Evaluating Lehman's Laws of Software Evolution within Software Product Lines Industrial Projects

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

[1]*Federal Institute of Sergipe, Campus Estância*
Rua Café Filho, 260, 49.200-000, Estância - Sergipe, Brazil
[2]*Federal University of Bahia, Campus Ondina*
Av. Adhemar de Barros, s/n, 40.170-110, Salvador - Bahia, Brazil

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 system and the results shown that these single systems evolve according to his stated laws over time. However, considering 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 all of the stated laws are still valid for SPL since they were partially evaluated in this context. Thus, this paper details an empirical investigation where Lehman's Laws (LL) of Software Evolution were used in two SPL industrial projects to understand how the SPL assets evolve over time. These projects are related to an application in the medical domain and another in the financial domain, developed by medium-size companies in Brazil. They contain a total of 71 modules and a total of 71.442 bug requests in their tracking system, gathered along the total of more than 10 years. We employed two techniques - the KPSS Test and linear regression analysis, to assess the relationship between LL and SPL assets. Results showed that one law was completely supported (*conservation of organizational stability*) for all assets within both empirical studies. Two laws were partially supported for both studies depending on the asset type (*continuous growth* and *conservation of familiarity*). Finally, the remaining laws had differences among their results for all assets (*continuous change*, *increasing complexity*, and *declining quality*).

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, (Mens and Demeyer, 2008). If a system does not support changes, it will gradually lapse into uselessness (Lehman, 1980).

Thus, 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 (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. (2007), 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. Moreover, the laws were fully evaluated in the context of single systems (Lehman et al., 1997, 1998; Xie et al., 2009; Israeli and Feitelson, 2010) and partially evaluated within the context of Software Product Lines (SPL) in a previous study of ours (Oliveira et al., 2015).

SPL represents a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market or mission. SPL products are developed from a common set of core assets in a prescribed way (Clements and Northrop, 2001) 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 (Clements and Northrop, 2001; Schmid, 2002).

In order to achieve the above mentioned benefits, the SPL 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) (Svahnberg and Bosch, 1999; Ajila and Kaba, 2004; Bailetti et al., 2004). Moreover, when compared to single systems, the evolution of SPL introduces more challenges due to the high level of complexity to evolve the variability and interdependency between products (Botterweck and Pleuss, 2014). Thus, evolving an SPL can be risky (Neves et al., 2015), since any change can impact common, variable, and product-

Email address: raphael.oliveira@ifs.edu.br, {alcemirsantos, esa}@dcc.ufba.br, gecynalda@yahoo.com (Raphael Pereira de Oliveira[1], Alcemir Rodrigues Santos[2], Eduardo Santana de Almeida[2], Gecynalda Soares da Silva Gomes[2])

Table 1: Lehman's Laws of Software Evolution (Cook et al., 2006).

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 product's 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.

specific assets.

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 hypotheses we put forward are that relationships between LL of Software Evolution and the software evolution in SPL environments exist. Therefore, we carried out empirical investigations within two SPL industrial projects in order to understand whether there is a relationship between the LL of software evolution and the SPL evolution process.

In our preliminary study (Oliveira et al., 2015), we evaluated the first category of laws (*Evolution of Software System Characteristics - ESSC*, which includes the *Continuous Change, Continuing Growth, Increasing Complexity, and Declining Quality laws*) within an industrial SPL project running in the medical domain. We extended this work in two ways: *(i)* we performed an evaluation of the second category of laws (*Organizational/Economic Resource Constraints - OERC*, which includes the *Conservation of Familiarity and the Conservation of Organizational Stability laws*) for the same industrial SPL project in the medical domain; and *(ii)* we performed another evaluation including both categories of laws (ESSC and OERC) for an industrial SPL project in the financial domain. So far, the *Meta-Laws - ML* (*Self Regulation* and *Feedback System* laws) were not evaluated in both industrial projects. To the best of our knowledge, this is the first study stating that most of evaluated LL of software evolution applies to SPLs, according to the two performed evaluations. Thus, results of this study can help both, understanding and improving the 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 both empirical investigations, including a comparison between their results and a discussion on the descriptive statistics results. Section 4 presents the threats to validity of our studies. Next, we discuss the key findings and contributions to the SPL community in Section 5. Finally, Section 6 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.

The first empirical work on software changes was carried out by Lehman (1980), in which he used a large-scale commercial system, the IBM OS/360, to validate his laws. From this work, two well know Laws of Software Evolution could be summarized as follows (Gupta et al., 2010): Law of continuing change (a system that is being used undergoes continuing change) and Law of increasing complexity (a computer program that is changed becomes less and less structured). After the publication of Lehman's eight laws of software evolution (Lehman et al., 1997), other researchers started to check their validity within open source and industrial software.

Godfrey and Tu (2000) 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 can influences the overall software product quality.

Barry et al. (2007) also investigated Lehman's Laws (LL); however within the context of industrial projects. They proposed some metrics as dependent variables (*number of activities; module count; cyclomatics per module, operands per module, and calls per module; number of corrections per module; percentage growth in module count; number of activities per developer*), which were also related to six LL (the *self-regulation* and the *feedback system* laws were not investigated in this study). In their study, four laws were supported (continuous change, continuous growth, declining quality, and conservation of familiarity laws) and two were not supported (increasing complexity and conservation of organizational stability). We have adapted some of the metrics (as shown in the next section) proposed by Barry et al. (2007) to support and evaluate the LL in an industrial SPL project.

