Evaluating Lehman's Laws of Software Evolution within Software Product Lines: A Preliminary Empirical Study

Raphael Pereira de Oliveira[1,2], Eduardo Santana de Almeida[1], and Gecynalda Soares da Silva Gomes[1]

[1] Federal University of Bahia, Campus Ondina
Av. Adhemar de Barros, s/n, 40.170-110, Salvador - Bahia, Brazil
[2] Federal Institute of Sergipe, Campus Estância
Rua Presidente João Café Filho, S/N, Estância - Sergipe, Brazil
{raphaeloliveira,esa}@dcc.ufba.br, gecynalda@yahoo.com

Abstract. The evolution of a single system is a task where we deal with the modification of a single product. Lehman's laws of software evolution were broadly evaluated within this type of systems and the results shown that these single systems evolve according to his stated laws over time. However, when dealing with Software Product Lines (SPL), we need to deal with the modification of several products which include common, variable and product specific assets. Because of the several assets within SPL, each stated law may have a different behavior for each asset kind. Nonetheless, we do not know if the stated laws are still valid for SPL since they were not yet evaluated in this context. Thus, this paper details an empirical investigation where four of the Lehman's Laws (LL) of Software Evolution were used in an SPL industrial project to understand how the SPL assets evolve over time. This project relates to an application in the medical domain developed in a medium-size company in Brazil. It contains 45 modules and a total of 70.652 bug requests in the tracking system, gathered along the past 10 years. We employed two techniques - the KPSS Test and linear regression analysis, to assess the relationship between LL and SPL assets. Finally, results showed that three laws were supported based on the data employed (continuous change, increasing complexity, and declining quality). The other law (continuing growth) was partly supported, depending on the SPL evaluated asset (common, variable or product-specific).

Keywords: Software Product Lines, Software Evolution, Lehman's Laws of Software Evolution, Empirical Study.

1 Introduction

Software evolution is a very important activity where the software must have the ability to adapt according to the environment or user needs, to keep its satisfactory performance, [1] given that if a system does not support changes, it will gradually lapse into uselessness [2].

Back in the 1970s, Meir Lehman started to formulate his laws of software evolution, after realizing the need for software systems to evolve. These laws, shown in Table 1, stressed that a system needed to evolve due to its requirement to *operate in or address a problem or activity in the real world*, what Lehman called *E-type Software*.

According to Barry et al. [3], these laws can be ordered into three broad categories: (i) laws about the evolution of software system characteristics; (ii) laws referring to organizational or economic constraints on software evolution; and (iii) meta-laws of software evolution. However, the laws were evaluated in the context of single systems.

Table 1. Lehman's Laws of Software Evolution [4]

Software Evolution Laws	Description
Evolution of Software System Characteristics (ESSC)	
(1974) Continuous change	E-type systems must be continually adapted else they become progressively less satisfactory.
(1980) Continuing growth	The functional content of an E-type system must be continually increased to maintain user satisfaction with the system over its lifetime.
(1974) Increasing complexity	As an E-type system evolves, its complexity increases unless work is done to maintain or reduce it.
(1996) Declining quality	Stakeholders will perceive an E-type system to have declining quality unless it is rigorously maintained and adapted to its changing operational environment.
Organizational/Economic Resource Constraints (OERC)	
(1980) Conservation of familiarity	During the active life of an evolving E-type system, the average content of successive releases is invariant.
(1980) Conservation of organizational stability	The average effective global activity rate in an evolving E-type system is invariant over a products lifetime.
Meta-Laws (ML)	
(1974) Self regulation	The evolution process of E-type systems is self regulating, with a distribution of product and process measures over time that is close to normal.
(1996) Feedback System	The evolution processes in E-type systems constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable baseline.

In contrast to single systems, a Software Product Line (SPL) represents a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market or mission. The products which compose an SPL are developed from a common set of core assets in a prescribed way [5], aiming to achieve benefits such as large scale reuse, reduced time to market, improved quality and minimized costs, large-scale productivity, maintain market presence, enable mass customization, and so on [5] [6].

In order to achieve the above mentioned benefits, an SPL's evolution needs special attention, since the sources of SPL changes can be targeted to the entire product line (affecting common assets), targeted to some products (affecting variable assets), or targeted to an individual product (affecting product-specific assets) [7] [8] [9].

