A VARIABILITY-GUIDED METHODOLOGY FOR MICROSERVICE-BASED DEVELOPMENT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BETÜL KURUOĞLU DOLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

DECEMBER 2022

Approval of the thesis:

A VARIABILITY-GUIDED METHODOLOGY FOR MICROSERVICE-BASED DEVELOPMENT

submitted by **BETÜL KURUOĞLU DOLU** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Ali H. Doğru
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Halit Oğuztüzün
Computer Engineering, METU _____

Prof. Dr. Ali H. Doğru
Computer Engineering, METU _____

Assoc. Prof. Dr. Pelin Angın
Computer Engineering, METU _____

Assist. Prof. Dr. Gül Tokdemir
Computer Engineering, Çankaya University _____

Assist. Prof. Dr. Selma Nazlıoğlu
Software Engineering, Atılım University _____

Date:20.12.2022

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Betül Kuruoğlu Dolu

Signature :

ABSTRACT

A VARIABILITY-GUIDED METHODOLOGY FOR MICROSERVICE-BASED DEVELOPMENT

Kuruoğlu Dolu, Betül
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Ali H. Doğru

December 2022, 60 pages

This thesis presents a microservice-based development approach, MSDeveloper (Microservices Developer), employing variability management for product configuration through a low-code development environment. The purpose of this approach is to offer a general-purpose environment for the easier development of families of products for different domains: a domain-oriented development environment is suggested, where domain developers and product developers can utilize the environment as a software ecosystem. Thus, genericity is offered through supporting different domains. A domain is populated with feature and process models and microservices in a layered architecture. Feature models drive the product configuration, which affects the process model and the microservice layer. An experimental study was conducted to validate the applicability of the approach and the usability of the development environment. Students from different courses were assigned system modeling projects where they utilized helper tools supporting the provided methodology. Furthermore, professional software developers were consulted about this recommended domain-oriented development environment. Feedback from student projects and remarks of professionals are analyzed and discussed.

Keywords: low-code development, microservices architecture, model-driven engineering, process modeling, software development, variability modeling

ÖZ

MİKROSERVİS TABANLI GELİŞTİRME İÇİN DEĞİŞKENLİK REHBERLİKLİ BİR METODOLOJİ

Kuruoğlu Dolu, Betül
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ali H. Doğru

Aralık 2022 , 60 sayfa

Bu tez, ürün yapılanması için düşük kodlu bir geliştirme ortamı aracılığıyla, değişkenlik yönetimi kullanan, MSDeveloper adlı mikroservis tabanlı bir geliştirme yaklaşımı sunmaktadır. Bu yaklaşımın amacı, farklı çalışma alanlarında ürün ailelerinin daha kolay geliştirilmesi için genel amaçlı bir ortam sunmaktır. Yazılım geliştiricilerinin ve ürün geliştiricilerinin, bir yazılım ekosistemi olarak kullanabilecekleri çalışma alanı odaklı geliştirme ortamı önerilmektedir. Böylece, jeneriklik farklı alanları destekleyerek sunulmaktadır. Bir çalışma alanı için katmanlı bir mimaride öznitelik ve süreç modelleri ve mikroservisler hazırlanır. Öznitelik modeli, süreç modelini ve mikroservis katmanını etkileyerek ürün yapılandırmasını yönlendirir. Yaklaşımın uygulanabilirliğini ve geliştirme ortamının kullanılabilirliğini doğrulamak için deneysel bir çalışma yapılmıştır. Farklı derslerden öğrencilere, önerilen metodolojiyi destekleyen yardımcı araçlar sağlanarak sistem modelleme projeleri verildi. Ayrıca, önerilen bu etki alanı odaklı geliştirme ortamı hakkında profesyonel yazılım geliştiricilere danışıldı. Bu tez kapsamında, öğrenci projelerinden gelen geri bildirimler ve profesyonellerin görüşleri analiz edildi ve tartışıldı.

Anahtar Kelimeler: düşük kod geliştirme, mikroservis mimari, model güdümlü mühendislik, süreç modelleme, yazılım geliştirme, değişkenlik modelleme

to my sweetheart Umay

ACKNOWLEDGMENTS

First and foremost, I want to express my deepest gratitude to my supervisor Prof. Dr. Ali Hikmet Doğru for his extremely inciting guidance, useful advices, and motivating encouragements he unsparingly presented throughout the research. With his inspiring view of life, he has been and will be an excellent role model for me.

I would like to thank my Thesis Monitoring Committee, Prof. Dr. Halit Oğuztüzün and Assist. Prof. Dr. Gül Tokdemir for spending their valuable time. I highly appreciate their patience and support. I am also very thankful to Assoc. Prof. Dr. Pelin Angın and Assist. Prof. Dr. Selma Nazlıoglu for being extremely helpful, encouraging and kind during the thesis defense.

I am also overwhelmed with gratitude for the very kind contributions of the members of our research group; Dr. M. Çağrı Kaya and Anıl Çetinkaya. Without their knowledge and assistance this study would not have yield a success.

I would like to offer my deepest appreciation and thanks to my working company ASELSAN that gives importance to my personal development as much as myself; to my director Gürsel Şahin, to my manager Hasan Konya; and to my team leader Özgür Bağlıoğlu for their kind supports; to all my officemates who passed through nearly all the stages of the thesis together with me.

I have been lucky enough to have the support of many good friends. Life would not have been the same without Gizem, Burçin, İlksen, Kamuran, Esra, Ece, Halime, Duygu, Cihat and Fikret.

Most importantly, however, as none of all these would have been possible without the loving care, earnest support and endless patience of my dearest mother, father and sister. My immediate family have always been a constant source of love, concern, support and strength all these years.

Last, but surely not least, I would like to offer my heartfelt thanks to my dearly

beloved Ozan and my sweetheart Umay. I am deeply grateful to them, since they, in my most distressful times during preparing this thesis, put up with my changing moods, have always been very kind and caring and supported and guided me out of the hardships I have fallen in. It is surely thanks to them that this thesis is now at your hands, ready to be read.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

BPMN	Business Process Modeling Notation
CRM	Customer Relationship Management
FM	Feature Model
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IT	Information Technology
LCDP	Low Code Development Platform
MM	Microservice Model
OVM	Orthogonal Variability Model
PLE	Product Line Engineering
REST	Representational State Transfer
SaaS	Software as a Service
SoA	Service Oriented Architecture
SPL	Software Product Line
SPLE	Software Product Line Engineering
UML	Unified Modeling Language
VM	Variability Model

CHAPTER 1

INTRODUCTION

With the changing world, software is rapidly being included in every aspect of our lives in an indispensable way. Therefore, low-cost, consistent, error-free, and user-friendly software production has become very important. Low-code development intends to serve these purposes and allows the fast development of products supported by visual modeling that will achieve needs by writing little to no code (Waszkowski, 2019). It is a development approach that responds quickly to customers' needs with reduced costs. However, most of the existing studies in this field are tool-centric, references are scarce (Sahay et al., 2020) and less inclusive about conceptual frameworks, and are methodologies especially lacking.

In the world of software development, which transforms, develops, and evolves every day, software development approaches are also being improved, renewed, and expanded. With a background of almost 10 years, low-code development is a software development approach that has become quite popular. Its practical use is supported by visual modeling, which allows quick development, especially leveraging on mature domains with low costs. Low-code development environments are easy to use in a product-oriented way, even for users without a programming background, which relieves them from mundane "infrastructural" tasks. These environments are being further established in the IT sector day by day (Sahay et al., 2020).

The expectations sound ambitious. We therefore impose some restrictions. First, the developers who will maintain coding phases in low-code development environments should be advanced computer users—preferably developers. Then, the development environments should be domain-oriented. This requirement relieves the developers from many development issues that are "infrastructural". For any application field,

the development domain should be prepared with inclusive models and executable code for the common functions in that field. We refer to such a domain as mature Togay et al., 2008 if there is no unwritten code left for the functions required in that field. Such code could take the form of components or microservices. This expectation is not unrealistic for today.

There are many no-code and low-code development environments that are gathering bigger developer communities. Owing to the long past of "faster development" initiatives, there are some common patterns of use and feel for such tools. Earlier Integrated Development Environments (IDE) offered drag-and-drop facilities for the "low-code" development of graphical user interfaces (GUI). Later, the "widgets" used in those environments were made database-aware. Additionally, earlier desktop database management systems were supported with such environments to quickly configure applications with GUI and databases glued with some procedural code and SQL.

Some enterprises such as Creatio (formerly bpm'online) (Creatio, 2022), Pega (Pega, 2022), Salesforce (Salesforce, 2022), and Zoho (Zoho Creator, 2022) specialised in Customer Relationship Management (CRM) solutions, and incorporated low-code development extensions within their products via the orchestration of processes in their business process model functionalities (Vincent et al., 2019). There are also low-code tools prepared for a specific domain, such as IoT, as researched in ref. (Ihirwe et al., 2020).

The influence of such earlier successful tools can be observed in modern approaches. We are, however, seeking a more systematic approach for related methodologies and supporting tools (Kaya et al., 2018; Kaya et al., 2019; Suloglu et al., 2020). The existing tools are usually preferred for specific domains. If a more generic approach was desired, software ecosystems could probably be used in the definition of various domains, making possible the configuration of products in such domains. For genericity, the infrastructure for development should allow the definition of different domains (domain engineering), as well as product development. Additionally, a simple high-level reference architecture would benefit genericity. An ecosystem that allows the easy definition and instantiation of new domains and products will support

continued infrastructural success and gain wider user communities.

Our vision is to provide a generic approach utilizing the existing experience in many architectural avenues, such as Service-Oriented Architecture (SoA) and Software Product Lines (SPL). A more foundational concern has been voiced in the past when programming languages were being discussed for offering generic capabilities: the Böhm Jacopini Theorem (Böhm & Jacopini, 1966) has proven genericity through the minimum requirements for the control structures to be included in the languages for that matter. They demonstrated the Turing Machine equivalency. In a similar fashion, fundamental design dimensions in this research are intended to be addressed for genericity. These dimensions are catered by the constituents of the modern development approaches organized in our methodology.

This thesis reports on our ongoing work, which offers graphical modeling environments for an example application domain (Webinar System) and supports the development of products without code writing once the developers refine their set of graphical models (Feature, Variability, and Business Process models) and Microservice Domain Model in conformance with each other. A two-layer architecture is proposed for accommodating these models, corresponding to the run-time environment (Kuruoglu Dolu et al., 2022).

1.1 Motivation and Problem Definition

Considering that software is involved in almost every field, it is of paramount importance to develop low-cost, consistent, error-free and user-friendly software. Therefore, there is a need to accelerate software processes without adversely affecting the quality of the software.

If various software systems are being developed for a specific domain or a domain group, it is essential to make this development process a fast production process. The similarity and the dissimilarity, that exist between software systems belonging to the same domain, need to be defined at the domain level.

We realized the need to define the low-code development method, that supports the

users in terms of the above-mentioned needs that are the acceleration of software processes, and the development of domain-specific software systems from a single source. It is possible to find solutions to the aforementioned needs with a variability-guided low-code development environment. This environment can be defined by a layered architecture and can be based on microservice architecture in coding. Besides, software developers as well as those who do not know can be included in the development processes.

1.2 Contributions and Novelties

Due to the increasing need to produce products with similar qualities, in order to reduce time and cost to develop products with almost similar requirement sets, new ways are being explored in the IT sector. Rather than defining the similar software needs of the same or different clients as separate projects, it is more effective to manage it as a product and present it to the client by preparing the product that meets the requirement set. The proposed methodology offers a generic environment, which is suitable for defining different domains, allowing the development of products in a variability-guided manner. For this approach, variability stands out as the ultimate development concept. Furthermore, SPLE is expanded to low-code directions, and the expansion bridges these two concepts.