Xie et al. (2009) 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.

Israeli and Feitelson (2010) examined 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, and also because within the Linux kernel they had access to all the source code. Thus, they were able to evaluate all of the laws within the Linux kernel and realized that the Linux kernel is an example of a *perpetual development*, where the development is performed in collaboration with its users. Hence, most of the laws were supported in their evaluation and only two out of the eight laws were not supported (i.e., *self-regulation* and *feedback system*).

Lotufo et al. (2010) 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 add new features. Their results showed that evolving large variability models is feasible and does not necessarily deteriorate the quality of the model.

Gonzalez-Barahona et al. (2014) studied the evolution of a long lived FLOSS[1] software project, called glib, based on information of around 20 years in the Source Code Management (SCM) repository. They observed which information could be extracted from the SCM repository and then evaluated the laws according to the available retrieved information (i.e.: number of commits, number of files per commit, number of changes, lines added and removed, Lines of Code (LOC), Source Lines of Code (SLOC)). They used these retrieved information to evaluate most of the laws, however, others (self-regulation, declining quality, and feedback system) could not be evaluated with the available data. The findings for this study were: the continuing change law was completely supported; the increasing complexity law was supported only in part and during some periods; the conservation of organizational stability law was supported at least to a certain extent; the conservation of familiarity law was not supported during most of the life of glibc and; the continuing growth law was not supported during the last phase of the project.

Godfrey and German (2014) performed some observations on modern software evolution and LL. They discuss how LL may need to evolve to accommodate new trends (i.e.: agile development, cloud-based services, powerful run-time environments). According to them, new metrics and new empirical models are necessary to deal with new approaches of software development and reuse. Thus, LL are *"beholden"* to Lehman's first law (continuous change), in which we need to continually adapt them or they will become less useful (Godfrey and German, 2014).

In our previous work (Oliveira et al., 2015), we evaluated the ESSC group of laws within an industrial SPL project in the medical domain. The 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*). This was a preliminary study evaluating four LL of software evolution within SPL. We extended this work in two ways: 1) we performed an evaluation of the OERC group of laws for the industrial SPL project in the medical domain and; 2) we performed another evaluation of both ESSC and OERC group of laws for another industrial SPL project in the financial domain. The ML group of laws (*Self Regulation* and the *Feedback System* laws) were not evaluated in both empirical studies. Barry et al. (2007) and Gonzalez-Barahona et al. (2014) also did not evaluated them. According to Barry et al. (2007), it is difficult to state which empirical model could be used to support or reject these laws. Gonzalez-Barahona et al. (2014) said that these ML require "feedback mechanisms" to be evaluated, which are key to their formulation. Thus, according to their available data, they could not evaluate them. We faced the same challenge within our empirical studies. We could not find an empirical model neither available feedback data within the companies to evaluate the ML.

[1] Free Open Source Software - http://freeopensourcesoftware.org

3. Empirical Studies

The empirical studies presented herein focuses on investigating the relationship between LL of software evolution and the common, variable, and product-specific assets, based on data from two industrial SPL projects. They were planned and executed according to Jedlitschka et al. (2008) and Carver (2010) guidelines.

We considered three dimensions for this study, namely *common*, *variable*, and *product-specific* assets represent the variability degree from both SPL. Common assets are present in all products from the SPL, variable assets are present in two or more products, but not in all of them, and product-specific assets are present only in a specific SPL product.

The first empirical study was performed in the medical domain and its replication was performed in the financial domain, both in an industrial setting.

Since most of the laws were supported for the first empirical study, we started to understand how SPLs evolve and we would like to have more insights to propose improvements in the SPL evolution process.

The replication within the financial domain was conducted by the same researchers (internal replication), using the same research questions, dependent variables, metrics and hypotheses, however, as a replicated study, the domain was changed to draw further conclusions.

Since both empirical studies used the same planning, we present first the general planning and after we show details of each empirical study.

3.1. General Planning

We started the planning for both empirical studies by defining their goals based on the *Goal Question Metric (GQM)* approach (Basili et al., 1994), as follows: the goal of each empirical study is *to analyze Lehman's Laws of Software Evolution* for the purpose of *evaluation* with respect to *its validity* from the point of view of *the researcher* in the context of *an industrial SPL project*. We proceed than with the addressed research questions:

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?

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?

RQ5. Is there a relationship between the *Conservation of familiarity* law, and the evolution of common, variable and product-specific assets?

RQ6. Is there a relationship between the *Conservation of organizational stability* law and the evolution of common, variable, and product-specific assets?

The term relationship used in the RQs seeks for evidence 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 Lehman et al. (1997), Kemerer and Slaughter (1999), Barry et al. (2007), and Xie et al. (2009) 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 environments. Moreover, since both companies only provided the total Lines of Code (LOC) of their modules, instead of using the Cyclomatic Complexity (McCabe, 1976) as in Barry's work, we decided to use the LOC metric, since LOC and Cyclomatic Complexity are found to be strongly correlated (Kan, 2002).

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. However, for the *Conservation of Familiarity* and the *Conservation of Organizational Stability* laws, we must not reject H_0. Thus, no growth trend should exist in the data during the years for these two laws.

In order to evaluate the hypotheses we applied the KPSS Test (Kwiatkowski et al., 1992). 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 an upward trend, we reject the null hypothesis. In this study, we use a significance level of 5%. We could evaluate the assets (common, variable, and product-specific) for all the laws using this statistical test.