In this study, our objective is to examine whether Lehman's Laws (LL) are reflected in the development of SPLs, where common, variable and product specific assets are built. The hypothesis we put forward is that there is a relationship between LL of Software Evolution and the software evolution in SPL environments. In this context, in order to understand whether there is a relationship between the LL of software evolution and the SPL evolution process, we carried out an empirical investigation in an industrial software product line project. As a preliminary study, we selected the first group of laws (Evolution of Software System Characteristics - ESSC, which includes the Continuous change, Continuing growth, Increasing complexity and Declining quality laws) to perform our evaluation. We evaluated each one of the laws from the ESSC group for common, variable and product-specific assets in the context of an industrial SPL. To the best of our knowledge, this is the first study stating that most of evaluated LL of software evolution can be applied in the context of SPL. Thus, results of this study can help in understanding and improving SPL evolution process.

The remainder of this paper is organized as follows: Section 2 presents related work and uses it as context to position this work. Section 3 describes the empirical investigation, followed by the discussion of key findings and contributions from our preliminary empirical study in Section 4. Finally, Section 5 presents the conclusions and future directions.

2 Related Work

Since the publication of Lehman's work on software changes, other researchers have investigated his laws within the context of open source and industrial projects. Israeli and Feitelson [10] examined Lehman's Laws (LL) within the context of the Linux kernel. They selected the Linux kernel because of its 14 years data recording history about the system's evolution, which includes 810 versions. Only two out of the eight laws were not supported in this experiment (i.e., *self-regulation* and *feedback system*). Barry et al. [3] also investigated LL; however within the context of industrial projects. They proposed some metrics as dependent variables, which were also related to six LL (the *self-regulation* and the *feedback system* laws were not investigated in this study). We have adapted some of the metrics proposed by Barry et al. [3] to support and evaluate the LL in an industrial SPL project.

Lotufo et al. [11] studied the evolution of the Linux kernel variability model. This model is responsible for describing features and configurations from the Linux kernel. They found that the feature model grows together with the code. Besides the growth of the number of features, the complexity still remains the same. Most of the evolution activity is related to adding new features and there is a necessity for tool support, mainly regarding constraints. Their results showed that evolving large variability models is feasible and does not necessarily deteriorate the quality of the model. Godfrey and Tu [12] found a similar conclusion after studying the evolution of the Linux kernel. They explored the evolution of the Linux kernel both at the system level and within the major subsystems and found out that the Linux has been growing in a super-linear rate over the years. However, as will be detailed later, within the context of our study we found a different behavior. The complexity within the assets has grown over the years and the quality has decreased. It is important to notice that the number of maintainers in a private context is smaller compared to maintainers of the Linux kernel and also the time-to-market pressure in a private context influences the overall software product quality.

Xie et al. [13] also investigated LL by studying 7 open source applications written in C and several laws were confirmed in their experiment. Their analysis covered 653 releases in total and sum 69 years of software evolution including the 7 applications. According to the authors, the definition of the *increasing complexity* and *declining quality* laws may lead to misinterpretations, and the laws could be supported or rejected, depending on the interpretation of the law definition. In our study, to avoid this misinterpretation, we consider that the increasing of complexity and the declining of quality must happen to support these laws.

The investigations assessing LL available in the literature are related to single systems and not to SPL. In our empirical study, we evaluated four laws of software evolution (ESSC group of laws) in an SPL industrial project, which can be considered as a first work in this direction. This project has a long history of data with more than 10 years of evolution records, as many of the related work. Nevertheless, it is a private system developed using the SPL paradigm, allowing the evaluation of the laws for the common, variable and product-specific assets.

3 Empirical Study

This empirical study focuses on investigating the relationship between LL (ESSC group of laws) of software evolution and the common, variable and product-specific assets, based on data from an industrial SPL project.

The industrial SPL project, used as basis for the investigation described herein, has been conducted in partnership with a company located in Brazil, which develops for more than 10 years strategic and operational solutions for hospitals, clinics, labs and private doctor offices. This company has $\sim$ 50 employees, of which six are SPL developers with a range of 4 to 19 years of experience in software development.

The company builds products within the scope of four main areas (hospitals, clinics, labs and private doctor offices). Such products comprise 45 modules altogether, targeting at specific functions (e.g., financial, inventory control, nutritional control, home care, nursing, and medical assistance). Market trends, technical constraints and competitiveness motivated the company to migrate their products from a single-system development to an SPL approach. Within SPL, the company was able to deliver its common, variable and product-specific assets. To keep the company name confidential, it will be called *Medical Company (MC)*. During the investigation, MC allowed full access to its code and bug tracking system.

Regarding the bug tracking system, we collected a total of 70.652 requests over 10 years, allowing an in-depth statistical data analysis. MC uses a bug tracking system called *Customer Interaction Center (CIC)*, which was internally developed. CIC allows MC's users to register requests for adaptations, enhancements, corrections and also requests for the creation of new modules.