1.3 Structure of the Thesis

This thesis is organized as follows: In Chapter 2, the background information which forms the basis of our study and studies regarding low-code development platforms and modeling approaches of these studies are presented. In Chapter 3, our approach and architectural design are explained in detail. Chapter 4 presents our experimental study conducted for the Webinar Systems domain, the survey we conducted after the study and professionals' remarks about the approach. The analysis we conducted on the results of the survey is also explained in Chapter 4. In Chapter 5, general applicability and validity of the approach is discussed along with concluding remarks.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Background

SoA has been a widely accepted and employed avenue for developing complex and distributed applications that usually connect through the Internet. The integration mechanisms supported through process models that are intuitive and graphical actually present an invaluable approach to development. They leverage the world market of web services that provide any kind of functionality and that are already written, tested, and published. The need to create more flexible and scalable systems leads to moving attention from developing monolithic applications to independently functioning microservices. With each managing their own data storage, microservices, which are potentially written in different languages, use lightweight protocols, such as HTTP and REST. As the focus is independent deployment, scaling, and testing, microservices require a bare minimum centralized management (Kyle, 2016; Lewis & Fowler, 2014).

Our previous work focused mainly on the structure dimension (Dogru & Tanik, 2003), which actually corresponds to the decomposition view of Software Architecture approaches. We later realized that our research neglected an analysis of an important issue—dynamic modeling. This need has been addressed in UML through sequence diagrams or activity diagrams and in SoA by the process model. Another very powerful concept offered by the SPL approaches, namely the domain conceptions presented through feature models and variability, has also been inspirational.

SPL constitutes an environment for the construction of a set of software systems from core assets by feature management, thus making it easier to satisfy specific require-

ments. Common assets lie at the core of the development; therefore, the handling of these assets is crucial, in addition to variability management in the creation of a final product. Ref. (Pohl et al., 2005) specified a requirement for a framework with two distinct processes: domain engineering and application engineering, with the first one being for the characterization and realization of necessary assets and the latter for the production of distinctive applications under the guidance of variability. A final product is created as a result of systematic decisions on variability resolution throughout the development phases. Each design decision corresponds to a variation point by incorporating/eliminating features. Delayed design decisions improve the efficiency of SPL by allowing the core assets to be used along with changing requirements (Van Gurp et al., 2001). This can be supported through variability, especially with later resolution times.

Some early approaches have been used for utilizing domain feature models in order to resolve variability; however, they were complicated because variability was accommodated in an already populated feature model. The Orthogonal Variability Model (OVM) (Pohl et al., 2005) and Covamof (Sinnema et al., 2004) are some of the pioneering variability models that stand alone outside of feature models. According to Shahin, OVM is a good choice for representing a variability model. In his study, variability modeling has been defined for each layer for SaaS (Software as a Service) applications generated by using SOA architecture. Transitions between these variability models are defined in (Shahin, 2014).

In an effort to orchestrate these concepts in the environments to further develop complex software more easily, some research was conducted regarding methodology. The decomposition of the structural view governing the components or the web services was later supported by configuring a process model for the integration and flow-control specifications. The latter addition enabled the ordering of method invocations to complete the definition of executable systems. Although today no development process should dictate a linear order of tasks, such as that of Waterfall, some priority for the modeling dimensions that would correspond to the abstraction levels of design was necessary. To support our expectations, the limited studies conducted in Ref. (Cetinkaya et al., 2019) showed that the process model can have a priority over the component model, as has practically been the case in the SoA world, in which an

implied two-level architecture is the hidden de facto standard: A process model at the top commands the web services at the bottom.

The desire to support many different ecosystem domains through potentially different communities for both defining and utilization (through no/low-code development) necessitates us to have command over the domain conceptions. That is why SPL was exploited for feature models and variability, ripening our environment. Variability will be the sole first-class citizen, probably accounting for the vast majority of the specification considerations. For this reason, we tried to align all the models under variability and support the microservices with domain-specific connector utilities (Kaya et al., 2018) that play the final roles in configuring with respect to variability.

2.2 Related Work

In this section, we present some work directly related with no- and low-code software development, as well as some related research. BPMN and SPL fields are considered especially related in our work. Additionally related is the platform concept, which can be offered as a cloud service to support ecosystem capabilities.

Increasing demand in the software field has led to a never-ending search for the next development methodology. Increasing demand for complex systems composed of heterogeneous components increased the requirements for reuse and automation to meet requirements. Low-code development is a fairly new topic, and according to reports published by Gartner (Vincent et al., 2019) and Forrester (Richardson & Rymer, 2016), it is expected to gain significant popularity over the next few years. The core principles are adopted from Model-Driven Engineering (MDE) approaches (Basciani et al., 2014).Waszkowski presents Aurea BPM that employs BPMN (Waszkowski, 2019). The target is to provide automation solutions for manufacturing. Model-driven software development, rapid application development, automatic code generation, and visual programming are presented as approaches that can lead to the creation of low-code programming.

A conceptual comparative framework was proposed, and a technical survey is presented in Ref. (Sahay et al., 2020). Eight low-code development platforms (LCDP)

are considered as market leaders based on Refs. (Richardson & Rymer, 2016; Vincent et al., 2019): Appian (Appian, 2022), Google App Maker (this service has been shut down since April 15, 2020), Kissflow (Kissflow, 2022), Mendix (Mendix, 2022), MS Power Apps (Microsoft Power Apps, 2022) , Salesforce Platform (Salesforce, 2022), Outsystems (Outsystems, 2022), and Zoho Creator (Zoho Creator, 2022). All of the overviewed LCDPs utilized a process designer and supported workflow management in their operations. However, only Kissflow, Salesforce App Cloud, and Zoho Creator allow users to configure workflows according to their needs.

The Business Process Model Notation (BPMN) specifies a Business Process Diagram (BPD), which is based on a flowcharting method designed for building graphical representations of business process operations. Therefore, a business process model is a network of graphical objects that represent activities (i.e., work) and the flow controls that determine the sequence in which they are performed (White, 2004).

H. Van Vilet defined Software Product Line Engineering (SPLE) as software engineering approaches, resources, manners with the aim of building comparable software systems from a shared set of software resources by employing a familiar instrument of production in 1993 (van Vliet, 1993).

Two essential steps in the software product line engineering process are domain engineering and application engineering. The process of software product line engineering that defines and realizes the commonality and variability of the product line is referred to as Domain Engineering. On the other hand, Application Engineering is the process of software product line engineering in which the applications of the product line are produced by reusing domain artifacts and taking advantage of the diversity of the product line (Böckle et al., 2005).

Bhushal et al. proposes a framework which is based on ontological rules. Their claim about this framework is that SPL's quality is improved by this method. This framework firstly converts Feature Model to Feature Model Ontology to formalize the model. In this conversion process, 11 rules are used to detect false-optional and dead features with their causes. These rules help Product Line Modelers to eliminate false-optional and dead features that are caused by incorporated relationships. The framework can also detect wrong group cardinalities in Feature Models resulting

false-optional and dead features (Bhushan et al., 2017).

Impact Analysis is an approach for Software Product Line change impact management. Existing Software Product Line change management approaches work on the feature model. This automated method gives chance to the designer to keep consistency of the design. It defines the needed modification set and each modification triggers changes from feature model to design (Maâzoun et al., 2016).

CHAPTER 3

METHODOLOGY

The proposed methodology, MSDeveloper supports the development of software systems in a top-down approach. This approach is suggested with a layered model, where the process model is the model at the top layer and the structural model is at the bottom layer (Cetinkaya et al., 2019). Both process and structural models are compliant with the variability model. The process model located at the higher level and the web services at the bottom layer point to the de facto architecture in SoA. Additionally, conventional development based on UML starts with the functional dimension modeled with the use case diagrams. These widely practiced approaches, along with our research, suggest a model hierarchy where processes are at the top (Cetinkaya et al., 2019). Rather than organizing our architecture based on structural components, such as GUI and data, etc., we prefer to process the top-level model, followed by the low-level executional models, which are microservices. These lower-level units also accommodate data structures to account for the data dimension. Following our discussion in the introduction related to genericity, fundamental design dimensions are considered in the shaping of the proposed reference architecture in Figure 3.1.

The modeling of function, data, and control support executability (Turing Machine equivalency) concerns, whereas structure supports understandability, hence the manageability of the development. These could be referred to as "3+1 dimensions of design". Table 3.1 lists the mapping of those dimensions to development structures. Furthermore, the process assumes control responsibilities at a higher (coordination) level. Since we are not interested in coding-level control issues, such as inside the methods, we have a better support in this dimension through process modeling.

It should be noted here that the microservice philosophy suggests a single responsibil-

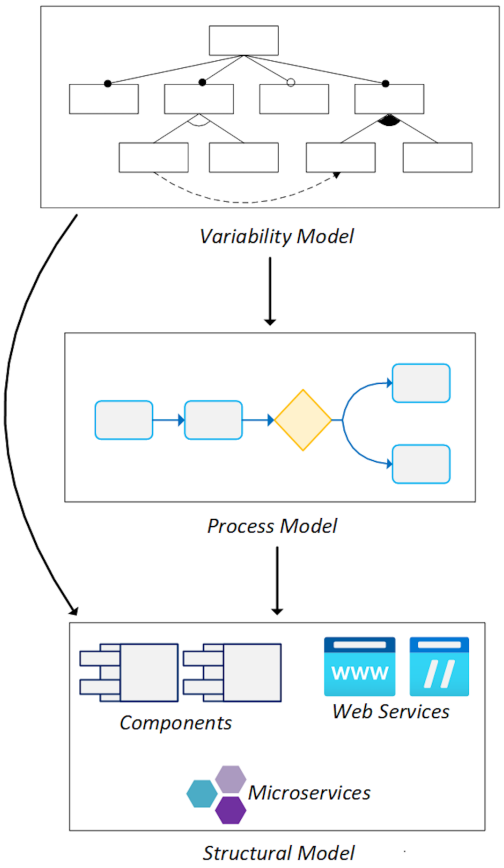

Figure 3.1: The Reference Model of MSDeveloper.

Table 3.1: Mapping of dimensions to development structures.

Design Dimension	**Architecture Element**
Control	Process model (BPMN)
Function	Microservice methods
Data	Microservice attributes
Structure	Two-layered architecture

ity per service, practically resulting in simpler and smaller services. The orchestration assumes more duties because services are simpler. Consequently, the reference architecture does not allocate complex structures for the data and the function dimensions.

Within the scope of this study, microservices have been prepared to be stateful. There is a one-to-many relationship between an optional feature and the tasks associated

with that feature. Likewise, there is a one-to-many relationship between a task and its associated microservices.

The suggested method requires a mature domain to be applied. Reusing existing code with a new configuration or generating new code from graphical models makes mature domains (Togay et al., 2008) possible, in which all common functionality and data were created earlier by domain experts. A software ecosystem can be created that supports different domains. In the suggested method, a domain in which applications will be created is assumed to be mature. Each domain in the ecosystem includes its own template process model for future applications. This step corresponds to the domain engineering stage of Software Product Line Engineering (SPLE). Additionally, developers will be able to use pre-implemented components in their designs or create their own based on their preferences. This step corresponds to the application engineering stage of the SPLE. Based on the selected domain, users will be presented with a feature model, in addition to a variability and a process model that correspond to the feature model.

Users will be selecting the features that will be included in the final product from the variability model. Based on these selections, the process model and its required microservices will be automatically configured to incorporate the selected features.

The emphasis here is on product engineering. Preparing a domain in the ecosystem consists of the activities given in Table 3.2, where the order is not imposed. It is suggested to start with the domain feature model and a variability model. Rather than taking one model at a time in a linear process, all models can be developed simultaneously. Simultaneous top-down development can be supported by the Axiomatic Design (Suh, 1996) approach for the development of complex domains.