We applied also a linear regression analysis (Yan and Su, 2009) to the collected data to evaluate the variance between the assets of the industrial SPL projects.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.

Figure 1 shows the workflow of the study, including step by step from research questions definition to the hypoteses testing and the linear regression analysis of the target SPL's assets. Next, we present details of the case studies.

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 (Barry et al., 2007)
Continuous growth	Lines Of Code	LOC	Number of lines of code of modules per year (Xie et al., 2009)
Increasing complexity	Number of Corrections per LOC	NCLOC	Total of *correction* requests divided by LOC of modules per year (adapted from Kemerer and Slaughter (1999))
Declining quality	Number of Corrections per Module	NCM	Total *correction* requests divided by the number of modules per year (Barry et al., 2007)
Conservation of familiarity	Relative Growth in Module count	RGM	New modules created in the year divided by the total number of modules from the previous year (Barry et al., 2007)
Conservation of organizational stability	Number of Activities per Developer	NAD	Count of *corrective*, *adaptive* and *perfective* requests divided by the count of developers in the year (Barry et al., 2007)

3.2. The Companies

The first industrial SPL project has been conducted in partnership with a medical 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 ~ 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 access to its total LOCs per module 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 company in the replicated study was an IT company in the financial domain. Thus, herein to preserve the company's name, it will be called *Financial Company (FC)*. FC is using the SPL paradigm with success to develop their common, variable, and product specific assets, and it has more than 3 years of historical data, which allowed the statistical data analysis.

FC builds products for four main areas: *financial support*, *account support*, *budget support*, and *fixed assets support*. During the investigation, FC allowed access to its total LOCs and its bug tracking system.

Regarding the bug tracking system, FC uses one developed by FC itself, called *Client Relationship Center (CRC)*. CRC allows FC's users to register requests for adaptations, enhancements, corrections, and also requests for the creation of new modules.

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

Figure 2 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.

Figure 3 shows the division of modules among the main areas in which FC develops products. One (1) module composes the commonalities of the FC SPL, seven (7) modules compose the variabilities of the FC SPL, and the others eighteen (18) modules compose the product specific assets, totalizing twenty-six (26) modules in the FC SPL.

Based on those modules, some of the laws could be evaluated with the records from CIC (MC). 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.

For the FC, most of the laws could be evaluated with the records from CRC, however, other ones required the LOC metric. From CRC, we collected data since 2010 concerning to the three types of maintenance (adaptive, corrective, and perfective). Regarding LOC, we were able to collect data since 2009.

