JOURNAL OF INFORMATION SYSTEMS ENGINEERING & MANAGEMENT

Journal of Information Systems Engineering & Management, 2(2), 11
ISSN: 2468-4376

lectito

Software Residence Application in the Versions of a Software Product Line

Alexandre L'Erario[1]*, José Augusto Fabri[1], José Antônio Gonçalves[1], Alessandro Silveira Duarte[1]

[1] *Federal University of Technology - Parana*, BRAZIL

***Corresponding Author:** alerario@utfpr.edu.br

Citation: L'Erario, A., Fabri, J.A., Gonçalves, J.A. and Duarte, A.S. (2017). Software Residence Application in the Versions of a Software Product Line. *Journal of Information Systems Engineering & Management*, 2(2), 11. doi: 10.20897/jisem.201711

Published: March 30, 2017

ABSTRACT

This article presents an experience in software residence, addressing jointly, software product line and version control process. A residency program in software is conceptually similar to the medical residency programs, which aims to train professionals in an area/specific activity. A real scenario was presented to three groups of master's students who had a mission to develop and specify a process to meet a specific problem. As a result of this experiment, we developed the process and the software of residence contributed to knowledge on the scene was internalized and disseminated among the participants.

Keywords: software residency, product line software, control system version

INTRODUCTION

In Brazil, a limited number of organizations producing software with certified quality standards for a model. The number of companies certified with CMMI in Brazil is 166, while in 5153 China, the United States in 2053 and India in 1059, such data can be found at https://sas.cmmiinstitute.com/pars/pars.aspx.

Factors arising from the high tax burden, plus the qualified labor shortage in software engineering area constitute an inhibitor scenario for Brazil's expansion process in software production industry. In order to train professionals in the sector, generating highly skilled labor, some companies and universities have joined the concept of residence in software (something similar to medical residency).

This work is an approach to residence in software, whose scope is focused on the product line with the version control system. Versioning arctifacts is a common practice in organizations, however, in this study, the approach incorporates the concept of software product line, increasing the complexity of residence and making it closer to the real environment (L'Erario, Fabri, Goncalves, & Duarte, 2016).

The objective of this work is to use the concept of residence in software for a group of master's program students can solve a problem related to product line of software and version management.

This problem was identified in a real company and led to the students by the GTI research group UTFPR. Teachers of the Federal Technological University of Paraná software engineering (UTFPR) created the Group of Information Technology Management (GTI). The group aims to provide directions in the process of improvement area. Currently the group consists of 6 teachers working actively with the software production companies, the software deployment process, assisting them in evaluations of CMMI, ISO and MPS-BR models.

LITERATURE REVIEW

Residence in Software

The residency idea was created in Brazil by decree number 80281 of 05 September 1977. This is characterized as a graduate teaching mode and aims to improve health in the form of a specialization course in which it is inserted into an organization's health.

The residence in software follows the same line of reasoning: to provide a real experience to the student graduate in this way promotes the dissemination of the concepts of quality, production process and management projects in the software engineering field.

Second (Fabri et al., 2010) in residence provides software in a real environment of software development, practical experience to undergraduate students, graduate and / or professionals involved in the industry. These real environments are prepared for residents to specialize a given area.

As in medical residency, second (Sampaio et al., n.d.), residence in software, should be developed into a center of learning, evolving expertise, relevant concepts and present formal educational features. The development of practices employed in the software industry should guide the attention of the tutors responsible for the residents.

For the authors (Fabri et al., 2010; L'Erario et al., 2016; Sampaio et al., n.d.), the software house has a resemblance to the residency, because both have the same purpose: specialize students / professionals interested in a particular area.

To (Silveira Duarte, L'Erario, Domingues, & Fabri, 2013) the residence of the execution environment can be classified into levels that start in fully simulated to actual execution environment. This paper presents a partially simulated environment that is a real problem a company has been imported into the academic environment, providing students with a problem / scenario identical to the real.

Software Product Line

According (Mahmood & Oxley, 2010) , the lines of software product (Software Product Lines -SPL) are used to increase productivity , improving the quality in a short period of time. The artifacts produced in this concept are reused in a systematic way into products. Such artifacts are defined as core assets.