The empirical study presented herein was planned and executed according to Jedlitschka et al.'s guidelines [14].

3.1 Planning

All the products at MC have some assets (called modules) in common (commonalities), some variable assets (variabilities) and also some specific assets (product-specific), enabling the creation of specific products depending on the combination of the selected assets.

Figure 1 shows the division of modules between the areas supported by MC. Four (4) modules represent the commonalities of the MC SPL, twenty-nine (29) modules represent the variabilities of the MC SPL and, twelve (12) modules represent the product-specific assets, totaling forty-five (45) modules in the MC SPL.

Based on those modules, some of the laws could be evaluated with the records from CIC. However, other ones required the LOC of these modules. From CIC and LOC, we collected data since 1997. Nevertheless, data related to the three types of maintenance (adaptive, corrective and perfective) just started to appear in 2003.

The GQM approach [15] was used to state the goal of this empirical study, as follows: the goal of this empirical study is *to analyze Lehman's Laws of Software Evolution (ESSC group of laws)* for the purpose of *evaluation* with respect to *its validity* from the point of view of *the researcher* in the context of *an SPL industrial project.* Based on the stated Goal, the following research questions were defined:

RQ1.) Is there a relationship between the *Continuous Change* law and the evolution of common, variable and product-specific assets?
RQ2.) Is there a relationship between the *Continuous Growth* law and the evolution of common, variable and product-specific assets?

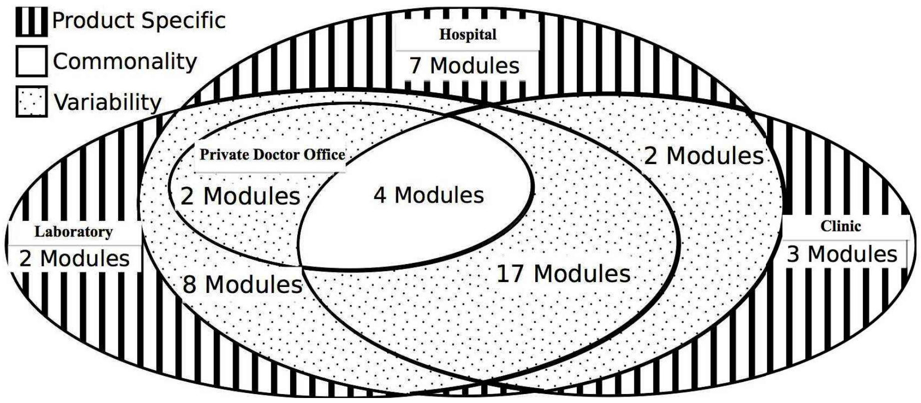


Fig. 1. Modules (assets) per Areas Supported by MC

RQ3.) Is there a relationship between the *Increasing Complexity* law and the evolution of common, variable and product-specific assets?
RQ4.) Is there a relationship between the *Declining Quality* law and the evolution of common, variable and product-specific assets?

The term relationship used in the RQs seeks for evidences of each evaluated law in the SPL common, variable, and product specific assets. In order to answer those questions, some metrics were defined. Since the SPL literature does not provide clear metrics directly associated to the laws, the metrics extracted from Barry et al. [3], Xie et a. [13], Kemerer and Slaughter [16], and Lehman et al. [17] were used herein. Barry et al. defined some dependent variables and some metrics for measuring each dependent variable. Based on their work, in order to evaluate each LL of software evolution, we have adapted the relationship among laws, dependent variables and the measurements (as shown in Table 2), according to the available data in the private industrial environment. Moreover, instead of using the Cyclomatic Complexity [18] as in Barry's work, we decided to use the LOC metric, since LOC and Cyclomatic Complexity are found to be strongly correlated [19].

For each one of the dependent variables, we have stated one null and one alternative hypothesis. The hypotheses for the empirical study are shown next:

H_0 : There is no growth trend in the data during the years (Stationary);
H_1 : There is a growth trend in the data during the years (Trend);

To corroborate Lehman's Laws of Software Evolution, *Continuous Change, Continuous Growth, Increasing Complexity* and *Declining Quality*, we must reject H_0. Thus, if there is a trend of growth in the data during the years, there is evidence to support these four laws.

The next subsection describes how the data were collected and grouped to allow the evaluation of the defined hypotheses.