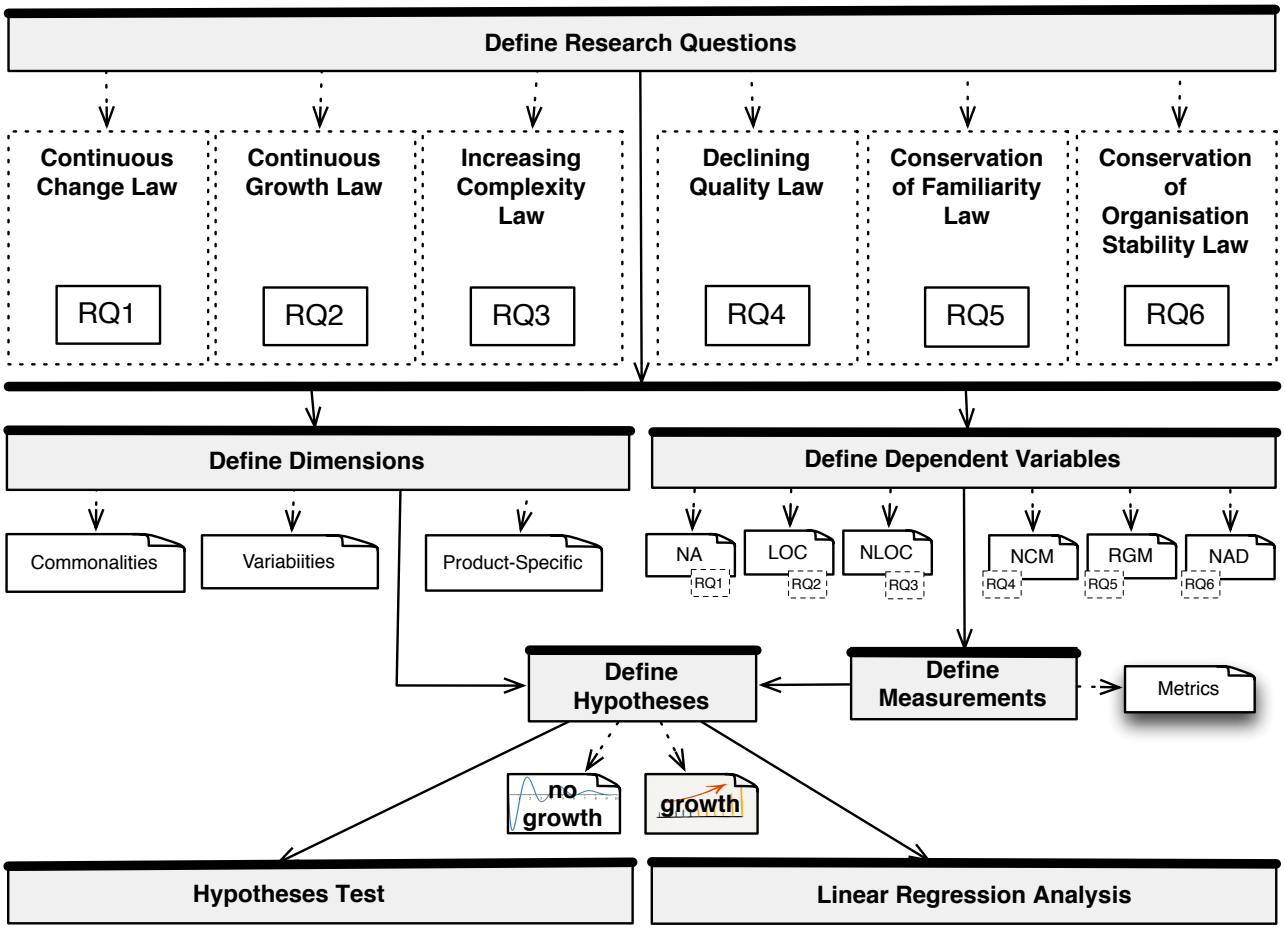

Figure 1: Planning overview.

3.3. Changes to the Original Experiment

The first empirical study was evaluated using a larger data set, which was grouped by year. Within this first empirical study, we were able to collect data since 1997 for LOCs and since 2003 for the bug tracking system. However, in the second study, the data set was smaller. Data from LOCs started to appear in January of 2009, and data from the bug tracking system started to appear in November of 2010. Thus, in order to check some tendencies using statistical methods, the second study was grouped by months.

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

3.4. Execution

The object of this first study was the MC SPL. To collect the necessary data (from LOC and the bug tracking system), we defined an approach composed of three steps. It is important to notice that during the data collection, MC only provided access to the total amount of LOC. Thus, MC did not provided access to the source code, since they are a private company. In the first step, we were able to collect data from CIC. These data correspond to all types of requests that the users can make within CIC. In a second step, we collected LOC data and in a 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 field. 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 (Lientz and Swanson, 1980; Gupta et al., 2010). 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 uses 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

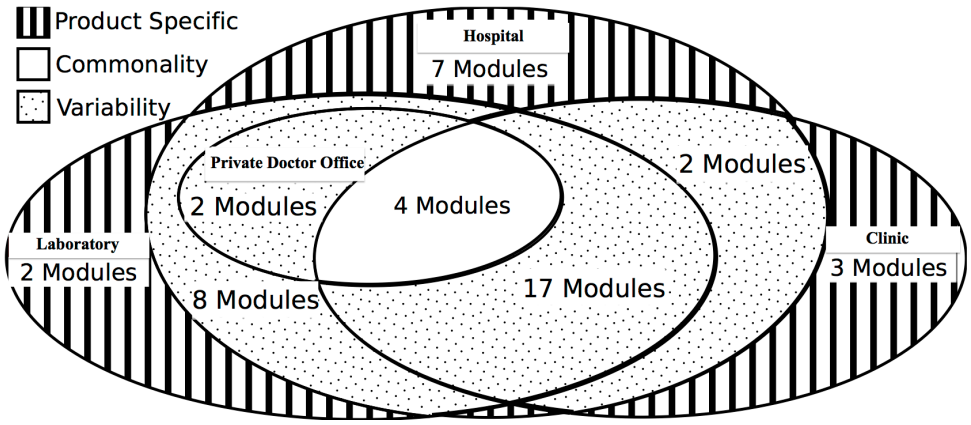

Figure 2: Modules (assets) per Areas Supported by MC.

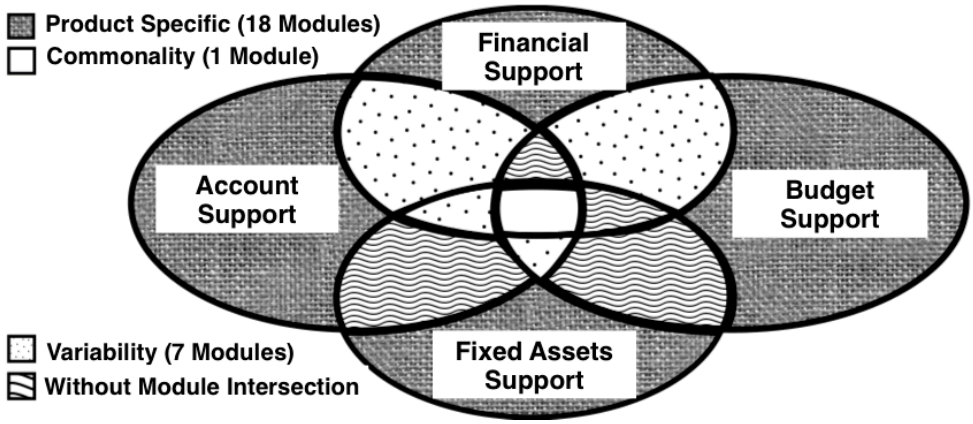

Figure 3: Modules (assets) per Area at FC.

Table 3: Maintenance Types Groups at MC.

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

the CIC system.

For the second empirical study we defined an approach composed of two steps to collect the necessary data (from LOC and the bug tracking system). Also, it is important to notice that during the data collection, FC only provided access to the total amount of LOC. In the first step, we collected data from CRC and LOC. These data correspond to all types of requests that the users can make within CRC and the total LOCs, respectively. In a second step, FC 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 CRC field. When registering a new request, the user must fill a field called *request type*. Based on this request type, the records from CRC were grouped according to the types of maintenance (adaptive, corrective, and perfective), as shown in Table 4.