Development guided by SPL, the **core assets** include the artifacts developed during the production process, such as requirements documents, architecture, use cases, code, and other (Alzahmi, Abu-Matar, & Mizouni, 2014; Mahmood & Oxley, 2010). Many products can be derived from the major active. These products are distinguished of the others given its characteristics (Thao, 2012).

SPL focuses on the systematic use of reuse and the concept is essential to the existence of a software product line (L'Erario et al., 2016; Murugesupillai, Mohabbati, & Gašević, 2011). To (Mahmood & Oxley, 2010) SPL should promote a mechanism to track changes and new versions of artifacts and products emerging from the baseline. SPL must ensure that it monitors the line as well as the propagation of changes.

Version Control Systems

Second (Ghezzi, Wursch, Giger, & Gall, 2012) the actual concept of a version control system and its first implementation was introduced by (Rochkind, 1975). In this publication, the author called the source code control system as a tool to assist the project schedule controlling changes in the source code. In this way it would be possible to store, update and retrieve all versions of all there allocated codes.

Currently the version control systems have features that allow remote access to process assets, as well as the possibility of co-developers work on the same project. You can identify which changes were generated by which developers and compare changes made to the user's computer and also the repository.

The version control tools are integrated into the developer environment. You can use the IDE to write part of your project (source code) with a text editor to modify / create a requirements document for example. All assets created would be stored in the version control system, whether centralized or distributed.

Such a concept supported by a set of tools arouses interest from researchers to analyze and optimize their use with respect to software production, as is the case of (An, Khomh, & Adams, 2014; Kamei, Ohira, Hassan, Ubayashi, & Matsumoto, 2014; Malhotra, Pritam, Nagpal, & Upmanyu, 2014).

METHODOLOGY

The methodology used in this research was experimental, guided by (Wohlin et al., 2012). In this case the variables are known and the final results are analysis of the change of values of these variables.

The objective of this experiment was to determine whether the knowledge gained from a residence process was enough that a group of students could solve a problem arising from the real world.

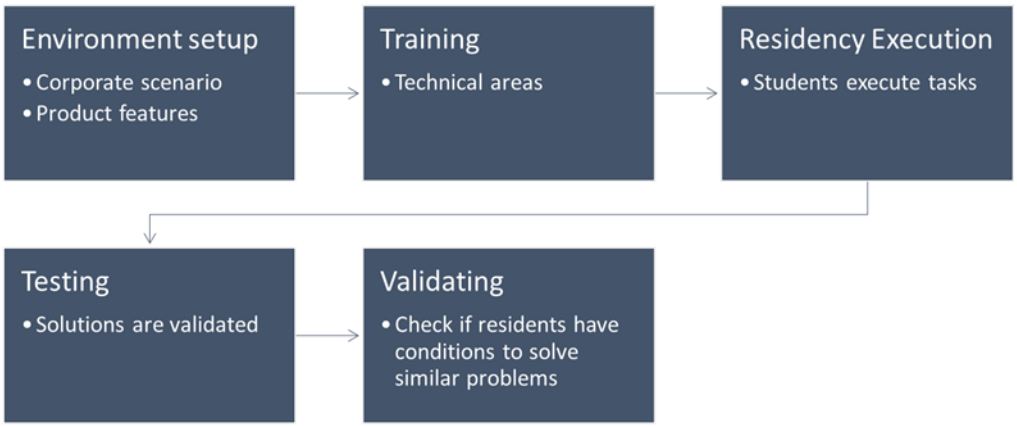

Figure 1. Implementing steps.

Figure 1 represents the set of steps to carry out this work. It is important to note that the implementation of residence, here treated, addresses stage in which students must solve the above problem. The test is the validation mechanism of the solution while validation, characterized as the last step, you should check that the skills and knowledge acquired by residents in relation to similar problems.

Defining the Problem Scenario

The group of teachers GTI / UTFPR identified in a consulting company with a critical problem of configuration management. The problem was solved and the solution was successfully deployed.

