Towards a Product Line for Electronic Shelf Labels Domain

Leonardo Venâncio Teixeira[1] **, André Luiz de Oliveira**[2] **, Fredy João Valente**[1]
Fabiano Ferrari[1] **, Valter Vieira de Camargo**[1]

[1]Universidade Federal de São Carlos (UFSCar)
São Carlos – SP – Brasil
leonardo.teixeira@estudante.ufscar.br, {fredy, fcferrari, valtervcamargo}@ufscar.br

[2]Instituto de Ciências Exatas (ICE) – Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brasil
andre.oliveira@ice.ufjf.br

Abstract. *Electronic Shelf Labels are devices used to dynamically display sales price tags and other product information on the retail shop floor. It is an Internet of Things (IoT) technology that defines a new system domain; the Electronic Shelf Labels (ESL) Domain. Developing solutions for this domain is challenging due to the distributed nature of these applications, and the high degree of variabilities and features. Software Product Lines is an approach that is able to model a domain so that one can facilitate the generation of specific products from this domain. Although the development of this kind of application is complex, currently there are no proposed approaches for making the development easier by means of tools/methodologies. The goal of this paper is to present our initial steps towards the definition of a product line for ESL solutions. The approach we are following for domain analysis is based on two main sources: i) white and grey literature and ii) an existing ESL solution. These two sources provide us with theoretical and technical information from the domain. Our preliminary results include a feature diagram and a mapping table that can be used for generating products from the SPL.*

1. Introduction

Not surprisingly, the adoption of Electronic Shelf Labels (ESLs) is growing rapidly worldwide. Retailers such as Kaufland Group (a German hypermarket chain) and Whole Foods (a US grocery chain) have already implemented them. Despite their growing prevalence, however, empirical and theoretical research on ESLs is still incipient [Suh et al. 2018]. ESLs are a prime example of how the Internet of Things (IoT) is transforming the retail industry towards smart cities, smart retail, and smart shops. By integrating ESL with IoT technology, retailers can leverage the power of connectivity and automation to enhance their operations and improve the customer experience [Ma and Ma 2017, Sung 2015, Wang and Hu 2013].

ESLs, coupled with IoT capabilities, enable seamless communication between the labels and various systems within a retail environment. Each ESL acts as a connected device that can receive real-time updates and instructions from management systems. This connectivity allows retailers to remotely control and update pricing, product information, promotions, and more, across multiple store locations [Suh et al. 2018]. Through IoT connectivity, ESLs can also collect and transmit valuable data back to the retailer. These labels can monitor stock levels, track product location changes, and provide insights about consumer behaviour. This data can be analyzed to optimize inventory management, streamline supply chain operations, and make informed business decisions [Ma and Ma 2017, Shekhawat 2022a].

While various ESL solutions/applications are offered by the industry, we did not find any research initiative for facilitating the development of such applications. In this regard, Software Product Line Engineering (SPLE) is a reuse approach that supports the development of a set of software-intensive systems by combining a common and manageable set of features (core assets) that addresses the specific needs of a particular domain [Clements and Northrop 2001]. Software variant management tools, e.g., GEARS [Big Lever 2024], pure::variants [Pure-Systems 2024], support SPLE: feature modelling, mapping features to artefacts, product configuration, and derivation processes. By defining an SPL for the ESL domain, the industry can benefit from improved software quality (by reusing pre-tested components), reduced development costs, and accelerated time-to-market for new solutions.

In this paper, we present our steps towards the definition of a SPL for the ESL domain. The approach we are following in the domain analysis is based on two main domain information sources: i) white and grey literature, and ii) an existing ESL solution we are currently involved in the development of. The analysis of the white and grey literature gave us a broad overview of the domain, allowing the identification of variability and features. The experience with the development of an ESL system gave us more detailed technical information. Together, these two sources of domain information provide us with a rich information set of the domain. Therefore, the domain analysis approach we are adopting can be seen as a two-way approach because we raise domain information from two distinct sources.