In a similar fashion, the product development is also suggested to start top-down and continue with a free-ordered process to modify any model any time. “Acquire Any Missing Services” activity in the product development steps will be required in the case of a non-mature domain.

Table 3.2: Domain and product development steps.

Domain Development Steps	**Product Development Steps**
1. Develop feature model;	1. Resolve variabilities;
2. Develop variability model;	2. Instantiate the process model;
3. Develop base process;	3. Instantiate the microservices model;
4. Develop optional processes;	a. Acquire any missing services.
5. Acquire microservices.	

3.1 Methodological Principles

In our work, we propose a methodology that can be used as a low-code development environment for mature domains, where the success largely depends on the maturity of domain models. The experience provided in SPL practices relieves us from the redundant declarations and definitions that took place in previous products. These are the fruits of domain orientation. Now, the low-code developers can be offered variations corresponding to a world of known products rather than the definition of requirements from scratch, which are often abstract.

There are four architectural units in MSDeveloper that are mostly defined in the domain engineering that takes place before the low-code development. Then, these models are used for the development of new applications. These units are the feature, variability, process, and microservice models. Additionally, there is a fifth unit, which was created during low-code product development: Graphical User Interface (GUI). This research excluded the GUI constituent because it has been previously addressed in depth by many applications and will be incorporated to our framework later. The framework supports the creation of different domains, and the presence of communities that can define different domains will enable the environment to support low-code development as a general approach for application development. We often use the word domain to define the set of four models that are defined for a specific application field. Figure 3.2 depicts the architectural components and their usage in MSDeveloper.

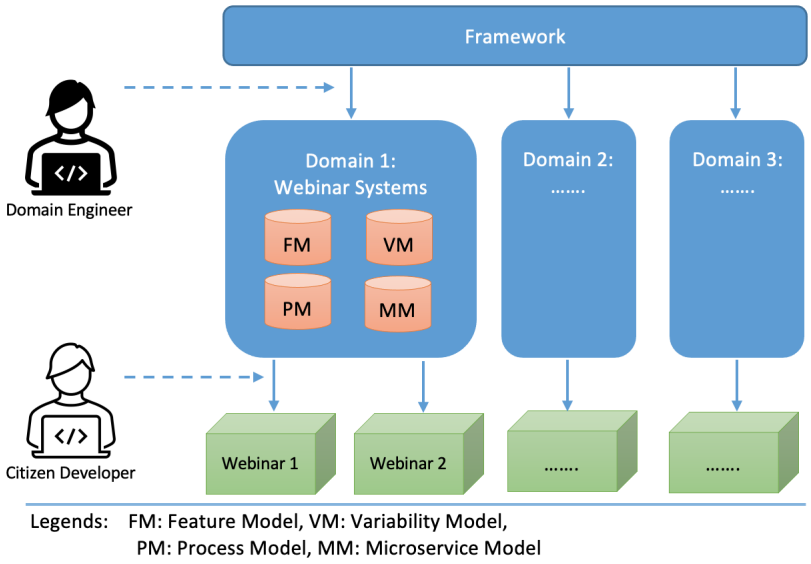

Figure 3.2: Usage of architectural components in MSDeveloper..

The transition from feature-model layer to process-model layer and the transition from the process-model layer to the microservice layer are explained below.

Domain engineering efforts are presented below, and are organized with respect to the corresponding models:

- A variability model (VM) is created (if necessary, from a feature model), as shown in Figure 3.3. The variability model is prepared according to the optional features and constraints that are defined in the feature model. All mandatory features that are included in every product are represented in the base feature model. The feature model is composed of the base feature model and the variability model;
- A domain process model (PM) is prepared with Business Process Modeling Notation (BPMN), which will include every feature. This process model corresponds to the domain. The tasks, gateways, and sequence flows associated with each optionally defined feature are extracted from this domain BPMN and kept as a partial BPMN for this optional feature. This operation is performed for all optional features. As a result, base (covering all mandatory features) and partial BPMNs for all optional features are prepared. Later, for the low-code development of the product corresponding to the product engineering part, ac-

cording to the optional feature selections made in the variability model, the relevant tasks, gateways, and sequence flows of selected optional features are added to the base BPMN model. As a result, a product-specific BPMN model will be instantiated according to the changes made to the variability model. The domain BPMN is composed of the base BPMN and the partial BPMNs;

- A microservice model (MM) layer is prepared to cover all the related features. A microservice may be required to be developed for each feature (some features may only correspond to processes). While the microservices prepared for mandatory features create a base microservice model, the microservices prepared for each optional feature create optional microservices. According to the optional feature selections made in the variability model, the relevant microservices and base microservice model's (mandatory features') microservices are prepared for method calls. The microservice model is composed of the base microservice model and the optional microservices;
- The optional microservice methods that are linked directly to an optional feature are associated to the relevant task(s) located in that feature's partial BPMN. The mandatory microservice methods that are linked directly to a mandatory feature are associated with the tasks located in the base BPMN. As a result, according to the changes made to the variability model, the microservice model methods are also prepared to be called by the BPMN.

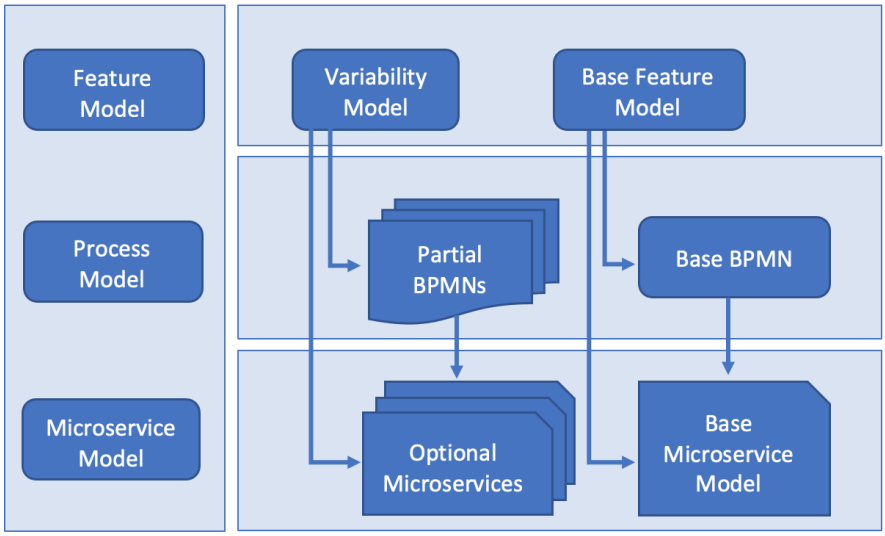

Figure 3.3: Models used in MSDeveloper.

MSDeveloper is shaped to carry the following properties:

- The environment supports feature-based development,
- Variability resolution and related constraints propagate through all the models,
- BPMN is automatically configured as a result of variability resolution, and
- Performing microservice method calls through the BPMN provides the final steps in executability.

Product engineering is the part that corresponds more to low-code development, especially if a mature domain exists. A methodological flow of activities can also be presented for this task. However, the flow of activities does not prescribe a strict order. A natural start from higher-level abstraction models, and the development activities concerning them, are natural. As the inclusive feedback arrows imply, modifying any model after any assessment is possible. MSDeveloper adapts the activity flow resembling the low-code methodology for the product development in Figure 3.4.

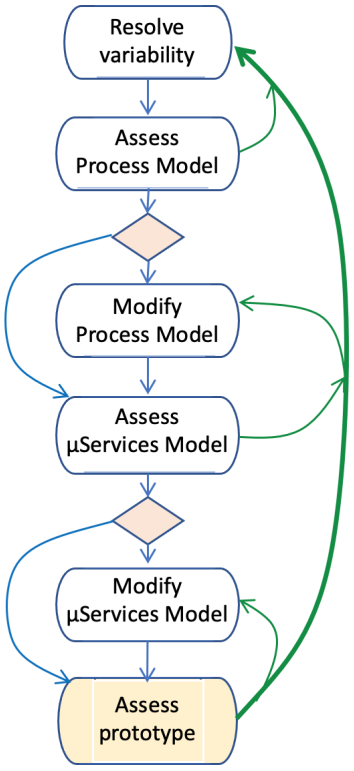

Figure 3.4: The low-code development methodology.

Consistency is an important criterion for both domain engineering and product en-

gineering outcomes. As stated above, variability resolution and related constraints propagate through all the models, which supports the consistency of the output. After variability decisions, which are handled as a product engineering phase, the temporary outcome in the process and the microservice models can be reviewed. Until the desired consistency is achieved, the developers can iterate through the activities of variability modifications and processes and microservice model inspections and modifications.

CHAPTER 4

VALIDATION

4.1 Experimental Study

In this section, we present our experimental study for evaluating the proposed methodology with student projects. Students used supportive prototype tools to model target systems by adhering to the methodology.

4.1.1 Development Environment

We set up an environment where Eclipse IDE for Java Developers (includes Incubating components) version December 2020 (4.18.0) is employed with the following plugins:

- FeatureIDE Plugin, Version: 3.7.0.202010141034;
- BPMN Plugin, Version: 1.5.2.SNAPSHOT-v20200602-1600-B1;
- Maven Plugin, Version: 1.17.1.20201207-1112

Moreover, we developed domain-independent and generic jar files that allow transitions (from feature model to BPMN and from BPMN to microservice method calls) and domain-related SpringBoot applications. Software structures for the selected domain and webinar system (feature model, auto-generated variability model, process model, and microservice model) are available in Ref. (Kuruoglu Dolu, 2022).

4.1.2 Selected Domain: Webinar System

Webinar is a combination of the words “web” and “seminar”. It is used for systems that allow users to organize and participate in a video workshop, lecture, or presentation in an online environment. With the current COVID-19, such systems have become a part of our lives. Due to its popularity, we chose webinar systems for this experiment’s domain.

Our chosen feature set for the webinar system contains the following features: live and pre-recorded video streaming, mobile accessibility, chat messaging, file sharing capabilities, virtual whiteboards, shared screens, virtual waiting rooms, breakout rooms, pass-presenter tools, polling tools, app integration, audio and video recording, in-app conference registration, automated reminders, Q and A tools, engagement analysis for attendance and lead generation, in-app offers, integration with live-streaming social media platforms, active speaker view, AI and gamification features, and backgrounds to create a branded virtual environment (Webb, 2022).

4.1.3 The Experiment: Webinar System Development

We conducted the experimental study in two phases. We asked the participants to complete the tasks given in Table 4.1 for Phase 1 and Phase 2.

Table 4.1: The experimental study phases.

Phase 1	**Phase 2**
Domain Engineering Stages	**Domain Engineering Stages**
-Preparation of the feature model;	-Preparation of the domain-specific methods;
-Preparation of the variability model;	-Invocation of all the related methods over the base BPMN;
-Preparation of the domain BPMN;	-Invocation of the related methods from the partial BPMNs;
-Preparation of the base BPMN;	**Product development stages**
-Preparation of the partial BPMNs.	-Selection of the variability model and automatic development of the product;
	-Execution of the auto-generated product-specific BPMN model;
	-Observation of correctness in the execution of related methods.

4.1.3.1 Participants

We conducted the experiment at the Department of Computer Engineering of Middle East Technical University during the 2020–2021 spring semester; a graduate class (CENG551 System Development with Abstract Design) and a senior class (CENG454 Introduction to Software Architecture) participated in this experiment. We asked students to form groups of two or three students. We gave groups a document (Cetinkaya, 2020b) that included a system specification for the webinar system's design, along with the user manual for the development platform to be used in this experiment. The manual included details about the installed plugins in the provided development environment and general instructions about how to operate the environment.