Table 2. Relationship among Laws, Dependent Variables and Measurement

Law	**Dependent Variable**	**Acronym**	**Measurement**
Continuous change	Number of Activities	NA	Count of *corrective*, *adaptive* and *perfective* requests per year [3]
Continuous growth	Lines Of Code	LOC	Number of lines of code of modules per year [13]
Increasing complexity	Number of Corrections per LOC	NCLOC	Total of *correction* requests divided by LOC of modules per year (adapted from [16])
Declining quality	Number of Corrections per Module	NCM	Total *correction* requests divided by the number of modules per year [3]

3.2 Execution

The object of this study was the MC SPL. To collect the necessary data (from source code and the bug tracking system), we defined an approach composed of three steps. In the first step, we were able to collect data from *Customer Interaction Center (CIC)*, in the second one we collected LOC data and at the third step, MC clarified some doubts, through interviews, that we had about the collected data.

After collecting all the data, we started to group them according to an CIC filed. When registering a new request at CIC, the user must fill a field called *request type*. Based on this request type, the records from CIC were grouped according to the types of maintenance [20] [21]. The records were grouped in three types of maintenance, according to Table 3.

It was possible to relate each request from the bug tracking system to either adaptive, corrective or perfective maintenance since each request has a field for its type, and each type is related to a maintenance type. The preventive maintenance type was not used because none of the records corresponded to this type. We also show other records not related to maintenance types from CIC, since MC also use CIC to register management information. These other records had a null request type field or their request type field was related to commercial proposals, visiting requests or training requests. Thus, they were not used in the analysis. We found a total of 70.652 requests in the CIC system.

Based on this classification and the LOC, we were able to investigate each dependent variable and also perform the statistical analysis as discussed in the next section.

3.3 Data Analysis and Discussion

For analyzing the evolution at MC, in the first step, we collected data related to all assets and we did not distinguish common, variable and product-specific records. This step can be seen in each graph from Appendix A as the

Table 3. Maintenance Types Groups

Maintenance Type	**Request Type in CIC**	**Total of Records**
Adaptive	Reports and System Adaptation Request	22,005
Corrective	System Error, Database Error, Operating System Error, Error Message of type General Fail in the System, Error Message (Text in English) and System Locking / Freezing	14,980
Perfective	Comments, Suggestions and Slow System	2,366
Other	Doubts, Marketing - Shipping Material, Marketing Presentation, Marketing Proposal, Marketing Negotiation, Training and Visit Request	31,301

Total line. As our objective was to evaluate the evolution in software product lines, we grouped the records into commonalities, variabilities and product-specific, facilitating the understanding of the evolution at MC.

The period in which the data were collected was not the same to evaluated the laws. The continuous growth law was evaluated using data from the period between 1997 and 2011. The other laws (continuous change, increasing complexity, and declining quality) were evaluated using data from the period between 2003 and 2011.

In order to evaluate the hypotheses we applied the KPSS Test [22]. This test is used to test a null hypothesis for an observable time series. If the series is stationary, then we do not reject the null hypothesis. Otherwise, if the series has a trend, we reject the null hypothesis. In this study, the level of confidence used was 5% (p-value). We could evaluate all assets (common, variable and product-specific) for all the laws using this statistical test.

We applied also a linear regression analysis [23] to the collected data to evaluate the variance between the assets of the SPL at MC (see Appendix B). Through this variance, we could understand which assets evolve more and should receive more attention. We have checked the variance ($Y = \beta_0 + \beta_1 X$) for each dependent variable and for each asset (Common = comm, Variable = var, Product-Specific = ps) of the SPL.

The descriptive statistics analysis and the discussion of the empirical study results are shown next grouped by each research question.

RQ1.) Is there a relationship between the *Continuous Change* law and the evolution of common, variable and product-specific assets?

For this law, the number of activities (adaptive, corrective and perfective) registered in CIC from January 2003 up to December 2011 were used, corresponding to the *Number of Activities* dependent variable as shown on Appendix A(a). The plot shows a growth for the commonalities and variabilities, however, for the product-specific activities a small decrease can be noticed for the last years. The number of activities related to the variabilities are greater than the activities related to the commonalities. For the product-specific activities, as expected, there is a smaller number of activities, since it corresponds to the small group of assets from the MC SPL.

Besides the small decrease for product-specific assets activities for the last years, we could identify a trend of growing in the number of activities for all assets (common, variable and product-specific) by applying the KPSS Test. Based on the confidence intervals analysis, Appendix B(a) indicates that the different assets from the SPL have different amounts of activities. The number of activities related to the variabilities are bigger in the SPL because *"variability has to undergo continual and timely change, or a product family will risk losing the ability to effectively exploit the similarities of its members"* [24].

RQ2.) Is there a relationship between the *Continuous Growth* law and the evolution of common, variable and product-specific assets?