This paper is organized as follows. In Section 2, we provide the background, including a brief overview of Software Product Lines (SPL) and Electronic Shelf Labels. Section 3 presents an ESL system we are currently involved in the development and that is being used as a domain information source for building an SPL. Section 4 presents the core of this paper, which is the methodological steps we are following for creating an SPL for generating ESL applications. In Section 5, we present the conclusions.

2. Background

2.1. Software Product Lines

A Software Product Line (SPL) is a set of software-intensive systems that share a common and manageable set of features that satisfy the specific needs of a particular market segment [Clements and Northrop 2001]. Feature stands for a distinct system characteristic visible to the end-user [Lee et al. 2002]. The commonalities and variabilities of a family of systems from a particular domain and their relationships are expressed in a feature model. A feature model represents the product line variability at the highest abstraction level. Features express high-level functional and quality requirements from an application domain. Feature-Oriented Domain Analysis (FODA) [Lee et al. 2002] is the first and widely used feature modelling notation, which supports the specification of mandatory, optional and alternative features and structural relationships between features, *i.e.*, decomposition, generalization, specialization, and parameterization.

The SPL development process encompasses domain engineering and application engineering phases. The product line core assets are produced during the domain engineering and further reused in the application engineering. The domain engineering phase involves domain analysis, where SPL engineers identify commonalities and variability in the domain requirements using a feature model, perform the realization of common and variant domain features by developing the SPL architecture and implementing their components (core assets), and specify the traceability between features and the core assets. In the application engineering phase, product engineers create system variants from the core assets, produced in the domain engineering, by exploiting variability and ensuring the correct binding of variation points according to the product requirements. A variation point stands for places in the domain artefacts where variability can arise. Software variant management tools, *e.g.*, pure::variants [Pure-Systems 2024] and GEARS [Big Lever 2024], support standard variability modelling in the domain, via specification of variability abstractions in a feature model. Variant management tools also support SPL engineers to map variability abstractions to their realization into the domain development artefacts (*e.g.*, requirements, design, source code, test cases). It also allows product engineers to configure their products, and derive variants based on configuration choices.

2.2. Electronic Shelf Labels

According to research conducted by Meticulous Research [Meticulous Research 2023], the Smart Shelves Market, in which Electronic Shelf Labels (ESLs) play a pivotal role, is projected to reach $10.62 billion by 2030. ESLs are electronic display devices that can be attached to shelves in stores to show the price of products. Retailers can use ESLs to employ dynamic pricing strategies, to streamline price adjustments, and also to avoid errors when using paper labels. The most recent ESLs use *e-ink* technology, hence offering the possibility of displaying information using several colours [Ma and Ma 2017].

Electronic Shelf Labels or Smart Labels are not a novel concept [Sung 2015]. They are leading to a complete paradigm shift in the retail environment by transforming non-communicative, dummy objects into smart objects. The Smart Labels are an application of the IoT (Internet of Things) in the retail space. ESLs can also be connected to the retailers' product sales price control systems, which allows for automatic price updating. There are several possible variations for electronic shelf labels, depending on the needs and available resources. Variations can be found in the display technology, size and format, connectivity types, power source, interactive features, integration with other systems [Shekhawat 2022b].

3. The XPTO-ESL-System

In this section, we provide an overview of an ESL system we are currently involved in the development. This system is devised for a global retailer whose name is omitted for confidentiality purposes. One of the main company's problems is the inefficient price update process currently in course. It involves printing numerous paper labels every time new price updates must be performed and asking for collaborators to change the prices on the shelves manually. This has generated a lot of problems in terms of performance and errors. Currently, the number of price updates in a unique store on a daily basis is around 5,000 products.