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.

3.5. Data Analysis and Discussion

For analyzing the evolution at MC and FC, in the first step, we collected data related to all assets and we did not distin-

Table 4: Maintenance Types Groups at FC.

Maintenance Type	Total of Records
Adaptive	25
Corrective	724
Perfective	41
TOTAL	790

guish common, variable, and product-specific records to check the general evolution of MC and FC assets. This step can be seen in each graph from the next Figures as the *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 and FC.

The period in which the data were collected at MC was not the same to all the laws evaluated. The continuous growth and conservation of familiarity laws were evaluated using data from the period between 1997 and 2011. The other laws (continuous change, increasing complexity, declining quality, and conservation of organizational stability) were evaluated using data from the period between 2003 and 2011. Also, within the FC, the period where the data were collected was not the same. The continuous growth and conservation of familiarity laws were evaluated using data from the period between January of 2009 and April of 2012. The other laws (continuous change, increasing complexity, declining quality, and conservation of organizational stability) were evaluated using data from the period between November of 2010 and June of 2012.

The descriptive statistics analysis and the discussion of the empirical studies 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 (MC) from January 2003 up to December 2011 were used, corresponding to the *Number of Activities* dependent variable as shown on Figure 4. 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, Figure 5 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

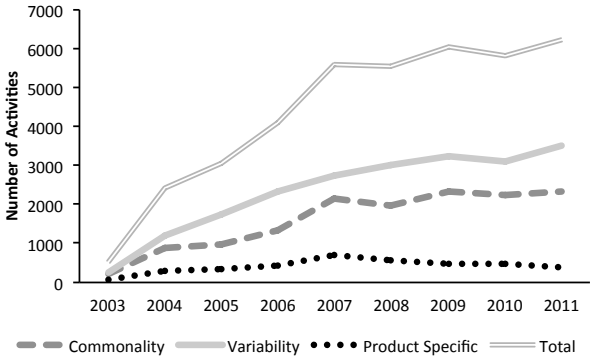

Figure 4: Number of Activities at MC SPL.

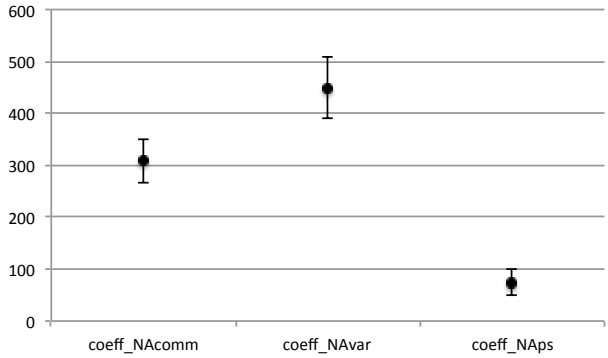

Figure 5: Confidence Intervals for the Regression Coefficients (NA) at MC SPL.

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"* (Deelstra et al., 2004).

For the FC, we also used the number of activities (adaptive, corrective, and perfective), however, we used only the activities registered in CRC from November 2010 up to June 2012, corresponding to the *Number of Activities* dependent variable as shown in Figure 6. The plot shows that all assets (common, variable, and product specific) grow up in the first months, however, after that, they present a ripple effect and they do not grow at all over the next months. By applying the KPSS statistical test, we could identify that commonalities, variabilities, and product specific assets have a stationary behavior over the months. Thus, we could not support the *continuous change* laws for all the assets at FC.

Based on the confidence intervals analysis, Figure 7, we could not identify which asset had more activities since they have intersections among their limits.

Most MC and FC SPL activities involve variable assets to allow the configuration, reuse, and customization of several products. However, since the companies are from distinct domains, we could realize that this law was completely supported for the medical domain, in which there are several requests for change coming from the Brazilian government healthy agency. On the other hand, the law was not supported within the financial do-

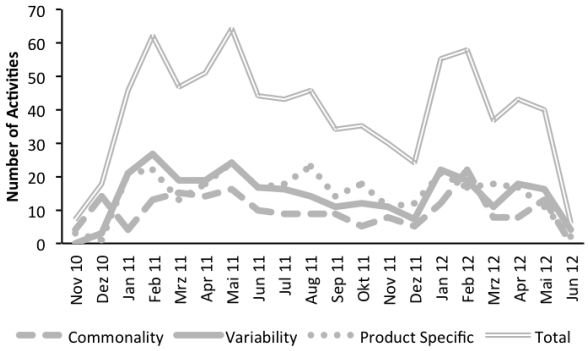

Figure 6: Number of Activities at FC SPL.

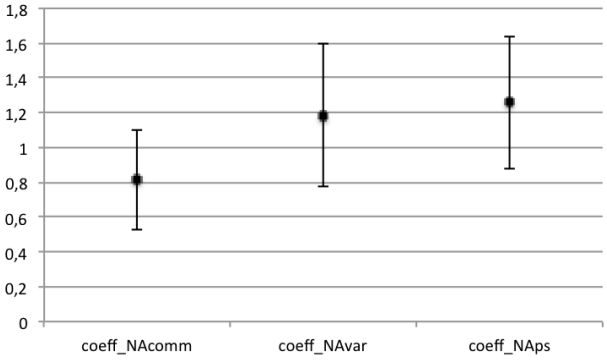

Figure 7: Confidence Intervals for the Regression Coefficients (NA) at FC SPL.