Regarding this law, it was possible to identify similar behaviors for each of the SPL assets. Appendix A(b) shows the lines of code from 1997 up to 2011, corresponding to the *Lines Of Code* dependent variable. For common, variable and product-specific assets, we can observe a tendency to stabilization over the years. They grow at a high level in the first years, but they tend to stabilize over the next years.

Due to the growth in the number of activities for common and variable assets according to the *Continuous Change* law, these activities had an impact on the LOCs. By using the KPSS Test, the commonalities and variabilities showed a trend of increasing. Despite the continuous change observed for the commonalities and variabilities in the SPL, MC does not worry about keeping the size of its common and variable assets stable, contributing with the increase of complexity.

For the product-specific assets, the KPSS Test showed a stationary behavior. Therefore, the *Continuos Growth* law to the product-specific assets is rejected. This happens because similar functions among the product-specific assets are moved to the core asset of the SPL [25].

Moreover, through the confidence intervals analysis, Appendix B(b) shows that the variabilities from the SPL have more LOC than other assets. This could be one of the reasons why variabilities have more activities (Continuous Change).

RQ3.) Is there a relationship between the *Increasing Complexity* law and the evolution of common, variable and product-specific assets?

The total number of corrections per line of code, corresponding to the *Number of Corrections per LOC* dependent variable is shown in Appendix A(c). As it can be seen, the complexity for commonalities, variabilities and product-specific assets is increasing up to 2007. This increase was bigger for the commonalities because at that time MC had to evolve the SPL to support new government laws. However, variable and product-specific assets have also grown up to 2007, since modifications within common assets also had an impact on variable and product-specific assets [7] [8] [9]. After 2007, MC started to try to reduce the complexity and prevent the system from breaking down.

However, we could also identify a trend of growing in the complexity for the commonalities, variabilities and product-specific assets by applying the KPSS Test in the *Increasing Complexity* law. Hence, considering that the complexity is always growing, the *Increasing Complexity* law is supported for all the assets in the SPL at MC.

Confidence intervals analysis (see Appendix B(c)) indicates that the complexity inside the commonalities raises more than inside other assets. It happens because the commonalities have to support all the common assets from the products of the SPL and any change can affect the entire product line [7] [8] [9] [26].

RQ4.) Is there a relationship between the *Declining Quality* law and the evolution of common, variable and product-specific assets?

The number of corrections per total of modules in the year, corresponding to the *Number of Corrections per Module* dependent variable, is shown in Appendix A(d). The number of corrections for the variabilities and for the specific assets follow almost the same pattern. However, for the common assets of the product line, we can notice a higher number of corrections per module in 2007, also caused by the adaptation of the system to the government laws. A small increase in the variabilities and in the specific assets also can be observed in the same year. From 2007, the number of corrections per module starts to decrease because of the feedback from users and corrections of problems related to the evolution to deal with the new government laws.

Besides the decrease after 2007, a trend of growing in the number of corrections per modules could be identified for the commonalities, variabilities and product-specific assets by using the KPSS Test. Hence, considering that the number of corrections per module is always growing, the *Declining Quality* law is supported for all the assets in the SPL at MC.

Based on the confidence intervals analysis, Appendix B(d), we could conclude that the number of corrections per module inside the commonalities is bigger than the other assets. As stated for commonalities the increasing complexity law, this happens because the commonalities have to support all the common assets from the products of the SPL and any change can affect the entire product line [7] [8] [9] [26].

The results of the KPSS test are shown in Appendix C. The next section discusses the threats to validity of the empirical study.

3.4 Threats to Validity

There are some threats to the validity of our study. They are described and detailed as follows.

External Validity threats concern the generalization of our findings. In the analyzed medical domain, government laws are published constantly. This may affect the generalizability of the study since those laws affect both common, variable and product-specific assets. Hence, it is not possible to generalize the results to other domains, however, the results maybe considered to be valid in the medical domain.

This empirical study was performed in just one company and only one SPL was analysed. Thus, it needs a broader evaluation in order to try to generalize the results. However, this was the first study in this direction evaluating each of the four laws of software evolution (ESSC group of laws) in an SPL industrial company with more than 10 years of historical data, which is not always available for researchers.

Internal Validity threats concern factors that can influence our observations. The period in which the data were collected was not the same. We evaluated some laws using the period between 1997 and 2011. Others were evaluated using the period between 2003 and 2011. Also, the requests from the bug tracking system were used in the same way no matter of their quality, duplication or rejected requests. Nevertheless, the available data period is meaningful because it is an industrial SPL project and both periods can be considered long periods, where statistical methods could be successfully applied.