Although authorization to disclose the problem, the company has not authorized the disclosure of his name and for this reason is not identified in this article. This scenario is typical and can be considered as the applicant in several companies in the sector.

They were conducted to students two aspects of this scenario: the characterization of the company and a product with the same architectural properties. The original product handled by GTI is the organization 's domain and for this reason a minimalist version, but with the same architectural significance was conducted to the residence. The difference between the original product and the product led to the residence was the implementation of the number of use cases. The product treated by the students implemented only a single use case representing a record, whereas original product implements some 70 cases, in addition, the registers.

The company in question operates exclusively in the software development industry, has a team of 10 developers and has a product here called P. The P product is the company's flagship product, is deployed in many customers and represents a significant portion billing.

The product in question was developed by a software architect who is no longer in the organization. In this case the knowledge regarding architecture, organization domain, restricted to only some documentation available.

For the company, the product P can be sold to a customer in its original form, though there are customers who need customization. The customization of a customer cannot meet each other, in addition, new updates (improvements or bug fixes) may be made available to all customers and new versions will be released, the latter on the form of contract negotiation. In real business, to customize the product to a particular customer the company spends about two weeks.

The product distribution structure for its customers, according to the company, is explicit in **Figure 2**. From a baseline is created a first version of the product, called PoV1 Product (Original Version 1). This is the software product ready for distribution. The PoV1 distribution in the case of **Figure 2**, has been delivered to customers 1 and 2. In the case of customers 3,4 and 5, it was necessary to perform a customizing the system, with the first personalized, represented by the rectangle Pd1 (Product Distribution 1) met the clients 3 and 4 while other customization, Pd2 (Product Distribution 2), attended the customer 5.

Figure 2 shows how the company distributes its products to its customers. In a real scenario the number of customers is much more significant and not just 5 as shown in the figure. Each client has an independent installation of the product.

Bug fixes or improvements to the product can be deployed. In this case when there is a modification in its original version all nodes associated with this need to receive such updates. In **Figure 2**, for example, if a modification is implemented in PoV1, customers 1 to 5 are also updated.

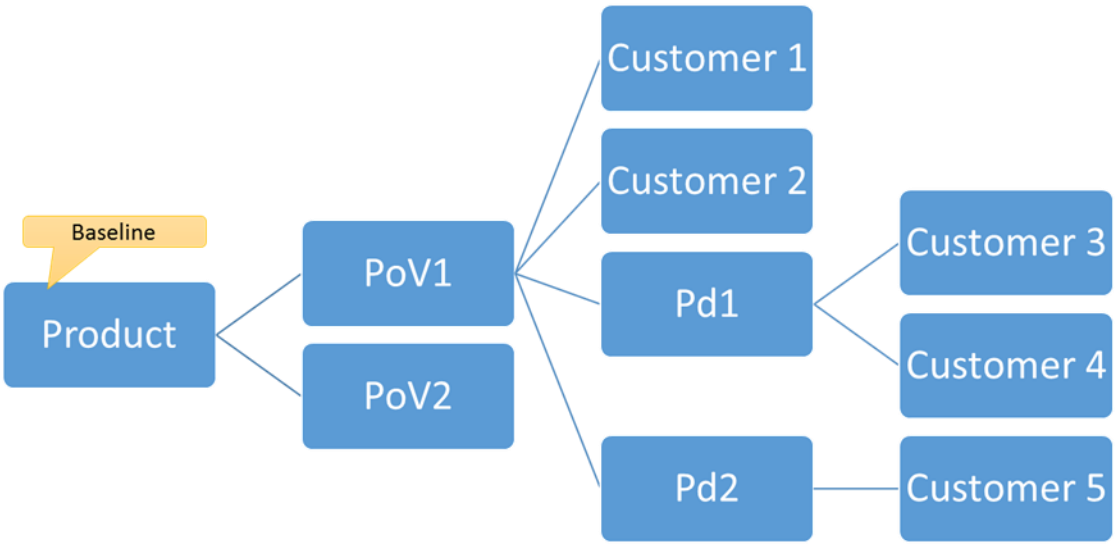

Figure 2. Structure distributions covered by company.