After the completion of the project, we asked participants to answer some questions about their experience, provided development environment, and the system they de-

signed. We conducted this questionnaire through the Internet. A total of 30 students participated in the questionnaire. 43% of the participants registered for the graduate class and 57% of them registered for the senior class. The survey questions are given in Appendix A and the answers are presented in Ref. (Cetinkaya, 2020a).

4.1.3.2 Phase 1

We introduced the development settings to the developers before sharing the experimental webinar system domain. The shared feature model, which takes place in the shared workspace, is depicted in Figure 4.1, and the shared variability model (which automatically performs all the controls regarding the features and constraints) is represented in Figure 4.2, and the shared BPMN model is shown in Figure 4.3.

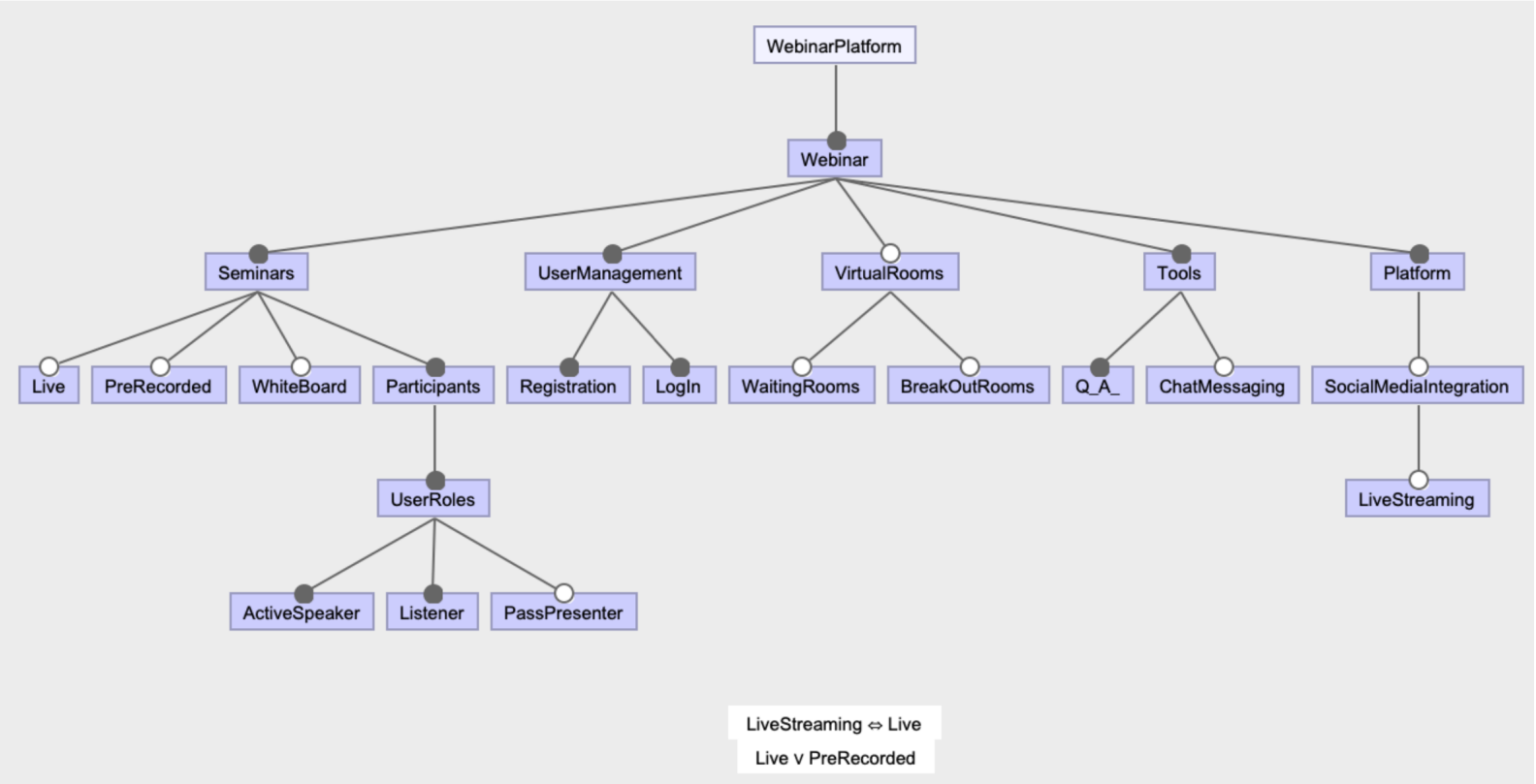

Figure 4.1: Webinar system domain feature model.

We asked developers to enrich the webinar system domain with optional and mandatory features to develop it towards a mature domain. The participants added the features they deemed necessary and enhanced the feature model by defining the constraints among the features. In this stage, they were able to perform visual modeling by using the drag-and-drop capability of the framework. Then, they prepared the variability model through support from the facility provided by the framework. A representation of the variability model is shown in Figure 4.2. After that, they prepared the base and the partial BPMNs for each optional feature.

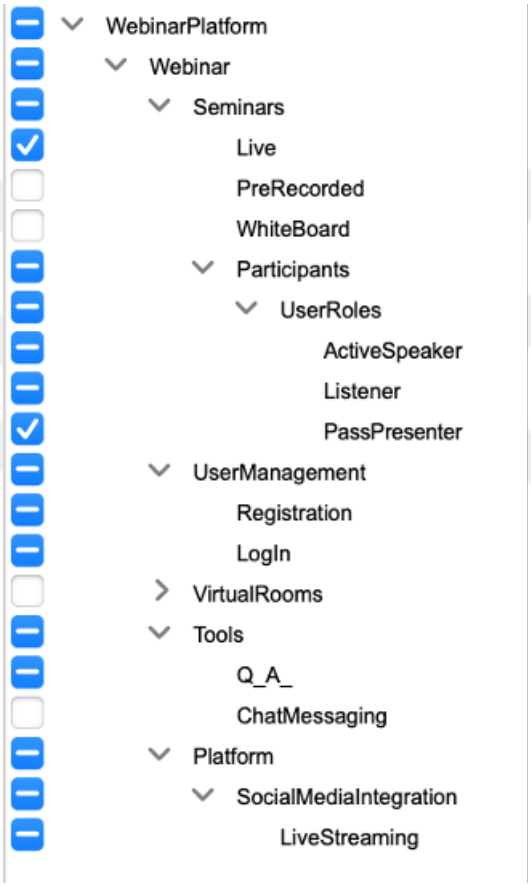

Figure 4.2: Variability model of webinar system domain.

After these preparatory stages, when the developers generated the product, they were able to observe that the resultant BPMN diagram was drawn according to the selections made in the variability model. The selected features were linked with corresponding process assets in the BPMN model. Additionally, we excluded assets that were directly linked with the unselected features from the BPMN Model.

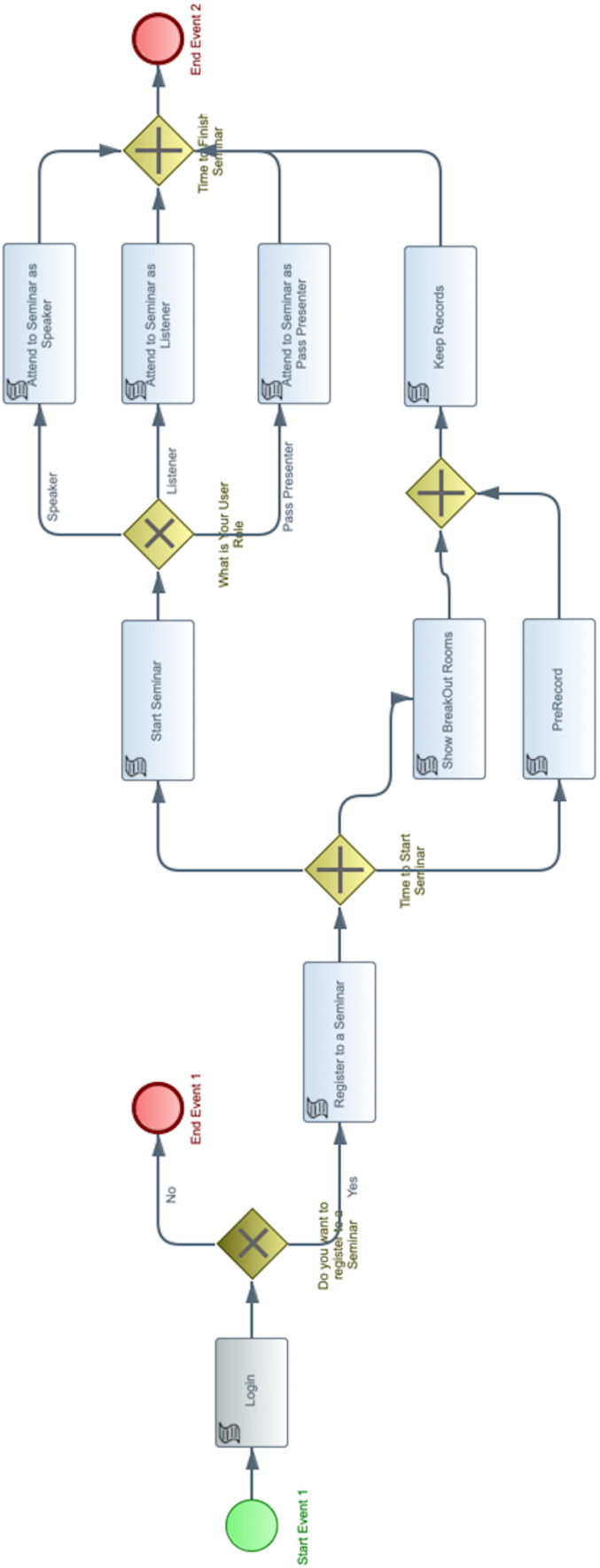

Figure 4.3: BPMN of webinar system domain.

4.1.3.3 Phase 2

We asked developers to perform method calls over the scripts corresponding to the tasks included in the BPMN. We asked them to call these methods by writing Java methods within these scripts. After, we asked them to add method calls to the partial BPMNs. As a result, the choices to be made on the variability model have a direct effect on the BPMN and the methods called over the BPMN. In this phase participants used methods from Java classes that can directly represent a set of microservices.

In the final stage, we asked the developers to generate products with different configurations on the development environment they prepared for the webinar system. In this stage, they experienced the ability to produce products that differ according to the choices they made simply by updating the variability model.

4.2 An Example Run for the Experiment

For clarity, it is appropriate to include an example of experimental work here. First of all, the feature model is prepared. Then, the variability model is generated according to the mandatory and optional features and the constraints of the feature model. A sample transition from the feature model to the variability model is given in Figure 4.4.

In its simplest form, the feature model for a Webinar System can be prepared with the following feature set below according to its hierarchy. The bold features are the optional ones. The variability model allows the user to define a configuration by selecting optional features by complying with the rules. In this example, while PassPresenter and WhiteBoard features are selected, ChatMessaging feature is not.