This system is being developed in a modular way as a separate subsystem so that we can integrate it with the company's ecosystem. It is being developed as a consumer (subscriber) of price update messages that are published by the company's message broker. Our system comprises three applications: i) a mobile application (called here Mobile App) which provides localization and binding services between ESLs and products on display on the shop floor; ii) a web application (called here Main App) whose goal is to provide managerial information; and a Price Update Module. Next, we provide a short explanation of each one.

The Mobile App is a tool that will be used by the staff. It involves four functionalities: i) installation of ESLs; ii) product commissioning; iii) searching for products; and iv) visualization of ESL details. The installation process consists of logically linking an ESL to a physical location. In the case of this system, ESLs are installed in physical "modules", which are slots on a shelf. So, every ESL knows the physical location where it is installed. The product commissioning process consists of binding an already installed ESL to a specific product. This process aims to link an ESL with a product which is going to show the price. The searching for products and visualization of ESL details are functionalities that allow the user to obtain basic information about the ESL and the product in which it is showing the price. Particularly, searching for a product allows the user to search for a product in the store; in this case, the ESL of the product will start blinking to facilitate the user to find the product.

The Main App has a managerial role, exposing the following functionalities: i) visualization of all installed and bound ESLs; ii) visualization of the price update process; iii) management of the physical information of the stores; iv) management of loading new ESLs into the system; and v) visualization of Key Process Indicators (KPIs) that provide managerial information for strategic decisions.

The Price Update Module was designed as a subscriber of the company's message broker. So, when new price updates arrive, our module ingests the messages into our system to process them. Each price update message (in Avro format) contains the price for one product, the product code and its description. This is rendered into an image, containing the new product sales price tag and sent to the correspondent ESL on the shop floor via publish/subscribe mechanism using MQTT (Message Queueing Telemetry Transport) over a wireless link of the IoT network. Each ESL uses the same mechanism to send data to the application reporting its state.

Figure 1 depicts the main components of the solution architecture of the system. The Web & App modules provide the interfaces to the ESL solution by encompassing functions for ESL management, ESL/product location, and ESL price update transaction monitoring. In addition, it also provides functions like linking an ESL device to a product as well as auditing and error checking. The requests either from a mobile device or a Web interface are sent to a BFF (Backend For Frontend pattern) module which registers requests using an 'in-memory' database located at the BFF module and then queues each incoming request using a messaging queue (Cloud Pub/Sub) to be processed by a worker module. The reply to each client request from the Web/App will be returned directly from the worker to the client via the BFF module.

Workers provide services to the ESL network such as installation, commission-

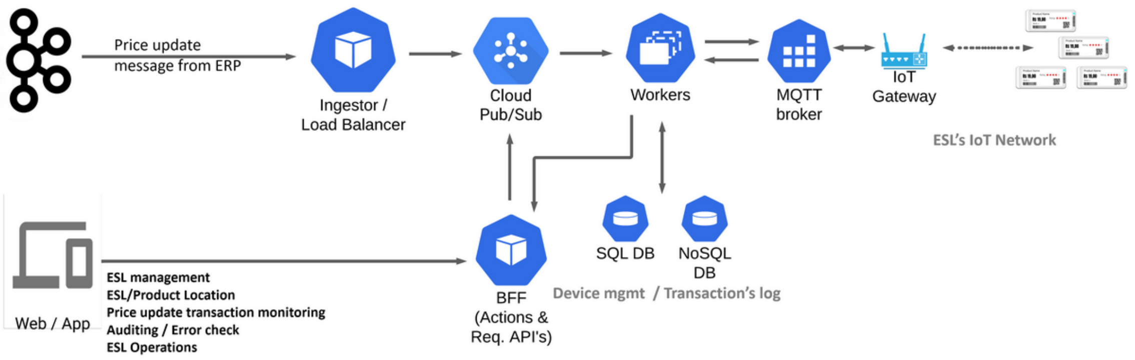


Figure 1. The Architecture of the XPTO ESL solution

ing, state monitoring and routing via an MQTT broker which functions as an application gateway between the IoT network and the main application in the cloud. The ESL IoT network installed on the shop floor communicates wirelessly through an IoT gateway. They make use of an SQL database for these functions.