Table 1. Research protocol.

Context	Residence in software, simulation of a real problem, obtained from the software industry.
Purpose of residence	Enable residents to treat a software product line with multiple distributions and versions.
Participants	Tutor and students of informatics graduate program.
Selected population	12 students of the program, all active in the software market, with at least 2 years of experience. Students were divided into groups of 4 participants.
Generated artifacts	A versioning structure and a report containing the specification of the processes required
Analyzed variables	Based on both generated artifacts the following issues were discussed: the ability to create new distributions, creating new versions, the creation of new updates and the creation of new features.

New versions of the product will be launched. In **Figure 2**, represented by PoV2, ie original product 2. There may be migrating from a client that is in version 1 for this, but it depends on the contractual arrangements between the company and its customer.

After presenting this scenario the following set of questions was conducted for the students:

a. How new developers start their work?
b. How new features can be implemented and deployed?
c. How specify configuration/versioning processes for this scenario?

Research Protocol

The protocol indicates that the activities performed by the researcher during the operation of the research. **Table 1** contains the description of the research protocol elements used here. It is noteworthy that in this procedure the experimental test (preliminary execution of the experiment in an attempt to validate the research process) was not run by GTI experience in residence in software (Fabri et al., 2010; Silveira Duarte et al., 2013) and also the experience of selected population, already active in the market.

At the end of the implementation of residence, the students delivered a report and also the versioning structure. The final analysis was drawn from artifacts (report and versioning directory) delivered by the teams.

Variables Monitored

Through the protocol, presented in **Table 1** together with the generated artifacts, it was possible to see how each team developed an solution to the problem. It is important to note that the results presented by the teams were analyzed individually.

There were two artifacts generated by each of the teams. The first one is a directory structure referring to the version control system. In this case, all they used Subversion. In this directory each team should create a basic working structure, which later in the testing activity, should, along with the other students allocate 5 distributions for 5 alleged clients as illustrated in **Figure 2**.

Table 2. Variables.

Problem (creating..)	Definition
New Installation	Created when a new customer is captured by the company. A new area in versions environment should be created.
New distributions	Created at the time that an identical copy of a product serves more than one client.
New updates	A bug or an improvement was detected. In this case it must be propagated to all clients in this release.
New features	It is deploying a new feature in an existing version and propagates it to all customers (distributions) in this release
New versions	A new generation of the product is created. Customers can migrate from earlier to later in accordance with contractual terms.

The second artifact is a report that should contain a set of specifications processes described in BPMN (Business Process Model and Notation), so that the entire development team was able to create new versions, distributions and product updates, and contemplating the beginning of new developers in the company.

The variables treated in this study are explained by **Table 2**. The first column refers to what problem the solution proposed by the group must solve, while the second description.

The variables listed in **Table 2** were measured in terms of yes or no. That is, the solution proposed by the group can meet a variable or not. Furthermore, three scales were used to measure. The ability to solve this problem automatically, partially automatic or manually.

Solve an automatically problem means that with the execution of a problem script will be resolved, partially automatic occurs when the developer process needs to run a set of scripts and manually when the developer needs to intervene with the software to verify and validate the action.

THE SOFTWARE RESIDENCY EXECUTION

The implementation of residence followed the steps outlined in **Figure 1**. The systematic implementation of these steps ensured that the residence was executed successfully and especially the knowledge was generated and disseminated among residents.

Creation Environment Process

In this first stage, the environment and also the problem presented were created and configured. The creation of such implied real case of adaptations within the university, featuring a simulation. While driving the organization's characteristics in question (size, number of employees and customers) into the residence is plausible, the product studied had to be adapted. In this step, a new version was created, following the precepts referring to the original architecture, so creating an imaginary product, but with the features of the original.

Training

As learner's evening, a training about 8 hours was jointly developed. It is noteworthy that the participants were already active in the market and for this reason the training has been optimized for such a team, since everyone already had some knowledge in Java and mainly in object orientation. This training covered subjects: JavaServer Faces, JPA, Subversion and BPMN.