- Webinar
  - Live
  - PreRecord
  - Participants
    - UserRoles

            - ActiveSpeaker
            - Listener
            - **PassPresenter**
        - **Whiteboard**
- UserManagement
    - Registration
    - Login
- VirtualRooms
    - WaitingRooms
    - BreakOutRooms
- Tools
    - QuestionsAndAnswers
    - **ChatMessaging**
- Platform
    - SocialMediaEntegration
        * LiveStreaming

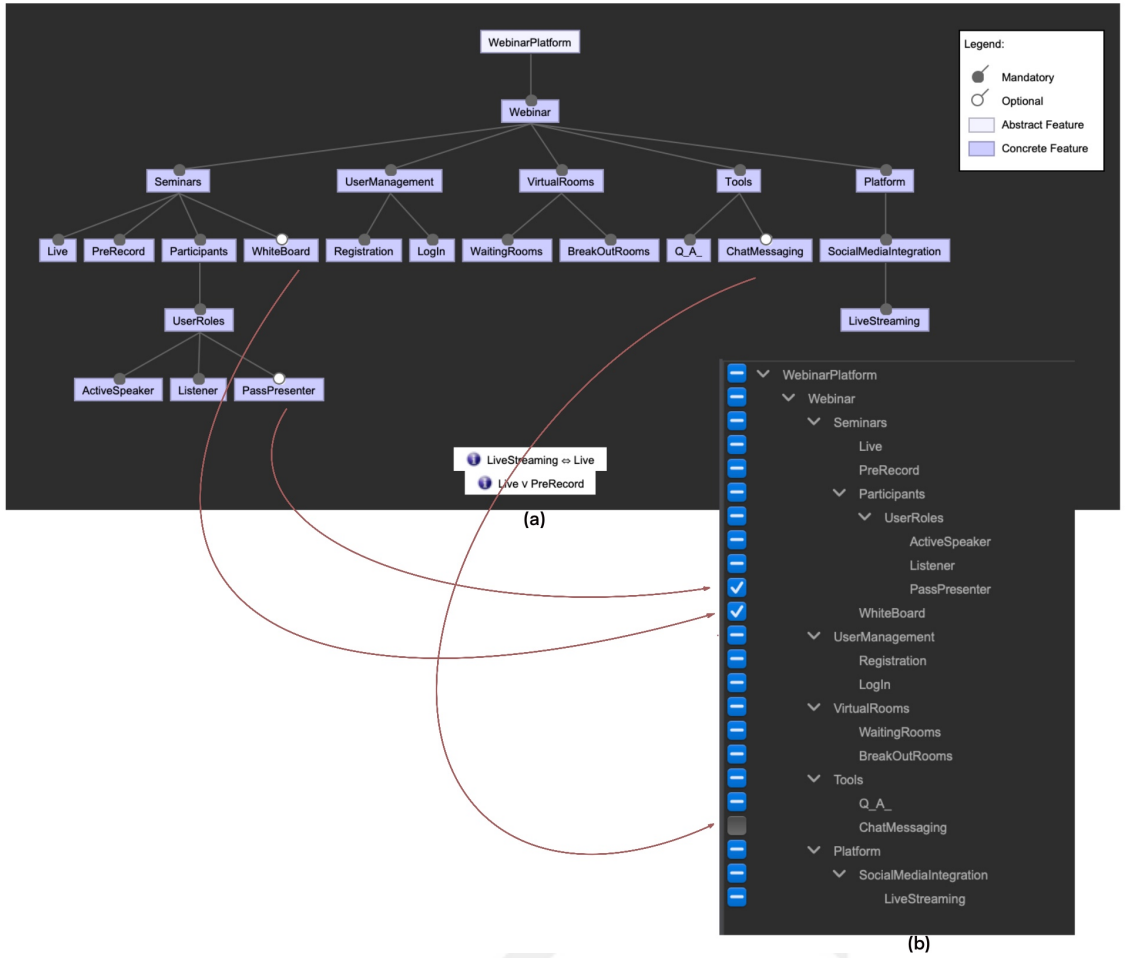

Figure 4.4: a. The feature model. b. The variability model. The transition from the feature model to the variability model.

After choosing optional features on the variability model in accordance with the constraints, the model is run. At this stage, the FeatureIDE plugin that was added to the Framework automatically performs constraints control. The types of constraints offered to the user by FeatureIDE while defining features or feature relations are as follows: Mandatory, Optional, Or, Alternative, Abstract, Concrete, False optional. Of these rules, Alternative and Or are used to define the relationship between features, while the others are used to define any feature. As an example, if two features are linked to each other with Or constraint, at least one of them has to be selected on the corresponding variability model. This process is the product creation phase itself according to the configuration prepared by the selections. As a result, the product-specific BPMN is automatically generated. The product-specific BPMN is obtained by combining the base BPMN and the partial BPMNs are prepared for selected features as described in Chapter 3. The partial BPMN example for ChatMessaging optional feature is given in Appendix C. In addition, the required microservices become active and ready to be used. The variability model, the generated BPMN, and the required microservices are given in Figure 4.5.

In the example above, while PassPresenter and WhiteBoard are selected from the optional features, ChatMessaging is not selected. As a result, it is seen that while there are tasks for PassPresenter and WhiteBoard on the generated BPMN, there is no task for ChatMessaging. Likewise, when looking at the status information of the microservices prepared for optional features, it is seen that only the microservices, which are relevant to the selected features, are active.

It is very important to activate only the necessary microservices. For instance, when the product needs to be prepared for a commercial purpose, it allows the product to be packaged in accordance with technical requirements and free from unnecessary capabilities.

After generating the BPMN and activating the relevant microservices, this specific BPMN, which is the product, is executed. Each task, mandatory or optional, calls the relevant method of the relevant microservice in accordance with the process and the controls defined in the BPMN. The logs printed on the console after the execution and the related BPMN tasks are given in Figure 4.6.

Looking at this console output, it can be seen that while microservice method calls are made for mandatory and selected optional features (PassPresenter and WhiteBoard), there is neither a task on BPMN nor a method call for ChatMessaging feature.

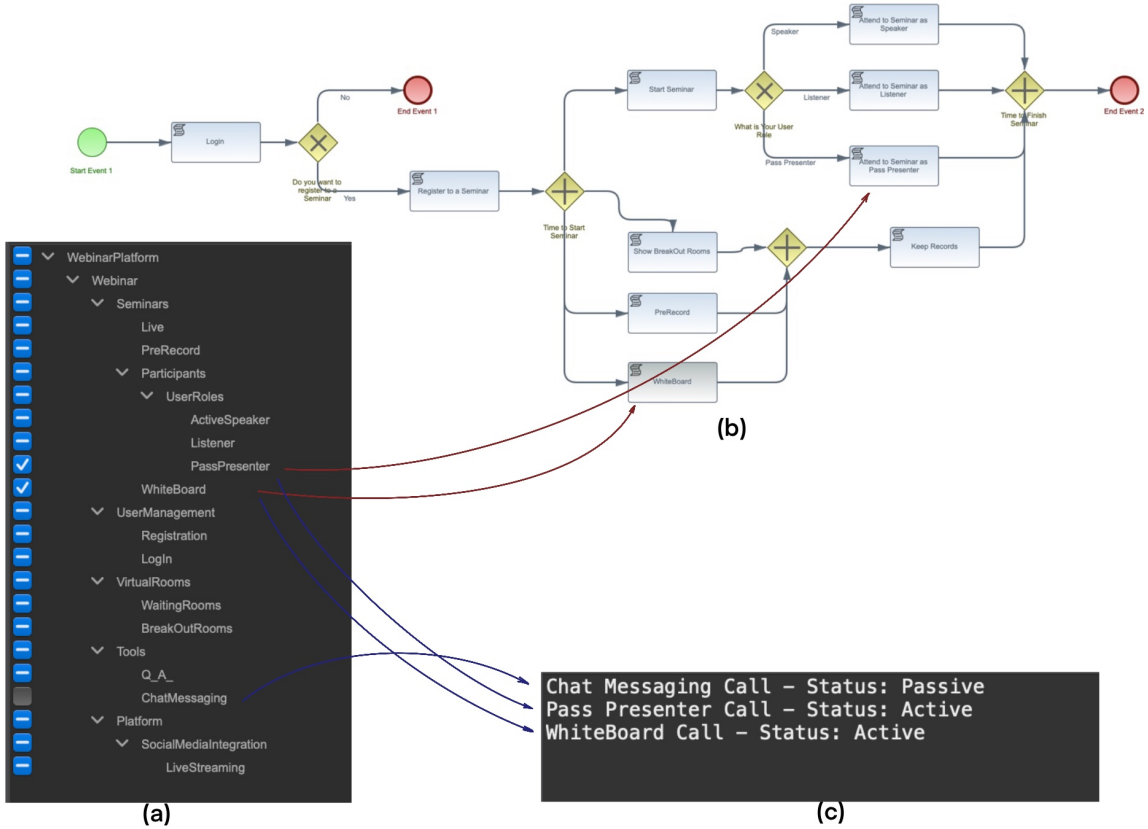

Figure 4.5: a. The variability model and the selections. b. The generated BPMN. c. The required microservices and their status information. The variability model and its effects on the generated BPMN and the required microservices.

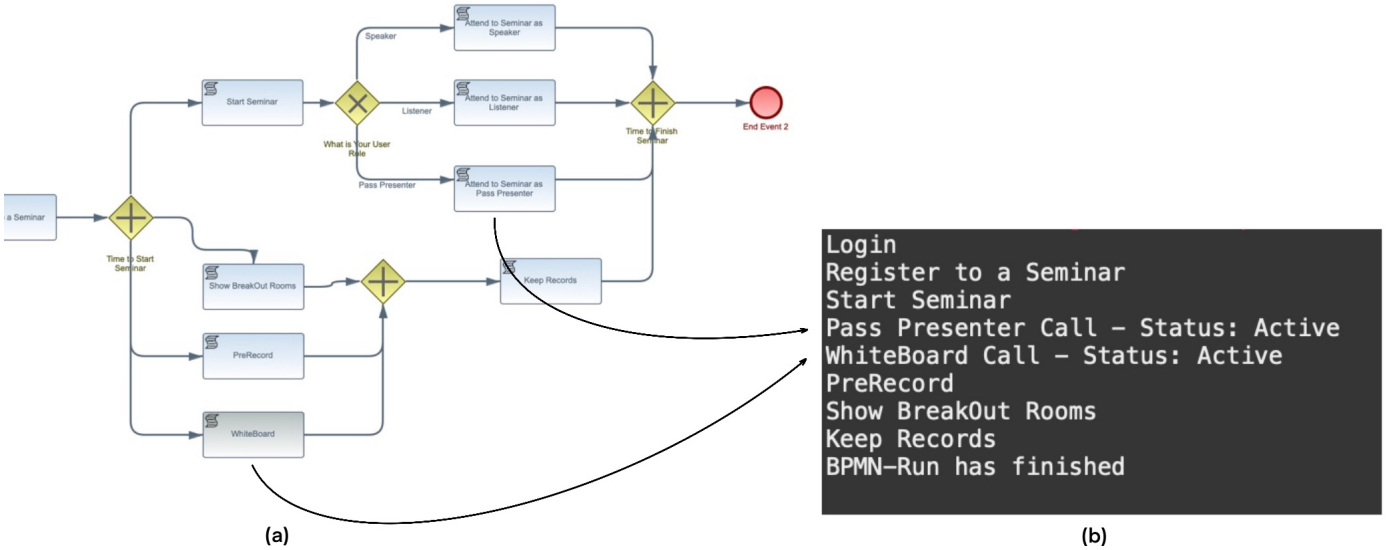

Figure 4.6: a. The related BPMN tasks. b. The logs of the microservice method calls. After the BPMN execution, the microservice method calls.

4.3 Results

In this section, the experiment is analyzed and the experimental study results are given. Besides, professionals' remarks on the proposed approach are given.

4.3.1 Experimental Study Results

After the completion of the project, the participants were asked to fill out the survey questions. We analyzed the information we gathered from the participants via questionnaires. The results of the surveys are analyzed below. It should be noted that these participants are not professionals. This experiment group with no familiarity with low-code development rated their experience on a scale of 1 to 10, 1 being the hardest and 10 being the easiest, and they were asked to answer questions.