Construct Validity threats concern the relationship between theory and observation. The metrics used in this study may not be the best ones to evaluate some of the laws, considering that there is no baseline for those metrics applied to SPL. However, metrics used to evaluate Lehman's laws of software evolution in previous studies were the basis for our work. For some metrics, we based on LOC. Even though LOC can be considered a simplistic method, LOC and Cyclomatic Complexity are found to be strongly correlated [19], thus we decided to use LOC since MC had this information previously available.

4 Key Findings and Contributions for SPL Community

In this section, we present the key findings and also discuss what is the impact of each finding for industrial SPL practitioners, according to each law.

a. *Continuous Change Law. Finding*: Variable assets are responsible for the greater number of activities performed in the SPL project. Practitioners should be aware of making modifications within those assets, since there are several constraints among them. Also we could realize that the number of product-specific activities decreases starting 2007 while the number of activities on common and variable assets increases. It could be that there are so many activities on the variable and common assets (compared to the product-specific assets) because their scope has not been chosen well (or has changed significantly in 2007), implying that more and more specifics assets have to be integrated into commonalities. This would be a typical product line behavior. Also, another reason for increasing the number of activities on common and variable assets is that in SPL more attention is by definition given to the commonalities and variabilities for the sake of reuse.
b. *Continuous Growth Law. Finding*:Variable assets had also the biggest growth in LOCs during the years. Practitioners should search, among the variable assets, those that share behavior and can be transformed into common assets. Transforming variable assets into common assets will reduce the total growth

of variable assets and also it will reduce their complexity. However, within this study, the transition from variability to commonality does not happen at all, however, it should happen. For example, when a variability is been used in almost all the products of the SPL.

c. *Increasing Complexity Law. Finding*: Complexity within common assets is bigger that for other assets. Practitioners should be aware of complexity in common assets since they have to support all the products from the SPL [26]. This makes any kind of change in common assets to be considered as critical, since they may affect the whole SPL.
d. *Declining Quality Law. Findings*: The number of corrections per modules were higher for common assets. In fact, for this empirical study we also have to consider the number of maintainers at MC, which was a small number and maybe they were overload with work and left behind quality issues. In a further study, we will check if there is any relationship between complexity and quality, because in this study, higher complexity is related with poor quality.

Based on the results of this empirical study, we propose the following initial items to improve the evolution within SPLs:

. *Creation of guidelines for evolving each SPL artifact.* Guidelines supporting evolution steps for SPL artifact should exist to systematize the evolution of common, variable and product-specific assets. These guidelines should consider why, when, where and how the SPL assets evolve.
. *For each evolution task, keep constant or better the quality of the SPL.* Measurements within the SPL common, variable and product-specific assets (including requirements, architecture, code, and so on) should be part of the SPL evolution process.
. *For each evolution task, try to decrease the complexity of the SPL.* After evolving the SPL code, measurements should be applied to check if the new change in the code increases or not the complexity of the SPL;

These are some improvements that can be followed according to the findings of the empirical study at MC. Next Section presents our conclusions and future work.

5 Conclusions and Future Work

Lehman's Laws of Software Evolution were published in the seventies and still are evaluated in recent environments, such as the one used in this empirical study. From this empirical study, commonalities, variabilities and product-specific assets seems to behave differently regarding evolution. Three laws were completely supported (continuous change, increasing complexity and declining quality) in this empirical study. The other law (continuous growth) was partly supported, depending on the SPL asset in question.

According to this study, all assets from the SPL industrial project are changing over the time. However, there is an increasing of complexity and a decrease of quality during the years. Changes in all assets will always happen, therefore, dealing with the complexity and quality in evolving an SPL needs special attention. To deal with the declining quality and increase complexity during the SPL evolution, we intend to propose guidelines. These guidelines will help during the whole SPL evolution starting from the SPL requirements up to the SPL tests.

This empirical study within an industrial SPL project was important to reveal findings not confirmed within open source projects where the complexity keeps the same, the quality is not deteriorated [11] and system grows in a super-linear rate [12].

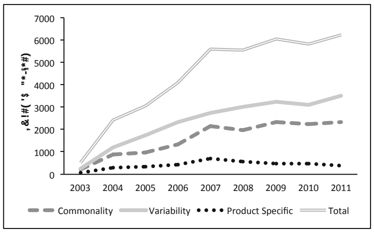
As future work, we would like to confirm if those laws (Continuous Change, Increasing Complexity, Declining Quality) also happen for other industrial SPL projects. Moreover, we would like to have more insights about the reason why continuous growth law is not supported to product-specific assets. In our opinion, we believe that the LL are also applicable in the SPL context, since most of the laws could be confirmed for most of the SPL assets (eleven at the total).