The ESL transaction processing module ingests and processes each new price update message that arrives from the ERP via a corporate message broker platform (Kafka) by a connector module (Ingestor / Load Balancer) which then queues the transaction to be processed by a worker, through a Cloud Pub/Sub. Price update transactions are processed by the workers resulting in an image containing the new sales price tag which will be sent to the corresponding ESL by using an MQTT broker via an IoT gateway. This asynchronous mechanism is bidirectional (to/from) the ESL IoT network. All the resulting transaction processing is logged in a NoSQL database for tracking purposes.

4. The Approach for Building a SPL

In this section, we briefly describe the steps of our approach towards the creation of the SPL for the ESL domain.

We have followed a four-step methodology: (i) analysis of scientific literature; (ii) analysis of grey literature; (iii) elaboration of a feature diagram; and (iv) reverse engineering and mapping. We describe each step in the following.

4.1. Analysis of Scientific Literature

In this step, we performed a search for scientific literature in digital libraries (ACM, IEEE, and Scopus) to identify studies related to product line development in the ESL domain. We aimed to identify primary studies describing ESL solutions to commonalities and variability in this domain. The second goal of this search was to identify studies that describe the development of an ESL software product line, but we did not find any study addressing this topic.

We performed two types of searches: a free search, and a string-based search. We performed the free search in Google Scholar, Scopus, ACM, and IEEE digital libraries. In this phase, we did not build a rigid search string, we combined terms related to ESLs and Domain Engineering such as Electronic Shelf Labels, Smart Labels, Smart Tags, Electronic Tags, Product Line, Feature Diagram, Domain Engineering, and Variability Management. In the string-based search, we built the string shown below:

(“Software Product Line” OR “SPL” OR “Domain Engineering” OR “Feature Model”) AND (“IoT” OR “Internet of Things” OR “smart devices” OR “Electronic Shelf Label” OR “ESL” OR “Smart Tag” OR “Digital Price Tag” OR “Smart Label”)

We executed the search string in the aforementioned digital libraries. In the end, we selected five primary studies related to the development of ESL solutions. We present a summary of each identified study in the following.

Yin-ping and Wen-rui [Ma and Ma 2017] present the design and implementation of an ESL system, involving the hardware and software parts. The proposed design is based on three kinds of devices: Access Points (APs), End Devices (ED), and Range Extenders (RE). Among them, AP is used to form the network and process the data sent from all the subsets or routes. ED receives the data packages from the AP node, picks up useful commodity information from the packages, and stores it in the FRAM (Ferroelectric RAM) of SCM (Single Chip Microprocessor) so that the data can be shown on the electronic display through the data transmission and processing capacities of SCM.

Shekhawat [Shekhawat 2022a] proposed the use of mobile phones as ESLs to decrease the cost and complexity of the infrastructure required for conventional ESLs. The author raises two challenges in this scenario. The first is how to associate a device with a product and how to update the right content to the right devices. The second is how to set up the initial devices correctly. To address these challenges, his proposal associates the device with a catalogue ID and an SKU (Stock Keeping Unit) ID. Based on that, the only setup needed initially is the store associate selecting the Catalog and SKU ID that needs to be displayed. This information can then be relayed back to the server which can push the right set of content + pricing data for that specific SKU to that device. Catalog information can be sent back to the server using an HTTP-based connection from client to server.

Wang and Hu [Wang and Hu 2013] present the design of a low-power ZigBee-based ESL system. According to the authors, the proposal overcomes the disadvantages of large power consumption, low transmission speed, and high costs of existing solutions. The proposal is concentrated on showing the hardware details of a proposed circuit and also of the routers. They also performed an experiment that showed the ESLs based on ZigBee were more accurate, stronger in anti-interference, and lower power consumption.