We can deduce from the following results stated in Figure 4.7 that half of the participants who have never developed in a low-code development environment do not find this environment very easy. On the other hand, it is seen that 64% of the participants have a positive opinion about the framework that we have prepared. Moreover, 86% of respondents think that they understood the framework well. This is encouraging feedback that suggested that we were able to explain our methodology and tools to the participants well. At this point, a remarkable result is encountered when the answers given by the students are analyzed considering the classes they were registered in. While graduate students answered the question "How well did you understand the framework?" with an average of 7.2 out of 10, senior students answered with an average of 5.8. We see that after the domain-oriented development processes, 79% of them find it easy to generate the products through different configurations corresponding to low-code development.

When we asked the participants about their experiences regarding the transitions among models, we encountered the results in Figure 4.8. The answers we obtained are not very surprising. Transitions between the feature model and BPMN model cover the creation of the domain BPMN and the preparation of the base and partial BPMNs. In summary, it includes drawing a business process model covering the entire domain and more. For this reason, only 57% of people think that this transition

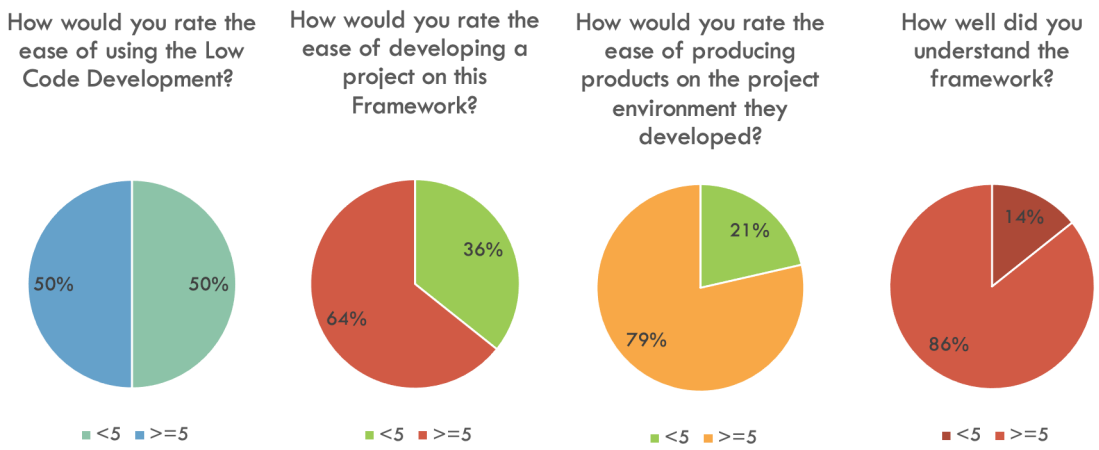

Figure 4.7: Easiness of the framework.

is easy. On the other hand, operations involving method calls from the BPMN model were found easy by 86%. We think that this is due to the fact that method calls from within the tasks defined in the BPMN model can be made through the easy-to-use GUI of the framework. It is also seen in this question that the opinions of the students differ according to the class information they were enrolled in. While graduate students answered the question "How would you rate the easiness of switching from BPMN to method calls?" with an average of 8.2 out of 10, senior students answered with an average of 6.2.

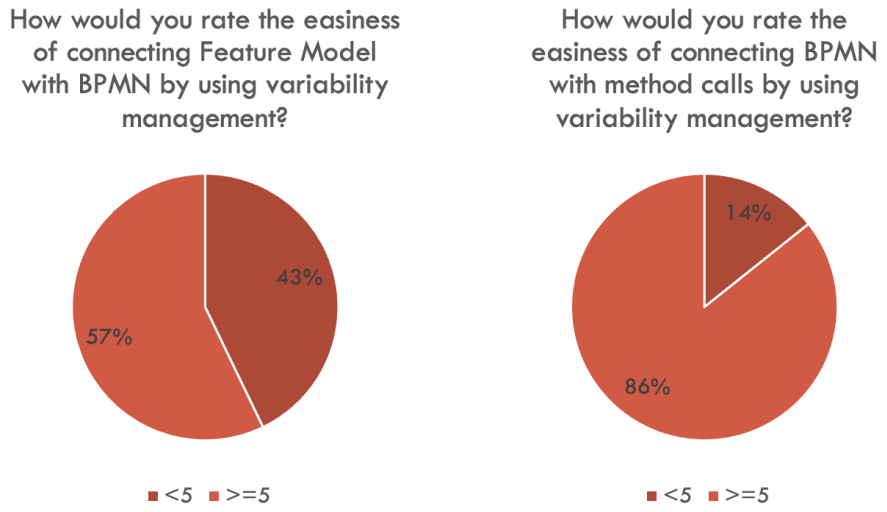

Figure 4.8: Questions about transitions between models and the responses.

One of the things we were most curious about during the experiment was the time

spent. The answers we received to the questions we asked for this purpose are given in Figure 4.9. A total of 62% of respondents said they spent less than or equal to 10 h developing their webinar system domain. When generating the products with different configurations, 86% explained that they spent less than or equal to 6 h. We inferred that the time spent here is due to an immature domain because the framework completes the creation process of products developed in different configurations in milliseconds. However, the participants, who probably did not have a chance to develop a mature domain, encountered errors, corrections, etc. So, this step may have taken a long time.

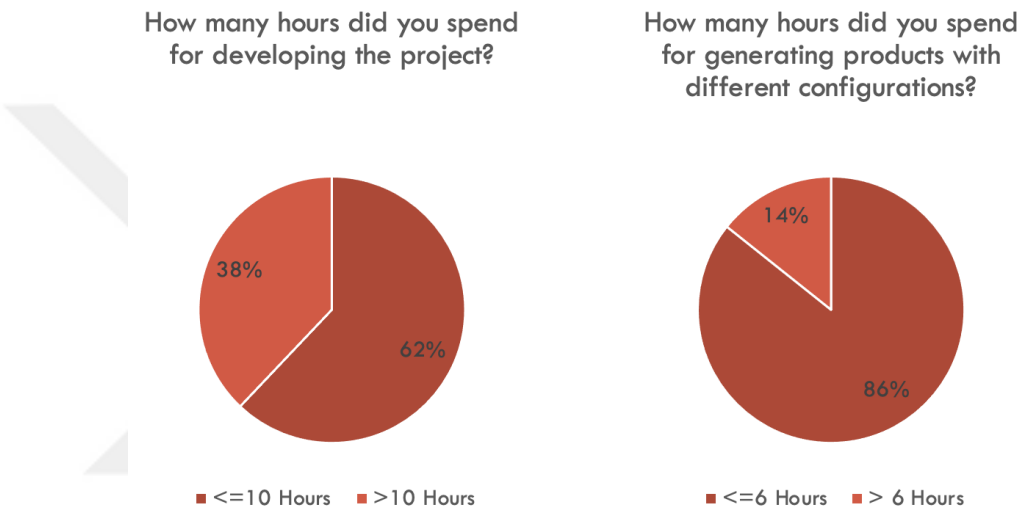

Figure 4.9: Questions about required time and the responses.

4.3.2 Professionals' Remarks

The methodology and the framework were also shared with the professionals. In a video, the framework was introduced, and the processes of adding an optional feature on a mature domain and the effects of this feature addition on the feature, variability, and the BPMN models, as well as the method calls, are shown (Kuruoglu Dolu, 2021b).

After watching the video, the professionals were asked to answer some questions about their opinions. This survey was conducted using the Internet. A total of 32 professionals participated in the survey. The survey questions are given in Appendix B

and the answers are presented in (Kuruoglu Dolu, 2021a). The profile of the professionals (their work experience, company profile, and current working type) is shown in Figure 4.10.

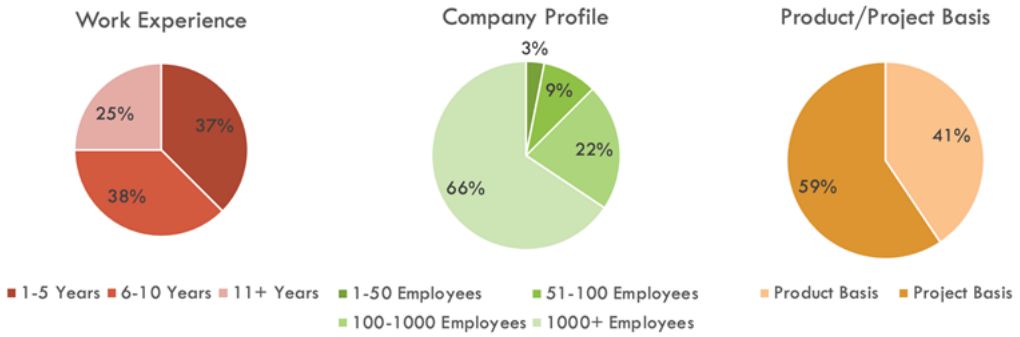

Figure 4.10: The profile of the professionals.

In Figure 4.11, we see the answers given by the project-based working professionals to the questions related with the projects they are working on. When we ask the question "Have you taken part in more than one project in the company you work for?", 89% of them answered "Yes". When we asked about the company's project quantity, all of them answered this question with "10+ projects have been developed by the company in the field that s/he works in". In addition, 38% of them mentioned that the similarity between these projects is more than 75% and 46% of them said that the similarity between these projects is more than 50%.

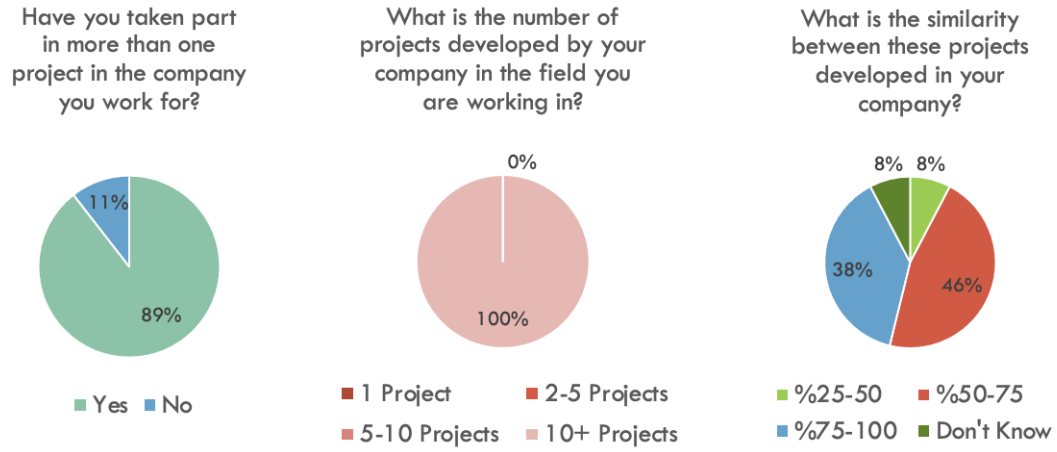

Figure 4.11: Project-based working professionals' answers.

In Figure 4.12, we see the answers given by the product-based working profession-

als regarding questions related with the products they are working on. When we ask the question “Are you maintaining a single version (the most recent version) of the product you are working on?”, 69% of them answered “No, Multiple Versions”. That means most of them maintain the products with different configurations. Additionally, 69% of them find it difficult to see the overall picture and 92% of them have difficulties because of the size of the product, especially when they want to add a new feature to the product. After watching the video that we introduced in our methodology on the framework, 100% of them answered “Yes” to the question “Specific to the field you are working in, would a Framework be useful?”.