The Implementation of Residence

After training, this step, which lasted 8 hours, the scenario presented, along with the problem and aimed to lead the groups to resolve tax problems.

The groups presented two versions of the report / process prior to the end of 8 hours. At the end, they delivered the specification of processes in BPMN, along with versioning directory Subversion. The teams had three days to prepare a report on such cases without the change this to make all documentation. At this stage the groups had access to documentation and also to the application source code.

Test

At this stage, 5 alleged customers were created and each team had to demonstrate in seminar format as would be the solution of the problem. The created customer match the structure shown in **Figure 2**, that is, customers who had the product P would be made in an original way, and also customization, represented in distributions. It was also exposed to possible migration of the first version to a second version of the clients.

Table 3. Results Group 1.

meets variables		No	Yes 1	2	3
new	distributions				X
	updates			X	
	Features		X		
	versions				X

Table 4. Results Group 2.

meets variables		No	Yes 1	2	3
new	distributions				X
	updates		X		
	Features	X			
	versions				X

Table 5. Results Group 3.

meets variables		No	Yes 1	2	3
new	distributions				X
	updates			X	
	Features		X		
	versions			X	

Validation

After the seminar, based on the delivered documentation, a study on the validation of the solutions was performed. In this study it was found that the artifacts generated by the groups met the needs presented in the previous section.

Validation simulated faithfully the events in real company. In this regard, it was considered that new distributions are created from the original version. For this, a copy of the source code of this version is copied to a new directory, whose name refers to customer identification. In addition, new features in this product, it means the need to implement new classes / methods and consequently a new interface or segment thereof. Once tested, the new functionality should be propagated to all the distributions of a given version. Moreover, in this case, the database can be modified with the addition of new fields.

The corrections made in the original version are also propagated to all distributions.

For each variable analyzed, based on the documentation submitted, two possible values were defined: yes / no. If a variable is taken into consideration (value yes), there are 3 levels at which this was measured.

Level 3 indicates that with just a single command or script execution, the variable has been met. For example, when creating a new distribution with the execution of a command line the group created and copied the original source code to a directory whose name refers to the customer.

Analogously, but using documentation to meet the variable, you need to run more than one command line / script, so has the level 2. Level 1 it is a manual solution, in which the developing need to understand concepts and processes before running a larger set of commands, in addition to a distribution to another, these commands can vary and human intervention is required to confirm changes.

Table 3, **Table 4** and **Table 5** report the individual results obtained from the analysis of the three groups used in this experiment. It is noteworthy that a better explanation of the variables is shown in **Table 2**. Problems related to distributions and facilities in new customers were treated equally in all groups.

CONCLUSIONS

This paper presented an experiment conducted to address issues relating to product line and version control system software residence context.

Therefore, we selected three groups of IT professionals (graduate students) from Federal Technological University of Paraná.

From the results it was found that the residency program was effective and made the groups could solve the stated problems. Individually analyzing the results, wenote that no group failed to meet the problem of creating a new distribution (or new installation), ie the delivered documentation, along with the directory structure fully contemplated this variable.

The relevant results to update could not be partially automated by a group (group 2). In this case, so a bug was corrected by adopting this solution, the company would need human intervention to confirm the distribution correction manually. The same happened with the functionality, in addition, it is impractical in the case of group 2 (**Table 4**). New versions can be created in an automated manner by groups 1 and 2, the same does not occur with the group 3.

The concern to automate these tasks is an eminent problem of real case presented. With a significant number of customers (150) becomes unfeasible perform such tasks manual way, with full human intervention.

It was found that the automation of procedures analyzed here may mean a reduction of work / time developers; however, there is not always a stable solution that you can rely entirely on created scripts. Such a problem is prominent, for example, when a new feature is created and all distributions are updated without human intervention. In a great setting, this problem could be solved only with a script, however, the differences between the specific features of each distribution make it unfeasible.

Even approaching a great result, that is, without human intervention, operations that update the product distributions and insert new features still need human intervention. Group 1 was the group that approached such a result.

