Adoption of Software Product Line to a Voice User Interface Environment

Diógenes R. F. Oliveira, Byron L. D. Bezerra, Elyda L. S. X. Freitas, Alexandre M. A. Maciel
Polytechnic School of Pernambuco – University of Pernambuco
Recife, Brazil
{drfo, byronleite, amam}@ecomp.poli.br, elyda.freitas@upe.br

***Abstract* — Software Product Line is a software development paradigm created to meet different market segments. This paradigm has shown great acceptance in the corporate environment (Motorola, Nokia, and Hewlett Packard) to allow the construction of more efficiently through reusing common components applications, besides being extensively researched by academics. The segment of voice interface, in turn, came up with the demand for systems capable of interacting with users, but in the application development process for this domain there is a lack of tools that make the task more productively. The FIVE (Framework for an Integrated Voice Environment) is a development environment for Voice Interface products designed to increase productivity in this segment. This paper aims to apply a SPL approach to FIVE. For this, a comparative evaluation of the process of construction of FIVE and SPL platforms was performed. Then adjustments in order to correct structural problems and, finally, the framework was validated using a set of experiments which sought to ensure the confirmation of such changes have been made.**

Keywords: Experience Report,s Software Product Line; Voice User Interface.

I. INTRODUCTION

In recent years, the area of Voice User Interface (VUI) has received great attention from academics, for two main reasons: first, due to improvements in the performance of automatic speech processing systems, including speech recognition and speech synthesis; secondly, due to convergence device and mass production of multimedia content, which requires means of user interaction faster and efficiency [1].

According to Huang *et al*. [2], the typical architecture for the development of VUI has three components: the first represents the set of engines responsible for the speech recognition or the speech synthesis; the second consists of a API (Application Programming Interface) used to facilitate communication between engines and applications; and the last one consists of a set of possible applications. This architecture has guided this area over the years and many resources have been created with the aim to assist in this process.

Much has been done, both academia and in industry to provide improvements in speech recognition rates and speech synthesis naturalness, however, little effort has been made to bring these advances at the application level. Given this scenario, was developed the FIVE (Framework for an Integrated Voice Environment) in order to assist in speech engines building and in instantiation of them in different technological environments (Telephone, Mobile, SmartTV) [3].

The FIVE has been used by the company Vocal Lab in a real development environment. With him, the time-to-market was considerably reduces and enable the mass development of products with voice interface. The Voc Refactoring the Environment al Lab, offers a products family for speech recognition (*VL Recognizer*), speech synthesis (*VL Synthesizer*) and speaker verification (*VL Identificator*).

According to Pohl [5] Software Product Line (SPL) is a set of software systems that have a certain set of features in common, and meet the needs of a particular market segment or mission and are developed with the same core assets. Although it is not explicit in the original work of Maciel [4], FIVE presents a SPL behavior, however, various features of SPL presents some problems inherent in this approach, as:

- Lack of variability management, which causes a lack of control of altered or removed features;
- Severe failure of the features configuration, generating products with errors and / or locking tool.
- No identification of features, preventing management for maintenance and evolution.

Given these problems, FIVE does not function properly as SPL in all family products. This leads to lost productivity, reducing the potential of time-to-market as suggested by the tool. In this sense, this paper aims to propose the adoption of Software Product Line approach in FIVE. For this, this paper is organized as follows: Section 2 provides a background to the literature of SPL adoption. Section 3 shows adoption process. Section 4 shows the experiments performed for the FIVE and his adaptation to the concepts of SPL and finally section 5 describes conclusions.

II. SOFTWARE PRODUCT LINE ADOPTION

The adoption the SPL concept emerged together with the practice of software reuse. In 1983, Doe and Bersoff [6] presented the software industry an initiative to increase productivity and quality by creating an environment composed of techniques and tools to assist the process of software development with reuse. The literature on the adoption of software product line is enough extensive. Bosch [7] reports the adoption of alternatives is generally much more diverse than those presented in the literature and the technical and

DOI reference number: 10.18293/SEKE2015-185

organizational criteria for adoption have more freedom than we might expect.

This diversity can be seen through an analysis of the major works published in recent decades. Bosch in [7] created a maturity model served as a reference in the evolution of product lines. Clements and Northrop [8] synthesized the fundamentals of SPL, practices and standards used, which provided a model with the essential approach to application of SPL. Linden et al. [9] presents the best practices of the industry in the adoption of SPL using the foundations created by Clements and Northrop, which provide practical actions used SPL processes. Finally, more recently, Apel *et al*. [10] present a features-oriented model for SPL with concepts and practical implementations.