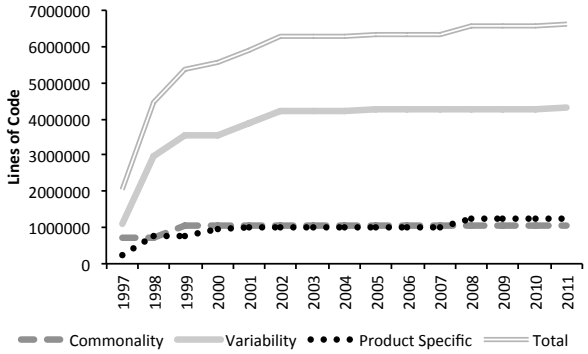

Figure 8: Lines of Code per Year at MC SPL.

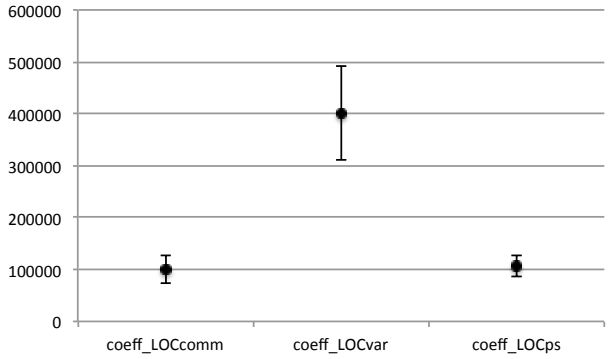

Figure 9: Confidence Intervals for the Regression Coefficients (LOC) at MC SPL.

main, which is considered a stable one. Thus, the SPL domain may affects on how its assets evolve over time.

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

Regarding this law at MC SPL, it was possible to identify similar behaviors for each of the SPL assets. Figure 8 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 at MC 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 at MC, 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 (Mende et al., 2008).

Moreover, through the confidence intervals analysis, Figure 9 shows that the variabilities from the MC SPL have more LOC than other assets. This could be a reason why variabilities have more activities (Continuous Change).

Figure 10 shows the lines of code from January of 2009 up to April 2012 at the FC SPL, corresponding to the *Lines Of Code* dependent variable. For common, variable, and product-specific assets, we can observe a small tendency of growing over the months. Besides, there is no such growing in the activities according to the *continuous change* law, overall, the total LOC of each asset had a growth trend.

By using the KPSS Test, the commonalities, variabilities, and product specific assets at FC showed a trend of increasing, thus, we could support the *continuous growth* law for all assets.

Moreover, through the confidence intervals analysis, Figure 11 shows that the product specific assets had more LOC than other assets at FC. One of the reasons is that product specific assets correspond to the majority of assets at FC SPL.

The increasing of LOC for variable assets at MC and product-specific assets at FC shows that there should be a control of these assets growth in order to avoid the SPL complexity growth. The law was not supported for product-specific at MC, however, it was supported for product-specific assets at FC. This occurred because within the financial domain these kind of asset is the majority, having more LOC than product-specific assets at MC. Thus, since most of the LOC at FC are

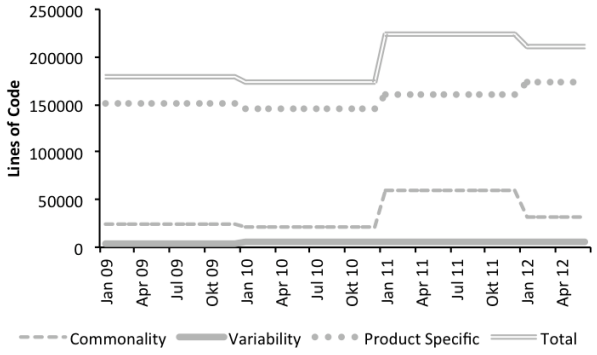

Figure 10: Lines of Code per Year at FC SPL.

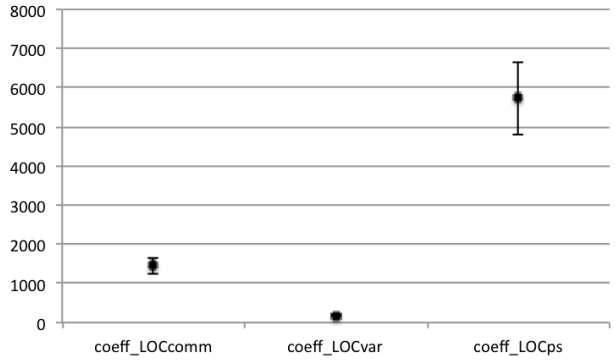

Figure 11: Confidence Intervals for the Regression Coefficients (LOC) at FC SPL.

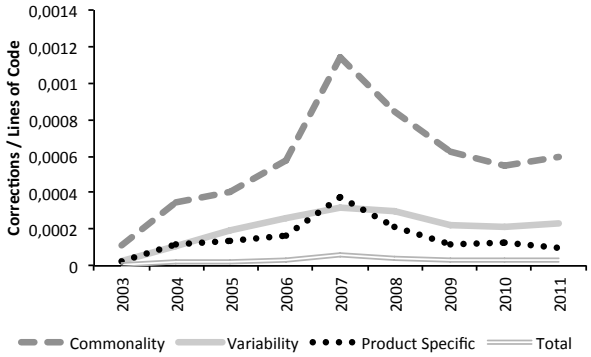

Figure 12: Corrections / LOC per Year at MC SPL.