At the end of the experiment other issue was addressed, it is the continuous build distributions. As future work is expected to the end of an automatic action, build and test routines are triggered for distributions, so ensuring that their basic features are in working order. This approach will be treated as the subject of a nearby residence experience in software.

REFERENCES

Alzahmi, S.M., Abu-Matar, M., and Mizouni, R. (2014). A Practical Tool for Automating Service Oriented Software Product Lines Derivation. In *2014 IEEE 8th International Symposium on Service Oriented System Engineering* (pp. 90–97). IEEE. http://doi.org/10.1109/SOSE.2014.16.

An, L., Khomh, F., and Adams, B. (2014). Supplementary Bug Fixes vs. Re-opened Bugs. In *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation* (pp. 205–214). IEEE. http://doi.org/10.1109/SCAM.2014.29.

Fabri, J.A., L'Erario, A., Begosso, L.R.C.R.C.R.C., de Lima, F. C., Begosso, L.R.C.R.C.R.C., and de Lima, F.C. (2010). Implementation of Software Residency at a graduation course. In *Frontiers in Education Conference (FIE)*, 2010 IEEE (p. F1H–1–F1H–6). inproceedings, IEEE. http://doi.org/10.1109/FIE.2010.5673498.

Ghezzi, G., Wursch, M., Giger, E., and Gall, H. C. (2012). An architectural blueprint for a pluggable version control system for software (evolution) analysis. In *2012 Second International Workshop on Developing Tools as Plug-Ins (TOPI)* (pp. 13–18). IEEE. http://doi.org/10.1109/TOPI.2012.6229803.

Kamei, Y., Ohira, M., Hassan, A.E., Ubayashi, N., and Matsumoto, K. (2014). Early Identification of Future Committers in Open Source Software Projects. In *2014 14th International Conference on Quality Software* (pp. 47–56). IEEE. http://doi.org/10.1109/QSIC.2014.30.

L'Erario, A., Fabri, J.A., Goncalves, J.A., and Duarte, A.S. (2016). Control version system process anxd software product line: Software residence experience. In *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). IEEE. http://doi.org/10.1109/CISTI.2016.7521368.

Mahmood, A.K., and Oxley, A. (2010). A proposed reusability attribute model for aspect oriented software product line components. In *2010 International Symposium on Information Technology* (pp. 1138–1141). IEEE. http://doi.org/10.1109/ITSIM.2010.5561503.

Malhotra, R., Pritam, N., Nagpal, K., and Upmanyu, P. (2014). Defect Collection and Reporting System for Git based Open Source Software. In *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)* (pp. 1–7). IEEE. http://doi.org/10.1109/ICDMIC.2014.6954234.

Murugesupillai, E., Mohabbati, B., and Gašević, D. (2011). A preliminary mapping study of approaches bridging software product lines and service-oriented architectures. In *Proceedings of the 15th International Software Product Line Conference on - SPLC '11* (p. 1). New York, New York, USA: ACM Press. http://doi.org/10.1145/2019136.2019149.

Rochkind, M.J. (1975). The source code control system. *IEEE Transactions on Software Engineering,* SE-1(4), 364–370. http://doi.org/10.1109/TSE.1975.6312866.

Sampaio, A., Albuquerque, C., Vasconcelos, J., Cruz, L., Figueiredo, L., and Cavalcante, S. (n.d.). Software test program: a software residency experience. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* (pp. 611–612). IEEe. http://doi.org/10.1109/ICSE.2005.1553611.

Silveira Duarte, A., L'Erario, A., Domingues, A.L.D.S., and Fabri, J. (2013). Proposal of a model to classify software residency environments. In *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on* (pp. 1–6). Lisboa.

Thao, C. (2012). Managing evolution of software product line. In *2012 34th International Conference on Software Engineering (ICSE)* (pp. 1619–1621). IEEE. http://doi.org/10.1109/ICSE.2012.6227224.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., and Pfleeger, S.L. (2012). Experimentation in Software Engineering. *Advances in Computers* (Vol. 44). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-29044-2.