In addition, we would like to confirm some findings from this empirical study, such as: to confirm if the number of activities and LOC for variable assets are bigger than for other assets; investigate if the commonalities have a higher number of corrections related to other assets and; check if there is any relationship between complexity and quality, because in this study, higher complexity is related with poor quality. Hence, we intend to replicate this study in another company using SPL. For replicating this empirical study, any kind of bug tracking system can be used, since the replicated empirical study can use the same research questions, metrics, hypotheses and statistical methods. After synthesizing the results from both empirical studies, we will try to elaborate some insights of how Lehman's laws of software evolution occur in the SPL environment and propose some guidelines to the SPL evolution process. Based on these guidelines, it will be possible to evolve each SPL asset (common, variable or product-specific) according to its level of complexity, growth and desired quality.

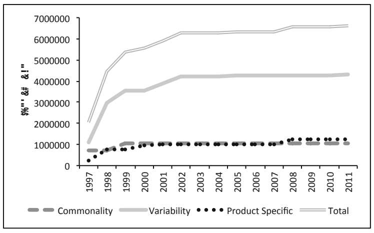
Acknowledgement. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[1]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08 and CNPq grants 305968/2010-6, 559997/2010-8, 474766/2010-1 and FAPESB. The authors also appreciate the value-adding work of all reviewers of this paper and some colleagues: Silvia Abrahão, Emilio Insfrán, and Emilia Mendes who made great contributions to the current work.

[1] INES - `http://www.ines.org.br`

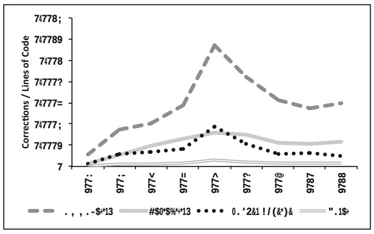
Appendix A. Plotted Graphs from CIC data and LOC



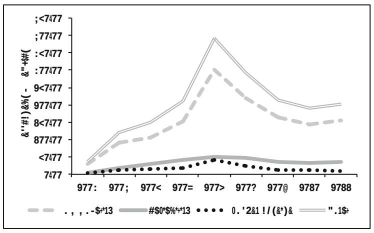
(a) Number of Activities



(b) Lines of Code per Year

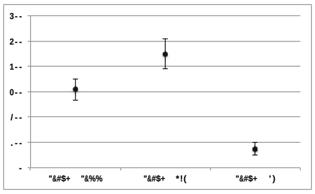


(c) Corrections / LOC per Year

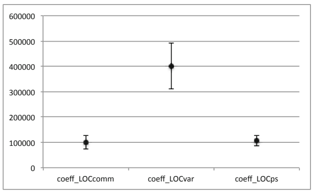


(d) Number of Corrections per Module

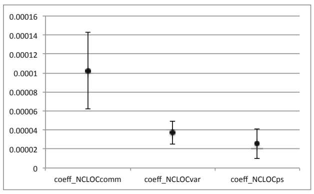
Appendix B. Confidence Intervals (Regression Coefficients)



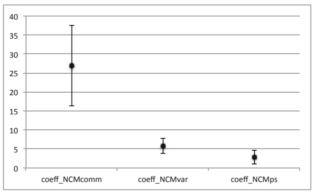
(a) NA



(b) LOC



(c) NCLOC



(d) NCM

Appendix C. KPSS Test and Hypotheses Results

Variable	Commonalities			Variabilities			Product-Specific		
	KPSS Test	p-value	Decision	KPSS Test	p-value	Decision	KPSS Test	p-value	Decision
NA	0.1651	0.0341	Reject H_0	0.2187	0.0100	Reject H_0	0.1979	0.0168	Reject H_0
LOC	0.2125	0.0113	Reject H_0	0.2400	0.0100	Reject H_0	0.1354	0.0697	Do Not Reject H_0
NCLOC	0.1856	0.0214	Reject H_0	0.2252	0.0100	Reject H_0	0.1764	0.0249	Reject H_0
NCM	0.1856	0.0214	Reject H_0	0.2274	0.0100	Reject H_0	0.1799	0.0236	Reject H_0

H_0: Stationary; H_1: Trend. The gray shading represents the supported laws/assets for this empirical study.

