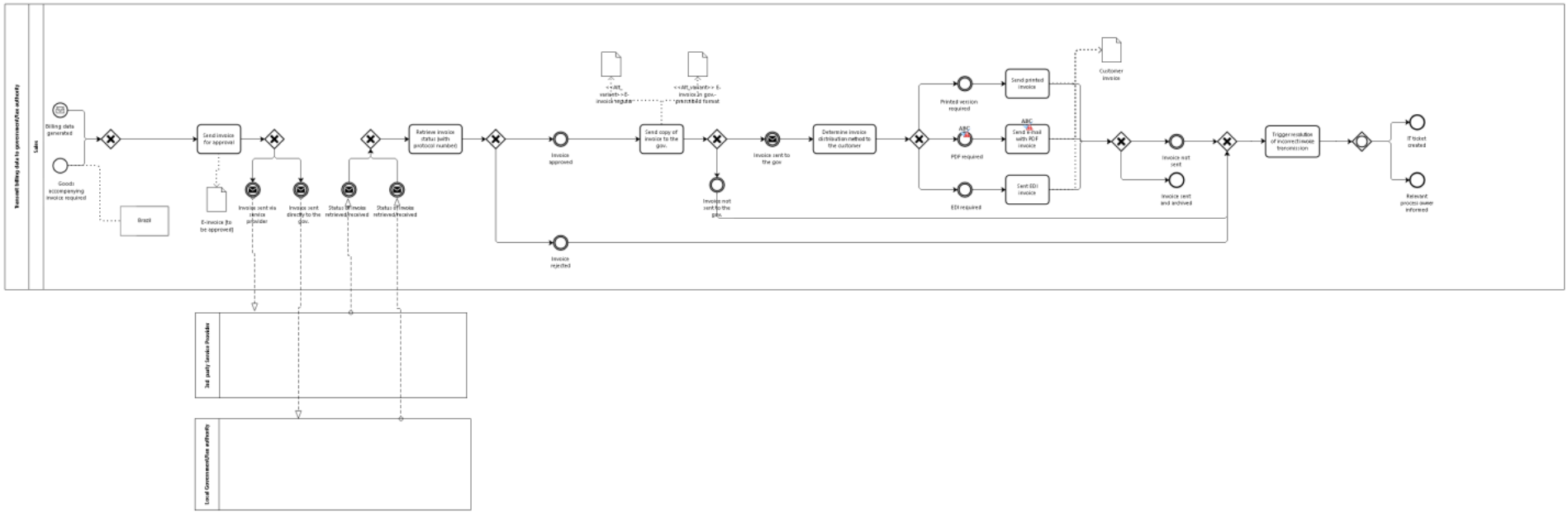


FIGURE 35. TRANSMIT BILLING DATA TO CUSTOMER (SHIPMENT AUTHORIZATION)