Sung [Sung 2015] proposes a smart label platform for a variety of smart labels. The platform implements an association concept between smart labels and related objects using (un)assign, delete, and update operations. The proposed platform consists of smart labels, gateways, a server, and an external system. Smart labels are cheap, small, battery-powered embedded devices equipped with an electronic display and a wireless transceiver. Gateways interact with smart labels wirelessly. The smart label server prepares display data for associated smart labels using associated object information. External applications or systems on the Internet or other networks use “push” or “pull” mode to deliver business-specific object information and association data to smart label platforms.

4.2. Analysis of Grey Literature

In this step, we used Google to search for websites related to ESL solutions. We aimed to find websites of companies that sell ESL solutions to get a better understanding of ESL solutions provided by companies. Based on the analysis of existing ESL solutions, we identified commonalities and variability in the ESL domain.

Table 1 shows the websites of companies we have found that commercialize ESL Solutions. Most companies offer videos and other marketing elements to describe their ESL solutions. From this research, we identified features like "Search Product," which enables integrated LEDs to flash rapidly to highlight products, and "Displaying ESL Information" a functionality in mobile applications that allows for the real-time update and monitoring of prices and product details.

Company	Website
Pricer	`https://www.pricer.com/`
Zkong	`https://www.zkong.com/`
Solum	`https://www.solumesl.com/`
Zhsunyco	`http://www.zhsunyco.com/`
Minew	`https://www.minewtag.com/`
Sunyco	`https://www.zhsunyco.com/`
Vusion	`https://www.vusion.com/solutions/sesimagotag/electronic-shelf-labels/`
Yala Technology	`http://www.yalatechnology.com/`
The last access of all the websites was in July 2024	

Table 1. Websites of Companies Providing ESL Solutions

Expanding our investigation, we examined additional companies such as Zkong and Solum, which contributed to our understanding of the "Screen Type" feature, encompassing various display technologies such as LCD, E-Paper, LED, OLED, and TFT. Similarly, our review of the offerings from Zhsunyco further enriched our feature set, particularly with "Displaying ESL Information" enabling the exhibition of a wide array of information on the ESL screens. Providers like Minew, Sunyco, Vusion, and Yala Technology further contributed to our understanding of features like "Installation/Desinstallation" and "Commission/Decommission" ensuring a comprehensive perspective on the ESL domain.

4.3. Elaboration of the Feature Diagram

Based on our domain analysis, we observed that ESL solutions in general involve three core elements. The first one is a device utilized by staff for installing ESLs on shelves, essential for the operational deployment of the system. The second is a web application, designed for higher-level management, enabling the viewing and managing of the ESL infrastructure. This application plays a crucial role in strategic decision-making and operational oversight, usually encompassing several dashboards. The third pillar comprises the ESL devices themselves, which are the fundamental IOT components for displaying information and integrating with the overall retail environment.

From the collected information, we developed an initial feature diagram illustrated in Figure 2. The diagram is decomposed into three subtrees, each representing one of the core elements of an ESL solution. The first subtree highlights the functionalities and technical aspects of the ESL installation device. The second subtree details the components of the web application, and the third subtree focuses on the diverse specifications of the ESL devices themselves.

One important point is the inter-dependencies (or constraints) among features, as can be seen in the lower part of Figure 2. These inter-dependencies greatly affect the generation of products from the SPL. The selection of a specific feature in a subtree, such as the type of ESL device, may influence the available options in other feature subtrees, like functionalities in the installation device or the web application. For instance, the selection of the "Viewing Product Information" feature implies the selection of the "Commissioning/Decommissioning" feature.