References

1. Mens, T., Demeyer, S.: Software Evolution. Springer (2008)
2. Lehman, M.: Programs, life cycles, and laws of software evolution. Journal Proceedings of the IEEE 68, 1060–1076 (1980)
3. Barry, E.J., Kemerer, C.F., Slaughter, S.A.: How software process automation affects software evolution: A longitudinal empirical analysis: Research articles. Journal of Software Maintenance and Evolution: Research and Practice 19, 1–31 (2007)
4. Cook, S., Harrison, R., Lehman, M.M., Wernick, P.: Evolution in software systems: foundations of the SPE classification scheme. Journal of Software Maintenance 18, 1–35 (2006)
5. Clements, P.C., Northrop, L.: Software Product Lines: Practices and Patterns. In: SEI Series in Software Engineering. Addison-Wesley (2001)
6. Schmid, K.: A comprehensive product line scoping approach and its validation. In: Proceedings of the 24th International Conference on Software Engineering (ICSE), pp. 593–603. ACM, New York (2002)
7. Ajila, S., Kaba, A.: Using traceability mechanisms to support software product line evolution. In: Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI), pp. 157–162. IEEE (2004)
8. Bailetti, A., Ajila, S., Dumitrescu, R.: Experience report on the effect of market reposition on product line evolution. In: Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI), pp. 151–156. IEEE (2004)
9. Svahnberg, M., Bosch, J.: Evolution in software product lines: Two cases. Journal of Software Maintenance and Evolution: Research and Practice 11, 391–422 (1999)
10. Israeli, A., Feitelson, D.G.: The linux kernel as a case study in software evolution. Journal of Systems and Software 83, 485–501 (2010)
11. Lotufo, R., She, S., Berger, T., Czarnecki, K., Wąsowski, A.: Evolution of the linux kernel variability model. In: Bosch, J., Lee, J. (eds.) SPLC 2010. LNCS, vol. 6287, pp. 136–150. Springer, Heidelberg (2010)
12. Godfrey, M.W., Tu, Q.: Evolution in open source software: A case study. In: IEEE International Conference on Software Maintenance (ICSM), pp. 131–142. IEEE Computer Society, Washington (2000)
13. Xie, G., Chen, J., Neamtiu, I.: Towards a better understanding of software evolution: An empirical study on open source software. In: IEEE International Conference on Software Maintenance (ICSM), pp. 51–60. IEEE (2009)

14. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting experiments in software engineering. In: Shull, F., Singer, J., Sjboerg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering, pp. 201–228. Springer, London (2008)
15. Marciniak, J.J.: Encyclopedia of Software Engineering. In: Basili, V.R., Caldiera, G., Rombach, H.D. (eds.) Goal Question Metric Approach 2, pp. 528–532. Wiley-Interscience, Hoboken (1994)
16. Kemerer, C., Slaughter, S.: An empirical approach to studying software evolution. Journal IEEE Transactions on Software Engineering (TSE) 25, 493–509 (1999)
17. Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., Turski, W.M.: Metrics and laws of software evolution - the nineties view. In: Proceedings of the 4th International Symposium on Software Metrics, pp. 20–32. IEEE Computer Society, Washington (1997)
18. McCabe, T.J.: A complexity measure. IEEE Transactions on Software Engineering (TSE) 2, 308–320 (1976)
19. Kan, S.H.: Metrics and Models in Software Quality Engineering. Addison-Wesley, Boston (2002)
20. Gupta, A., Cruzes, D., Shull, F., Conradi, R., Rønneberg, H., Landre, E.: An examination of change profiles in reusable and non-reusable software systems. Journal of Software Maintenance and Evolution: Research and Practice 22, 359–380 (2010)
21. Lientz, B.P., Swanson, B.E.: Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations. Addison-Wesley (1980)
22. Kwiatkowski, D., Phillips, P.C.B., Schmidt, P., Shin, Y.: Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? Journal of Econometrics 54, 159–178 (1992)
23. Yan, X., Su, X.G.: Linear Regression Analysis: Theory and Computing. World Scientific Publishing, River Edge (2009)
24. Deelstra, S., Sinnema, M., Nijhuis, J., Bosch, J.: Cosvam: a technique for assessing software variability in software product families. In: 20th IEEE International Conference on Software Maintenance (ICSM), pp. 458–462. IEEE (2004)
25. Mende, T., Beckwermert, F., Koschke, R., Meier, G.: Supporting the grow-and-prune model in software product lines evolution using clone detection. In: 12th European Conference on Software Maintenance and Reengineering (CSMR), pp. 163–172. IEEE (2008)
26. McGregor, J.D.: The evolution of product line assets. Technical Report, Software Engineering Institute, CMU/SEI-2003-TR-005 (2003)