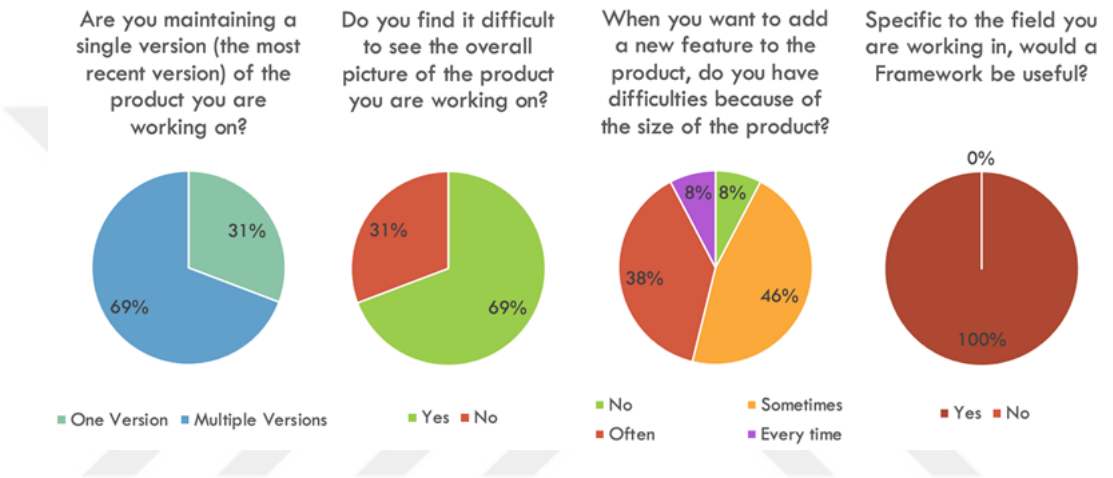

Figure 4.12: Product-based working professionals’ answers.

In Figure 4.13, we see the answers given by all the professionals about the easiness of the framework in different perspectives, including developing a domain, creating a product, adding a feature, connecting a feature model with BPMN, and connecting BPMN with method calls. The evaluations of professionals with an average of 7.73 years of work experience are around 8 points, and these responses are greater than students’ evaluations.

An open-ended question is also asked: “What are your views on the suitability of this methodology for professional life?”. Some answers are listed below:

- In general, it will be useful for the structural standardization of the code. Models created by experienced teams will automatically ensure that less experienced employees stay within the quality constraints;

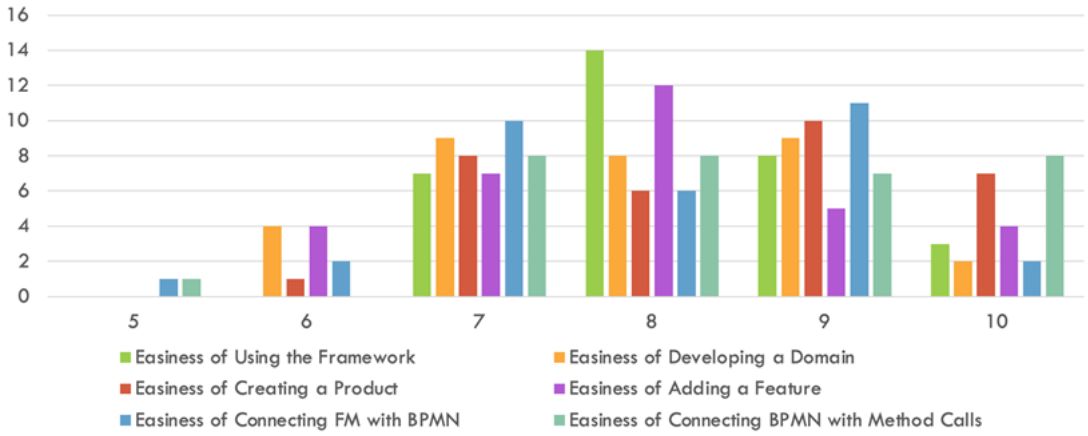

Figure 4.13: Easiness of the framework.

- It can be used easily and effectively in many large and small projects, and the time it saves in the development process cannot be underestimated;
- Low-code libraries make software development processes much easier and shorter today. Low-code development will become much more widespread in the future of the rapidly developing software industry;
- In general, the requirements are analyzed, a software/system is designed on top of it, and the whole development process continues, both waterfall and iterative. The Variability-Guided Development method seems to be able to parallelize both high- and low-level designs in waterfall or iterative processes. At the same time, it is a method that will minimize the workload when a new project is desired with the capabilities already in the product pool. It has an important place in creating backlogs for new features that need to be implemented. When Software Product Line is considered, maybe it will be helpful to deliver a project without the need for a developer;
- It can minimize the deficiencies that may occur during the manual management of multiple versions and the varying configurations of these versions. In addition, it will reduce the time spent managing the configurations with each changing version;
- It is very logical and solution-oriented because the project I am working on needs a framework just like this one. If given the opportunity and time, I would consider using it;
- It will be very useful, and it will save a lot of time since there are very similar

projects with different configurations and code bases.

CHAPTER 5

CONCLUSION

The experiments and expert opinions support the usability of our proposal. The claim to offer a generic capability for the development of professional software would be validated through further experience, especially in industrial media. Expected experience is the definition of various domains and the creation of products in commercial settings. Nevertheless the conducted study provides a proof of concept. Our future work will continue in this direction to gain wider acceptance.

It was stated in the previous section that 100% of the participating professionals would find such a framework useful. Considering the difficulties of creating a mature domain in the professional world, it will be necessary to facilitate the domain development process. We aim to find new fields of study for the suitability of the domain development process with agile approaches, for example feature-driven development.

All features and constraints depend on the feature model, and all other layers are developed in accordance with corresponding constraints. This makes the whole process easier. For example, the function of a microservice, and whether it will be mandatory or optional, are very clearly revealed before starting its development. This situation is similar for the BPMN-model layer.

One of the strengths of this study is that the produced products are not given with unnecessary and extra capabilities. Especially if there are capabilities managed by configurations (with files, with database tables, etc.) for a product family in a particular domain, these capabilities may be changed by users. However, in this study, the microservice layer is activated in accordance with the variability model. In this way, it will not be possible for users to reach a capability that is not in the configuration of

the product they use.

The framework we developed in academic settings is still in its infancy. However, by understanding the literature, examining the previously developed tools in detail, and by correctly understanding the needs of users, we aim to prepare a tool that is more useful and beneficial. According to the survey results, even though it is a new framework, the framework was found to be useful and promising. In the upcoming period, we will continue our work by evaluating the feedback we received from the participants and the remarks of the professionals.

Within the scope of this study, the users who mostly participated in the experiment with the role of domain developer were asked to prepare partial BPMNs for optional features during the transition from the feature model to the process model. Considering that all the other stages were conducted through graphical user interfaces, we can observe that this is the only part in which the participants were challenged; therefore, there was a need for improvement on this matter.

We can conclude that the framework was successful with its low-code development goal. Microservices were appropriate to support the lower-level layer of the proposed architecture because of their granularity. We demonstrated the use of the environment for the product engineering task. However, to complete the spectrum for general-purpose development, domain engineering tasks also need to be experimented on. Our experimentation provided a prepared domain; however, the simple mechanisms adopted, while shown to be suitable, also indicate that effort would be involved in their development—domain engineering. Nevertheless, future work is required to experiment with this phase.

For future studies, it is aimed to create products with Graphical User Interfaces (GUIs) as the output of the product engineering phase. In this way, real, usable products will be produced, going far beyond the proof of concept. Experiments with this kind of product that appeal to users will also be realistic and guide our future work.

In this study, microservices are defined as stateful. Even microservices that serve for common use have to be defined and used separately. In future studies, to prepare a more effective and simplified Microservice Layer, the state information will be

kept and used in Process Model. In this way, common microservices will serve for different tasks and features.

In this study, a product family development that will start from scratch is described and experiments have been carried out in this way. However, this methodology is also very suitable for creating products in different configurations from a product already under development. At this point, the process of adapting an existing product to the MSDeveloper method might be defined. It is a question of defining a partial migration to MSDeveloper. This can be also one of our future works.

It is not difficult to realize that the environment is suitable for general-purpose development. We are planning to continue our work in two avenues: integrating the tools to achieve a stand-alone and consistent development environment and conducting experimentations for the domain engineering phase. For this, the project groups will identify different domains and populate them with the feature, process, and microservice models.

REFERENCES

Appian. (2022). Appian platform overview [Accessed: 2022-08-19]. https://appian.com/platform/overview.html

Basciani, F., Iovino, L., Pierantonio, A., et al. (2014). Mdeforge: An extensible web-based modeling platform. *2nd International Workshop on Model-Driven Engineering on and for the Cloud, CloudMDE 2014, Co-located with the 17th International Conference on Model Driven Engineering Languages and Systems, MoDELS 2014*, *1242*, 66–75.

Bhushan, M., Goel, S., Loura, A., & Negi, A. (2017). Managing software product line using an ontological rule-based framework, 376–382. https://doi.org/10.1109/ICTUS.2017.8286036

Böckle, G., Pohl, K., & Linden, F. (2005). A framework for software product line engineering. https://doi.org/10.1007/3-540-28901-1_2

Böhm, C., & Jacopini, G. (1966). Flow diagrams, turing machines and languages with only two formation rules. *Communications of the ACM*, *9*(5), 366–371.

Cetinkaya, A. (2020a). Low-Code Development Environment Survey and Students' Responses. [Accessed: 2022-08-19]. https://docs.google.com/spreadsheets/d/1itr5W4dWq3lUBHEYmQ2okDDUR8m1iWs8oMFN-XcbvPg/

Cetinkaya, A. (2020b). Project Specification and Development Platform Manual Prepared for Students. [Accessed: 2022-08-19]. https://drive.google.com/file/d/1sJC27hAGwfBIMpAuV4-cUeGqzbyK0hVF/

Cetinkaya, A., Suloglu, S., Cagri Kaya, M., Karamanlioglu, A., Tokdemir, G., & Dogru, A. H. (2019). An experimental study on decomposition: Process first or structure first? *International Symposium on Business Modeling and Software Design*, 279–289.

Creatio. (2022). Creatio studio platform overview [Accessed: 2022-08-25]. https://www.creatio.com/studio

Dogru, A. H., & Tanik, M. M. (2003). A process model for component-oriented software engineering. *IEEE software*, *20*(2), 34–41.

Ihirwe, F., Di Ruscio, D., Mazzini, S., Pierini, P., & Pierantonio, A. (2020). Low-code engineering for internet of things: A state of research. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 1–8.

Kaya, M. C., Cetinkaya, A., & Dogru, A. H. (2018). Off-the-shelf connectors for interdisciplinary components. *Journal of Integrated Design and Process Science*, *22*(3), 35–53.

Kaya, M. C., Suloglu, S., Tokdemir, G., Tekinerdogan, B., & Dogru, A. H. (2019). Variability incorporated simultaneous decomposition of models under structural and procedural views. In *Software engineering for variability intensive systems* (pp. 95–115). Auerbach Publications.