It is important to highlight the feature diagram presented in Figure 2 is an excerpt of the complete model. We selected the most relevant and high-level features for illustrative purposes. The complete diagram encompasses a broader range of functionalities and technical details, which, though not displayed here, are crucial for the construction of a comprehensive ESL solution tailored to specific user needs. The 'ESL Installation Device' subtree captures the initial setup functionalities, with 'MobileAppFunctionalities' ensuring straightforward ESL management and operational efficiency. 'Web Application' enables strategic control and adaptability through deployment options and comprehensive data oversight. The 'ESL' subtree details device-specific characteristics, including connectivity and display technologies, ensuring system longevity and visual clarity. Interdependencies across the diagram emphasize strategic integration for a seamless ESL ecosystem.

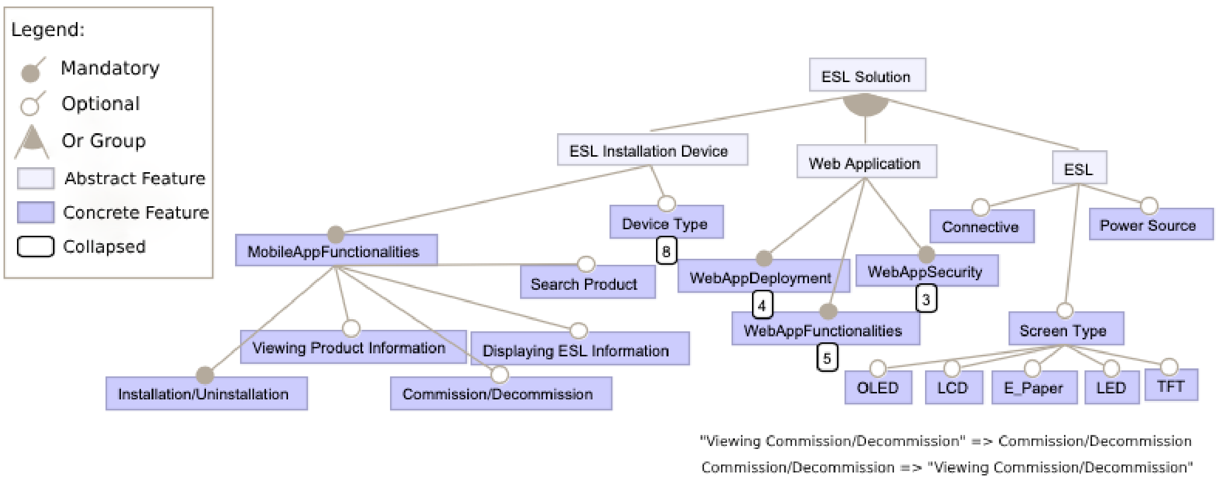


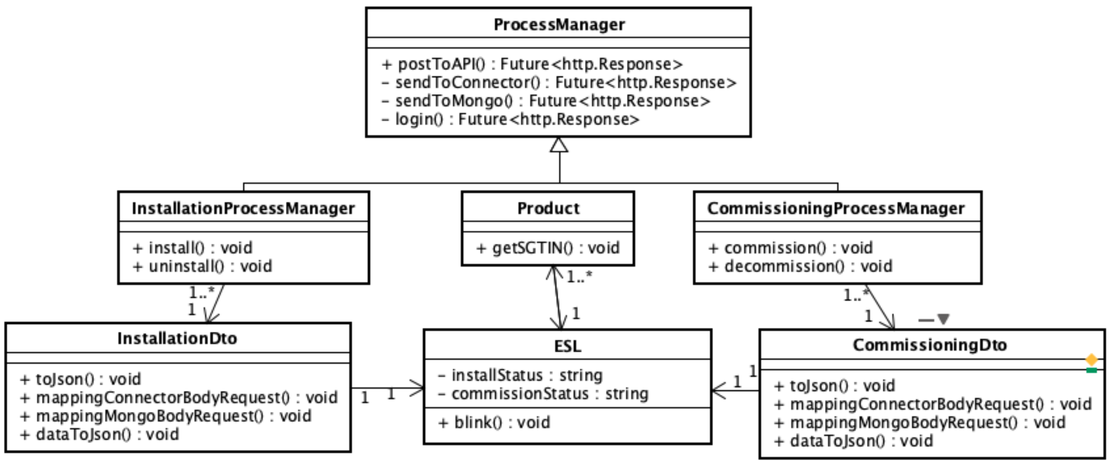
Figure 2. Feature Diagram of the ESL Domain