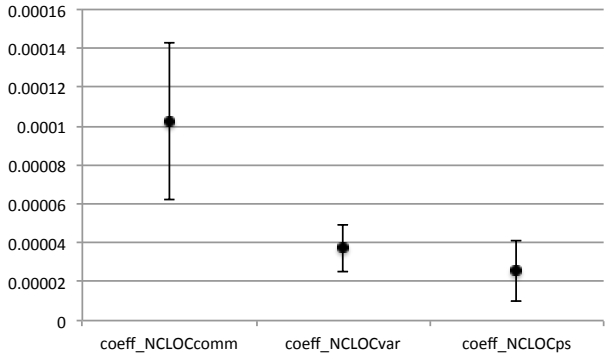

Figure 13: Confidence Intervals for the Regression Coefficients (NCLOC) at MC SPL.

product-specific assets, we could realize that MC deals better with the management of the variability within its SPL than FC.

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 Figure 12 for the MC SPL. 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 (Svahnberg and Bosch, 1999; Ajila and Kaba, 2004; Bailetti et al., 2004). 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 at MC SPL. 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 Figure 13) indicates that the complexity inside the commonalities raises more than inside other assets at MC. 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 SPL (Svahnberg and Bosch, 1999; McGregor, 2003; Ajila and Kaba, 2004; Bailetti et al., 2004).

For the FC SPL, the *Number of Corrections per LOC* dependent variable is shown in Figure 14. As it can be seen, the complexity for variabilities is higher compared to other assets. However, we could not 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 not growing over the months, the *Increasing Complexity* law is not supported for all the assets in the FC SPL.

Confidence intervals analysis (see Figure 15) indicates that the complexity inside the variabilities raises more than inside other assets at FC SPL. It happens 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"* (Deelstra et al., 2004). These continual and timely changes raise the complexity of variabilities at FC, however, the company deals with this complexity avoiding its growth.

The stabilization of the financial domain compared to the medical domain also influenced in the results of this law. Since there are more changes within the medical domain, the MC SPL

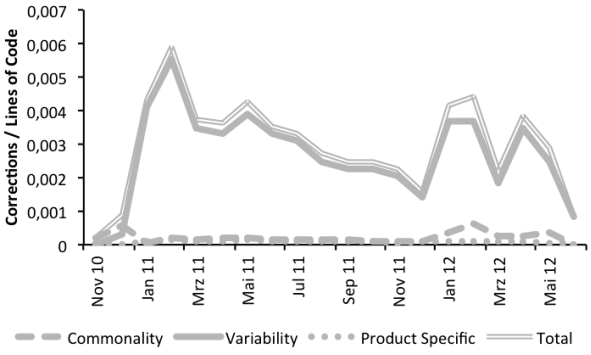

Figure 14: Corrections / LOC per Year at FC SPL.

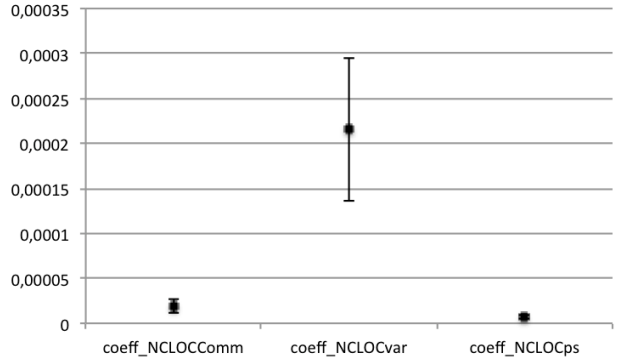

Figure 15: Confidence Intervals for the Regression Coefficients (NCLOC) at FC SPL.

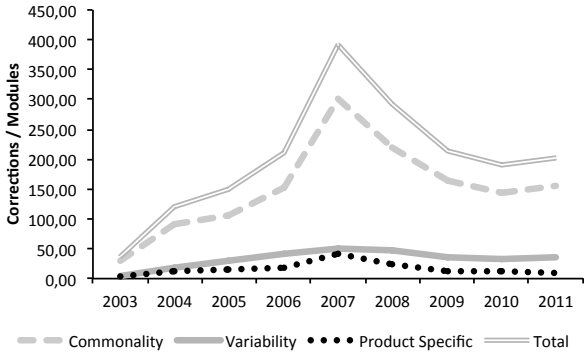

Figure 16: Number of Corrections per Module at MC SPL.

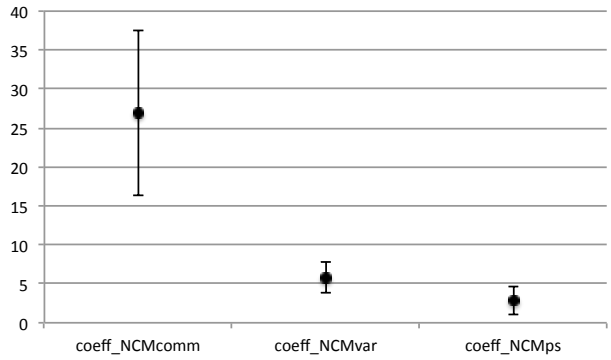

Figure 17: Confidence Intervals for the Regression Coefficients (NCM) at MC SPL.

complexity increased more. Nonetheless, the FC SPL complexity did not increased due to the domain stabilization. The complexity for common assets at MC and variable ones at FC are bigger because they are responsible for creating new SPL products.