Despite the freedom in SPL adoption, these models have common areas: *Domain Engineering*, responsible for collecting, organizing and storing past experience in systems development activities; and the *Application Engineering* responsible for the process where applications are built by reusing the artifacts of Domain Engineering and by exploration the variability [5]. Given this scenario, this work has been inspired Pohl *et al*. [5] and Apel *et al*. [10] to making the decisions about the necessary actions that would be applied to FIVE environment.

III. ADOPTION PROCESS

Considering the reading of the Hall of Fame of SPL adoption we propose a adoption methodology composed of four steps: Interview with Experts; Evaluation by Inspection; and Refactoring the Environment.

A. Interview with Experts

The purpose of these interviews was to obtain qualitative information regarding conceptual and architectural issues of FIVE as a SPL solution. Three researchers were selected with practical proven of the SPL approach, and by the vast theoretical knowledge through the publication of articles, consultancies and projects.

At this stage it was possible to better understand existing problems in the framework and know the possible actions that could be taken to solve the problem of traceability of artifacts of the tool. Two key points were mentioned by specialists. First one was the need to identify the features that FIVE pretend to present within the domain of VUI. The second point raised was the need to implement a Configuration Knowledge in order to manage the features and the dependencies control. According to experts, these actions guarantee the stable operation of FIVE by defining constraints as the variability of the components of the platform should compose the derivation of products.

B. Evaluation by Inspection

Pohl [5] propose two steps for SPL adoption: evaluation of Domain Engineering and Evaluation of Application Engineering.

1) Evaluation of Domain Engineering

a) Product Management

At this step, we analyzed the original work of Maciel [4] and reports on the design of FIVE had scoped the area of VUI. Tool's market strategy would provide developers, a productive mechanism, with fast learning curve and multi-platform. This observation was confirmed from the many difficulties on the part of developers, which apply in systems the interface models. A more detailed evaluation of the product was carried out, using three basis listed by Pohl [5] to be observed in the phase of product management activities. Table 1 shows the result of the evaluation.

TABLE I. EVALUATION OF PRODUCT MANAGEMENT

Activity	Description	Application in FIVE
Scope of product portfolio	Determines the range of products to be offered	Engines for speech recognition, speaker verification and speech synthesis.
Scope domain	Identifies major functional areas that are relevant to SPL	Pattern Acquisition, Feature Extraction, Pattern Classification and Engine Generation.
Scope core asset	Define the required features of components	Functionalities needed are techniques for feature extraction and pattern classification.

b) Requirements Engineering Domain

In FIVE, requirements engineering domain were made through the use of questionnaires, interviews and surveys with VUI experts. Thus, the notation used for specifying the specific language was the domain of VUI. Based on the identified functional areas, process was a questionnaire made available in order to meet the difficulties, the needs, the environment, suggestions, among other aspects of the process of construction and application of models of voice interface. Table 2 shows the main requirements raised.

TABLE II. EVALUATION OF ENGINEERING DOMAIN

Scope Domain	Requirements
Pattern Acquisition	Features integrated audio recording and edition.
Feature Extraction and Pattern Classification	Provide mechanisms for analysis and comparison of techniques.
Engine Generation	Platform independence and ease of integration with applications.

c) Design Domain

The general architecture of FIVE was designed independently based on three modules: CORE consisting of a framework of classes where the central implementation of the framework; API that is a proprietary implementation that provides a set of resources needed to mediate between the speech engines and the application layer; and the GUI (Graphical User Interface) which is a graphical interface that assists the development of projects.

Although the proposed architecture meets in an adequate manner the generation of products, a failed architecture was identified. In Engine Generation step all features are loaded to the end product, with no differentiation in architecture on unused features.

d) Realization of the Domain

Previous study evaluated the implementation of FIVE that the best approach to implementing a mechanism for mass

production voice interface, according to know requirements would be through the mechanism Framework Gray-box [11].

The class diagram is centered on the Project Class that relates to the classes: Utterance, Sample, Speaker, Extraction, Classification and ProjectType. All these classes have their methods of adding, updating, deleting and research, except ProjetcType which is just a class of type Enum. The Utterance class has a special behavior to receive the grapheme-phoneme of the utterance of helper classes Phrase, Word and Syllable are responsible for representing the linguistic details of each phrase. They make common variability in code level serving different products and product-specific aspects are addressed from use of the inheritance mechanism.