Kissflow. (2022). Kissflow platform overview [Accessed: 2022-08-19]. https://kissflow.com/platform/

Kuruoglu Dolu, B. (2021a). Low-Code Development Environment Survey and Professionals' Responses. [Accessed: 2022-08-19]. https://docs.google.com/spreadsheets/d/1BWFIp2CFhmcXRiXqyXQdJayz5IFqMWtM0SF4dUJzptI

Kuruoglu Dolu, B. (2021b). MSDeveloper Development Environment Introductory Video. [Accessed: 2022-08-19]. https://vimeo.com/664041103/ea798f30bc

Kuruoglu Dolu, B. (2022). The development workspace. [Accessed: 2022-10-07]. https://github.com/betulkuruoglu/developmentWorkspace

Kuruoglu Dolu, B., Cetinkaya, A., Kaya, M. C., Nazlioglu, S., & Dogru, A. H. (2022). Msdeveloper: A variability-guided methodology for microservice-based development. *Applied Sciences*, *12*(22). https://www.mdpi.com/2076-3417/12/22/11439

Kyle, B. (2016). Beyond buzzwords: A brief history of microservices patterns [Accessed: 2022-08-19]. https://developer.ibm.com/articles/cl-evolution-microservices-patterns/

Lewis, J., & Fowler, M. (2014). Microservices, a definition of this new architectural term [Accessed: 2022-08-19]. https://martinfowler.com/articles/microservices.html

Maâzoun, J., Bouassida, N., & Ben-Abdallah, H. (2016). Change impact analysis for software product lines. *Journal of King Saud University - Computer and Information Sciences*, *28*(4), 364–380. https://doi.org/https://doi.org/10.1016/j.jksuci.2016.01.005

Mendix. (2022). Mendix platform overview [Accessed: 2022-08-19]. https://www.mendix.com/low-code-guide/

Microsoft Power Apps. (2022). Microsoft power apps platform overview [Accessed: 2022-08-19]. https://powerapps.microsoft.com/en-us/

Outsystems. (2022). Outsystems platform overview [Accessed: 2022-08-19]. https://www.outsystems.com/

Pega. (2022). Pega platform overview [Accessed: 2022-08-25]. https://www.pega.com/products/platform

Pohl, K., Böckle, G., & Linden, F. (2005). Software product line engineering: Foundations, principles, and techniques springer. *URL http://www. springer. com/us/book/9783540243724*.

Richardson, C., & Rymer, J. R. (2016). Vendor landscape: The fractured, fertile terrain of low-code application platforms. *FORRESTER, Janeiro*.

Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178.

Salesforce. (2022). Salesforce platform overview [Accessed: 2022-08-19]. https://www.salesforce.com/eu/products/platform/overview/

Shahin, A. (2014). Variability modeling for customizable saas applications. *International Journal of Computer Science and Information Technology*, *6*. https://doi.org/10.5121/ijcsit.2014.6503

Sinnema, M., Deelstra, S., Nijhuis, J., & Bosch, J. (2004). Covamof: A framework for modeling variability in software product families. *International Conference on Software Product Lines*, 197–213.

Suh, N. P. (1996). Chapter 1 introduction to axiomatic design.

Suloglu, S., Kaya, M. C., Cetinkaya, A., Karamanlioglu, A., & Dogru, A. H. (2020). Cloud-enabled domain-based software development. In *Software engineering in the era of cloud computing* (pp. 109–130). Springer.

Togay, C., Dogru, A. H., & Tanik, J. U. (2008). Systematic component-oriented development with axiomatic design. *Journal of Systems and Software*, *81*(11), 1803–1815.

Van Gurp, J., Bosch, J., & Svahnberg, M. (2001). On the notion of variability in software product lines. *Proceedings Working IEEE/IFIP Conference on Software Architecture*, 45–54.

van Vliet, J. C. (1993). *Software engineering - principles and practice*. Wiley.

Vincent, P., Iijima, K., Driver, M., Wong, J., & Natis, Y. (2019). Magic quadrant for enterprise low-code application platforms. *Gartner report*.

Waszkowski, R. (2019). Low-code platform for automating business processes in manufacturing. *IFAC-PapersOnLine*, *52*(10), 376–381.

Webb, T. (2022). Best virtual conference platforms for online events [Accessed: 2022-08-19]. https://getvoip.com/blog/virtual-conference-platforms/

White, S. (2004). Introduction to bpmn.

Zoho Creator. (2022). Zoho creator platform overview [Accessed: 2022-08-19]. https://www.zoho.com/creator/product-overview.html?src=hdd

APPENDIX A

STUDENTS' SURVEY QUESTIONS

Framework: Low-Code Development Environment shared with you

Project: Interconnected Feature Model – Configuration – BPMN – Method Calls

Product: Each "Configuration – BPMN – Method Calls" after you run your projects by changing the configuration

1. Complete following fields
   - Course Code
   - Group Number
2. Do you have any previous experience on a low-code development environment? (Answer this question per group member)
3. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of using the framework shared with you?
4. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of developing projects using the framework?
5. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of creating a product by changing the feature model configuration?
6. On a scale of 1 to 10, 1 being the lowest and 10 being the highest, how well did you understand the framework?
7. Plugins that offer you a user interface to prepare Feature Model and BPMN with constraints are given in the framework. On a scale of 1 to 10, 1 being the

hardest and 10 being the easiest, how would you rate the easiness of creating Feature Model and BPMN by using the Framework?

8. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of switching from Feature Model to BPMN by using variability management?

9. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of switching from BPMN to method calls by using variability management?

10. If you would have to develop similar products by changing the configuration, would you consider developing them by using a low-code development environment?

11. Specify how many hours you spent

    - for learning the framework
    - for developing the project
    - for creating the product according to a specific configuration

12. Which features do you think are essential for a Webinar system?

13. What do you think about the fidelity of your project? (Fidelity searches for the answer to the following question: Does the project represent a real Webinar System)?

APPENDIX B

PROFESSIONALS' SURVEY QUESTIONS

1. What is your current position?
2. How many years of work experience do you have in the Information Technologies and Software Development Industry?
3. In which field(s) did you receive your university degree(s)?
4. What is the size of your firm (number of employees)?
5. Are you working on project or product basis?
6. Have you taken part in more than one project in the company you work for?
7. What is the number of projects developed by your company in the field you are working in?
8. If the number of projects developed by your company in the field you work in is 2 or more, what is the percentage of the similarities (area, requirements, coding, testing, etc.) of these projects?
9. Are you maintaining a single version (the most recent version) of the product you are working on?
10. Do you find it difficult to see the overall picture of the product you are working on?
11. When you want to add a new feature to the product you are working on, do you have difficulties due to the size of the product?

12. Would it be useful to have a framework that keeps assets specific to the field you are working in, such as common requirements, business processes, code bases, and allows these assets to be prepared with a common infrastructure?

13. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, can you evaluate the easiness of the Framework shared with you?

14. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, can you evaluate the easiness of developing a Domain using the Framework shared with you?

15. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, can you rate the easiness of creating a Product using the Framework shared with you?

16. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, how would you rate the easiness of creating a Feature on this Framework?

17. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, can you evaluate the easiness of transitioning from Feature Model to BPMN using Variability Management?

18. On a scale of 1 to 10, 1 being the hardest and 10 being the easiest, can you rate the easiness of transitioning from BPMN to Method Calls?

19. If you need to develop similar products by changing/adding features, would you consider developing using low-code development environment?

APPENDIX C

"CHATMESSAGING" FEATURE'S PARTIAL BPMN

```
?xml version="1.0" encoding="UTF 8 " ?
bpmn2:definitions
        xmlns:bpmn2="http://www.omg.org/spec/BPMN
           /20100524/MODEL"
        xmlns:bpmndi="http://www.omg.org/spec/BPMN
           /20100524/DI"
        xmlns:dc="http://www.omg.org/spec/DD/20100524/DC
           "
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema
           instance"
        xmlns:di="http://www.omg.org/spec/DD/20100524/DI
           "
        xmlns:ns1="http://www.jboss.org/drools"
        id="fpbVUESsEeuPSqUp8HCc4g" exporter="org.
           eclipse.bpmn2.modeler.core"
        exporterVersion="1.5.2.SNAPSHOT v20200602 1 6 0 0
           B1"
        targetNamespace="http://org.eclipse.bpmn2/
           default/process "
         bpmn2:process
                bpmn2:scriptTask id="ScriptTask16"
                       name="Enable Chat Messaging"
                          scriptFormat="http://www.java
```

```
                .com/java"
              bpmn2:extensionElements
                      ns1:metaData name="
                        elementname"
                              ns1:metaValue
                                 ![CDATA[
                                Enable Chat
                                Messaging
                                ]] /ns1:
                                metaValue
                            /ns1:metaData
              /bpmn2:extensionElements
              bpmn2:incoming SequenceFlow14
                 /bpmn2:incoming
              bpmn2:outgoing SequenceFlow35
                 /bpmn2:outgoing
              bpmn2:script RestClientCalls.
                callService("http://localhost
                :8080/chatMessaging");
              /bpmn2:script
      /bpmn2:scriptTask
      bpmn2:sequenceFlow id="SequenceFlow35"
             sourceRef="ScriptTask16"
               targetRef="ParallelGateway4"
                /
      bpmn2:sequenceFlow id="SequenceFlow14"
             sourceRef="ParallelGateway1"
               targetRef="ScriptTask16" /
/bpmn2:process
bpmndi:BPMNDiagram
      bpmndi:BPMNPlane
              bpmndi:BPMNShape id="
```

```
BPMNShapeScriptTask16"
      bpmnElement="
         ScriptTask16"
         isExpanded="true"
        dc:Bounds height="50.0"
          width="110.0" x
         ="1088.0"
              y="450.0" /
        bpmndi:BPMNLabel
                dc:Bounds
                  height="11.0"
                   width="96.0"
                   x="1095.0"
                       y
                          ="469.0"
                           /
        /bpmndi:BPMNLabel
/bpmndi:BPMNShape
bpmndi:BPMNEdge id="
  BPMNEdgeSequenceFlow14"
      bpmnElement="
         SequenceFlow14"
      sourceElement="
         BPMNShapeParallelGateway1
         "
        di:waypoint xsi:type="
         dc:Point" x="1010.0"
         y="275.0" /
        di:waypoint xsi:type="
         dc:Point" x="1010.0"
         y="475.0" /
        di:waypoint xsi:type="
```

```
                    dc:Point" x="1085.0"
                    y="475.0" /
                  di:waypoint xsi:type="
                    dc:Point" x="1095.0"
                    y="475.0" /
                  bpmndi:BPMNLabel id="
                    BPMNLabel23" /
                /bpmndi:BPMNEdge
                bpmndi:BPMNEdge id="
                  BPMNEdgeSequenceFlow35"
                    bpmnElement="
                      SequenceFlow35"
                    sourceElement="
                      BPMNShapeScriptTask16
                      "
                    targetElement="
                      BPMNShapeParallelGateway4
                      "
                  di:waypoint xsi:type="
                    dc:Point" x="1198.0"
                    y="475.0" /
                  di:waypoint xsi:type="
                    dc:Point" x="1283.0"
                    y="475.0" /
                  di:waypoint xsi:type="
                    dc:Point" x="1283.0"
                    y="344.0" /
                  bpmndi:BPMNLabel id="
                    BPMNLabel52" /
                /bpmndi:BPMNEdge
        /bpmndi:BPMNPlane
/bpmndi:BPMNDiagram
```

```
/bpmn2:definitions
```

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Kuruoğlu Dolu, Betül
Nationality: Turkish (TC)

EDUCATION

Degree	Institution	Year of Graduation
M.S.	METU	2012
B.S.	METU	2009

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2009-2011	Havelsan EHSIM	Software Engineer
2011-2018	TÜBİTAK YTE	Senior Software Engineer
2018-...	ASELSAN	Leader Software Engineer

PUBLICATIONS

Science Citation Index Expanded Journal Article

Kuruoglu Dolu, B.; Cetinkaya, A.; Kaya, M.C.; Nazlioglu, S.; Dogru, A.H. MSDeveloper: A Variability-Guided Methodology for Microservice-Based Development. Appl. Sci. 2022, 12, 11439. https://doi.org/10.3390/app122211439

National Conference Paper

Ünal, Y., Kuruoglu Dolu, B.; Uyar, E.; Dikici, A. National and Original Medical Device Track and Trace Infrastructure: Product Tracking System (Tıbbi Cihaz Takip ve İzleme Ulusal ve Özgün Altyapısı: Ürün Takip Sistemi). UYMS. 2018.

Poster

Kuruoglu Dolu, B. Automatic Cartoon Generation by Learning the Style of an Artist. Expressive Graphics. 2015.