4.4. Modeling an Existing Product by Reverse Engineering

The main premise of this step is to understand that one existing system of the ESL domain can be seen as a possible product from the SPL that is being created. Therefore, the XPTO system acts here as a possible product from our SPL.

Therefore, we conducted a static analysis of the source code of the XPTO solution and created a UML class diagram, representing the structural view of the system. As previously mentioned, the XPTO system comprises three applications, but up to this moment, we have concentrated on reverse engineering just for the Mobile Application.

Figure 3 shows part of the class diagram of the Mobile App, focusing on the main classes. As can be seen, there are three hierarchies in this diagram. The DTO hierarchy involves classes whose role is to represent data transfer objects for the commissioning/decommissioning and installation/uninstallation features. The ProcessManager



ProcessManager
+ postToAPI() : Future<http.Response>
- sendToConnector() : Future<http.Response>
- sendToMongo() : Future<http.Response>
- login() : Future<http.Response>
InstallationProcessManager
+ install() : void
+ uninstall() : void
Product
+ getSGTIN() : void
CommissioningProcessManager
+ commission() : void
+ decommission() : void
InstallationDto
+ toJson() : void
+ mappingConnectorBodyRequest() : void
+ mappingMongoBodyRequest() : void
+ dataToJson() : void
ESL
- installStatus : string
- commissionStatus : string
+ blink() : void
CommissioningDto
+ toJson() : void
+ mappingConnectorBodyRequest() : void
+ mappingMongoBodyRequest() : void
+ dataToJson() : void
1..*
1

Figure 3. Class Diagram of the Mobile App

hierarchy represents the managers for the commissioning and installation functionalities. Additionally, the class ESL is highlighted as the core of the whole system, representing the abstraction for an Electronic Shelf Label.

4.5. Elaboration of the Mapping Between Features and Model Elements

This step provides the conceptual basis for the creation of products from the SPL. The idea is to map each identified feature of the Feature Diagram to the element of the UML class diagram (Figure 3) so that we can later derive product-specific class diagrams using a software variant management tool, such as Pure Variants [Pure-Systems 2024] . In our case, we elaborated a preliminary mapping table that maps the features to the UML class diagram we have generated from reverse engineering. Therefore, up to this moment, we would be able to generate only the class diagrams for a specific product.

This mapping is shown in Table 2. The first column lists the features and the second column shows the classes and methods that contribute to the implementation of these features. As can be seen in the table and also in the feature diagram (2), there are mandatory features and optional ones. Therefore, one possible product could be composed of the following features: Installation/Uninstallation (mandatory) + Comissioning/ Decommissioning (optional) + Search Product (optional). As long as this mapping is implemented in a tool like Pure Variants, it would be possible to generate a class diagram just with the classes and methods that contribute to these features.

For example, the method *getSGTIN()* appears in the features Commission/ Decommission, View Product Information, and Search Product. This indicates that within the body of this method there is source code that directly contributes to these features. Similarly, the method *blink()* is mapped to the features Commission/Decommission, View Product Information, Displaying ESL Information, and Search Product.

We specified the mappings by performing the following steps:

- We took one feature from the Feature Diagram and understood its role/functionality.
- We searched in the source code a pattern-matching by relating terms to find classes, attributes, methods, or any other source code element that seems to be related to

the feature. As a result of this step, we generate a document with the candidate code elements.