2) Evaluation of Application Engineering

The environment for generation of products FIVE is a platform that has the format wizard that has sequential tabs, the classic process of recognizing patterns defined by Duda et al. were inspired. [12]. The user-friendly interface, how they are arranged the information, the ease of applying the techniques of feature extraction and classification, and the definition of parameters, all these criteria together, decrease the learning curve in the generation of models for VUI applications.

For this evaluation were built several engines and it was observed that even with the use of the framework mechanism for implementing the variability of the components, it hurts not enough for a proper functioning of FIVE as a SPL. Thus, taking into account the background regarding the adoption of SPL and interviews with experts, if make know that other activities should be performed: the construction of the Feature Model and a Knowledge Configuration.

In the Feature Model to identify the features available, its constraints and dependencies occurs, and from it is possible to know the potential variability of the platform. Configuration Knowledge has the role of expressing the relationships and dependencies between the variables of the product line and features, and their interactions. Thus, observed the need of carrying a refactoring of FIVE environment given that, in its initial implementation, the proposed fast generate products was achieved, but there is a lack of the integrity of the products and the operation of the line from of features selected.

C. Refactoring the Environment

The process of FIVE refactoring the environment in two steps: first the Feature Model was developed and then implemented a mechanism of Configuration Knowledge.

1) Development of the Feature Model

The development of the Feature Model was accomplished with the aid of pure::variants tool [13]. The choice of this tool was made because it is widely used in both academia and industry. From the evaluation of all features FIVE and its dependencies were identified. Some numerical features were created in order to empirically parameter settings are adopted in the area of voice interface.

The Feature Model, which is attributed as the main element of the project itself FIVE contains five features as direct daughters, three of which were considered more complex due to the number of variations, specifically Rating Standards and Feature Extraction, where various techniques are implemented and the internal variations caused mainly by setting parameters. Features like numbers are justified because of being parameters adopted in the literature for specific techniques.

2) Implementation of Configuration Knowledge

The development of the Knowledge Configuration happened according to the proposal of Domain-Specific Modeling of Cyril [14]. According to him the use of domain specific abstractions tends to facilitate the understanding of the variability, this mechanism has been used implicitly in FIVE, even to the point of an individual with experience in VUI applications do not need to run the platform.

According to the results of the evaluation of five to failure dependence between the features is the main problem in the platform. For example, given an "A" feature selected at a time, necessarily requires a "B" functionality later for maintenance operation. Thus, the implementation of a knowledge configuration meets solve this dependency failure.

The development of the Knowledge Configuration was done through crosstree constraints following three phases: change in GUI, register control and inclusion of features extracted from the field and ranked. In the original version of FIVE, the graphical interface allowed the indiscriminate selection of techniques for extracting's characteristic, regardless of the technique of pattern classification. To resolve this problem with a new interface control features available for selection was developed.

The control record of the features was necessary because the original version of FIVE, Each new selection of features for feature extraction, the data were overwritten, not allowing to have a history of previous features. To mitigate this problem adaptation was performed for selected features were stored in the corresponding techniques of extraction of features selected subdirectories.

The inclusion of the field extracted and classified was necessary as the original version of FIVE only the last feature extraction of selected features could be used for classification. To mitigate this problem it was tailored to data structure for addition of a new Boolean attribute (extracted). This solution allowed the use of any features extraction that are available.

IV. EXPERIMENTATION WITH DEVELOPERS

In this study, an observational assessment of the use of FIVE original and the new version after refactoring, followed by the application of a questionnaire was carried out. The purpose of this evaluation was to assess the implementation of FIVE experiencing the potential variability of products, specifically variants of techniques used in generating speech engines, both feature extraction as the classification of patterns.

The experiment with the collaboration of five developers who had no prior knowledge about the FIVE, however, all were familiar with the process of pattern recognition. The developers used the FIVE in the environment composed by Windows 8 operating system with as NetBeans with Java 7 Update 45 operating system, with all the default settings. FIVE were present in all the features of the production line required to build a product.

Observational assessment began with the realization with an orientation about the concept of SPL and it is the FIVE within the context of VUI, to equalize the knowledge of users. Then were distributed both versions of FIVE and requested the construction of a speech recognition engine for isolated words. Then a database of audio and text with five control commands (Open, Close, Follow, Stop, No) was available. The developers were free to choose the features for feature extraction and pattern classification. At the end, everyone was able to successfully generate the engines in both versions.

During the FIVE observational assessment metrics were used: time (in minutes), number of turns to earlier stages, the tool crashes, errors and doubts. Tables 3 and 4 present the results observed in two scenarios: evaluation with the original version, and evaluation with new version, respectively.

TABLE III. EVALUATION WITH THE ORIGINAL VERSION

Developer	Time	Turn Back	Crashes	Errors	Doubts
A	34	7	6	3	9
B	44	6	8	3	9
C	38	8	6	1	10
D	42	9	7	2	11
E	43	10	6	3	9

TABLE IV. EVALUATION WITH NEW VERSION

Developer	Time	Turn Back	Crashes	Errors	Doubts
A	26	7	0	0	6
B	35	7	0	0	8
C	31	5	0	1	6
D	32	7	1	0	7
E	28	9	0	0	6

It is observed that the average for the construction of speech in scenario 2 engine time was 25% faster than in scenario 1, Although, the generation of speech remained in a short period of time engines. The number of turns was similar in both scenarios. The crashes were practically used, since their occurrence occurred due to the absence of Configuration Knowledge, specifically in the areas of feature extraction and pattern classification. The crash that occurred with the user D in scenario 2 was due to internal problems with the operating system. Errors in scenario 2 were reduced because the account Configuration Knowledge and doubts about the features decreased smoothly in scenario 2 since at that time there was already a greater familiarity with the tool.

V. CONCLUSIONS

A major strength of this study was the exploration of the SPL approach in the field of VUI, since this area is little explored by Software Engineering. Another important contribution was the refactoring of FIVE to make it really a SPL. With this the FIVE passes to carry around a set of values, among them, the possibility of development of research techniques of extraction and classification, because reading from the perspective of oriented features.

The correction of faults in the functioning of FIVE process, through the Knowledge Configuration and Feature Model solved the problems found in the previous version by defining the constraints of the features. The identification of features and construction of the model feature that provides visualization and potential of the platform.

In the experiments the features were willing to users only in accordance with the availability of the same features as the previously chosen, proving the importance of Configuration Knowledge for the correct operation of the platform and product generation correctly. After the restructuring, the FIVE happened to have a clear definition as to its engineering software, making their understanding for researchers and developers easier.

VI. REFERENCES

[1] Kurniawati, E.; Celetto, L.; Capovilla, N.; George, S., "Personalized voice command systems in multimodal user interface," Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on , vol., no., pp.45,47, 12-14 Jan. 2012

[2] Huang, X., Acero, A., Hon, H.W., Spoken Language Processing – A Guide to Theory, Algorithm, and System Development, Prentice Hall, 2001.

[3] Maciel, A.; Carvalho, E.; FIVE – Framework for an Integrated Voice Environment, IWSSIP, 2010.

[4] Maciel, A., Investigação de um ambiente para o desenvolvimento integrado de interface de voz. Tese de doutorado, CIn/UFPE, 2012.

[5] Pohl, K.; Böckle, G.; Van Der Linden, F.: Software Product Line Engineering – Foundations, Principles, and Techniques. Springer, Heidelberg 2005.

[6] Doe, D. D. and Bersoff, E. H. (1986). The Software Productivity Consortium (SPC): An industry initiative to improve the productivity and quality of mission-critical software. Journal of Systems and Software, 6(4), 367–378.

[7] Bosch, Jan. Maturity and Evolution in Software Product Lines: Approaches, Artefacts and Organization. Software Product Lines, 257-271,2002,Springer Berlin Heidelberg

[8] Northrop, L. M. e Clements, P.C. A Framework for Software Product Line Practice. Version 5.0. Pittsburg. Software Engineering Institute, 2007. Disponível em: < http://www.sei.cmu.edu/productlines/framework.html >. Acesso em: 02/12/ 2013 às 10:24.

[9] Linden, Van der; Frank J., Schmid, Klaus; Rommes, Eelco. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag Berlin Heidelberg, pp. 26, 2007.

[10] Apel, S., Batory, D., Kstner, C., and Saake, C. Feature-Oriented Software Product Lines: Concepts and Implementation. Springer Publishing Company, Incorporated. 2013.

[11] Fayad, M. E.; Schimidt, D. C.; Johnson, R. E. Building Application Frameworks: Object-Oriented Foundations of Frameworks Design. New Jersey: Wiley, 1999.

[12] Duda, R.O., Hart, P.E., Stork, D.G., Pattern Classification, Wiley-Interscience. 2000.

[13] pure::variants, available em: <http://www.pure-systems.com/> Acessado em Março de 2014.

[14] Cirilo E., Nunes, I.; Kulesza, U.; Lucena, C.; Automating the product derivation process of multi-agent systems product lines. No SBES '09, página 12, Brasil, 2009. IEEE.