- We then, conducted an interview with a Domain Specialist involved in the development of the XPTO solution to verify if the source code elements are associated with each given feature. As a result, we generate a more refined list.
- We asked the specialist for any other source code element that is related to the feature under analysis.

Feature	**Corresponding Class(es) and Method(s)**
Installation/Desinstallation	**InstallationProcessManager**: install(), uninstall() **InstallationDto**: toJson(), mappingConnectorBodyRequest(), mappingMongoBodyRequest(), dataToJson() **MAC**: getMac()
Commission/Decommission	**CommissioningProcessManager**: commission(), decommission() **CommissioningDto**: toJson(), mappingConnectorBodyRequest(), mappingMongoBodyRequest(), dataToJson() **MAC**: getMac() **Product**: getSGTIN() **ESL**: blink()
View Product Information	**Product**: getSGTIN() **ESL**: blink()
Displaying ESL Information	**ESL**: blink()
Search Product	**Product**: getSGTIN() **CommissioningDto**: toJson() **ESL**: blink()
...	...

Table 2. Mapping Table - Features – Class and Method

5. Conclusion

In this paper, we presented our initial efforts towards the development of a product line for the ESL domain. We developed an initial feature diagram with the high-level features and we have mapped it to classes and methods of a UML class diagram representing one product from de SPL. The results we have presented represent initial steps towards the creation of a SPL for the ESL domain. In order to have an initial version able to generate a operational product, many development cycles must be done.

An interesting aspect of this paper is the approach to creating a product line by considering a system under development combined with a domain analysis. This reflects a real scenario for many organizations, where the initial demand is for developing a single system. Subsequently, the team may identify the opportunity to develop a product line, allowing for the generation of similar products in the future. As a technical part we intend to specify the feature model and mapping variability abstractions (features) to their realization into the ESL class diagram and source code using pure::variants [Pure-Systems 2024] tool to automate the derivation ESL products based on customer configuration choices. Besides, we also intend to consider the possibility of using an Aspect-Oriented Strategy for modelling the SPL [Júnior et al. 2010].

6. Acknowledgements for the Financial Support

We would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES - 001), FAPESP and CNPq 312086/2021-0.

References

Big Lever (2024). Gears: Product line engineering tool & lifecycle framework. Last access in: 12th, July, 2024.

Clements, P. and Northrop, L. (2001). *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional.

Júnior, J. U., Penteado, R. D., and de Camargo, V. V. (2010). An overview and an empirical evaluation of uml-aof: An uml profile for aspect-oriented frameworks. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 2289–2296, New York, NY, USA. Association for Computing Machinery.

Lee, K., Kang, K. C., and Lee, J. (2002). Concepts and guidelines of feature modeling for product line software engineering. In *International Conference on Software Reuse*.

Ma, Y.-p. and Ma, W.-r. (2017). Design of electronic shelf tag system based on simpliciti. *International Journal of Engineering and Applied Sciences*, 4(9).

Meticulous Research (2023). Smart Shelves Market - Global Opportunity Analysis and Industry Forecast (2023-2030). Technical Report MRICT - 1041022, Meticulous Market Research Pvt. Ltd.

Pure-Systems (2024). pure::variants. Last access in: 12th, July, 2024.

Shekhawat, S. (2022a). Decentralized pricing on mobile phone-based esls. In *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 245–249.

Shekhawat, S. (2022b). Decentralized pricing on mobile phone-based esls. In *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 245–249.

Suh, T., Kim, H.-s., Ko, J., Badrinarayanan, V., and Bahk, S. (2018). Electronic shelf labels: Prototype development and validation using a design science approach. *Journal of Information Technology Management*, XXIX:23–38.

Sung, J. (2015). End of paper labels: Emerging smart labels toward internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 216–221.

Wang, Y. and Hu, Y. (2013). Design of electronic shelf label systems based on zigbee. In *2013 IEEE 4th International Conference on Software Engineering and Service Science*, pages 415–418.