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 for MC SPL, corresponding to the *Number of Corrections per Module* dependent variable, is shown in Figure 16. 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 at MC SPL. 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, Figure 17, we could conclude that the number of corrections per module inside the commonalities is bigger than the other assets for the MC SPL. As stated for commonalities in 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 (Svahnberg and Bosch, 1999; McGregor, 2003; Ajila and Kaba, 2004; Bailetti et al., 2004).

For the FC SPL, the *Number of Corrections per Module* dependent variable is shown in Figure 18. The number of corrections for the variabilities and for the specific assets follow almost the same pattern. However, common assets of the SPL had a higher number of corrections per module. Besides this growth for common assets, the commonalities, variabilities, and product-specific assets had a stationary behavior over the months by using the KPSS Test. Hence, considering that the number of corrections per module is not growing, the *Declining Quality* law is not supported for all the assets in the FC SPL.

Based on the confidence intervals analysis at FC SPL, Figure 19, we could conclude that the number of corrections per module inside the commonalities is bigger than the other assets. As stated, 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 (Svahnberg and

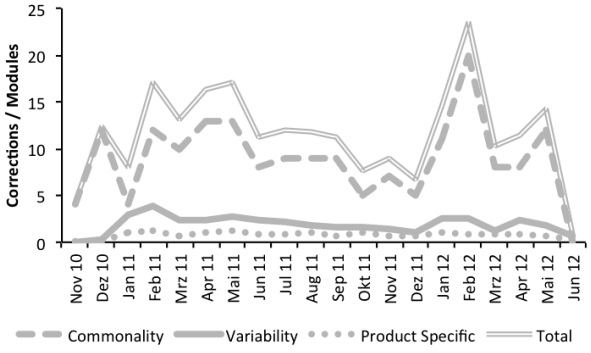

Figure 18: Number of Corrections per Module at FC SPL.

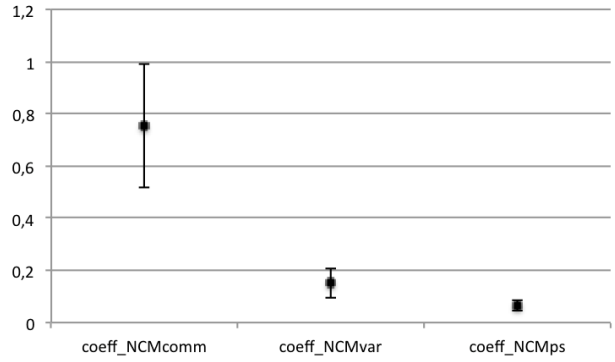

Figure 19: Confidence Intervals for the Regression Coefficients (NCM) at FC SPL.

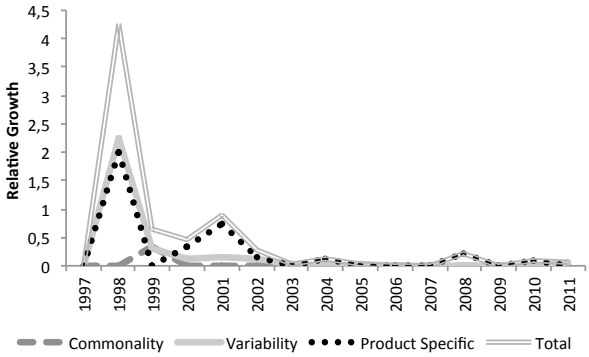

Figure 20: Relative Growth in Module Count at MC SPL.

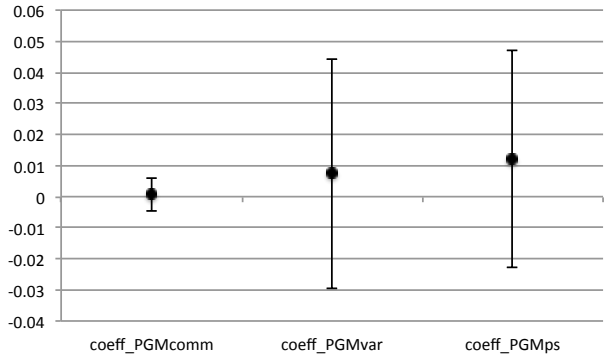

Figure 21: Confidence Intervals for the Regression Coefficients (RGM) at MC SPL.

Bosch, 1999; McGregor, 2003; Ajila and Kaba, 2004; Bailetti et al., 2004).

Once again, the stabilization of the financial domain compared to the medical domain influenced in the results of the declining quality law. Since there are more changes within the medical domain, the MC SPL quality is decreasing over time. However, the FC SPL quality did not decreased over time due to the domain stabilization. Moreover, the number of corrections per module was higher for common assets within the MC and FC SPLs. The decrease of quality within the common assets in both SPLs shows that both companies need to improve the quality of their common assets because they play a central role within the SPLs.

RQ5.) Is there a relationship between the *Conservation of Familiarity* law and the evolution of common, variable, and product-specific assets?

The relative growth in module count, corresponding to the *Relative Growth in Module count* dependent variable, is shown in Figure 20 for the MC SPL. We can notice that in 1997, MC had already a well defined scope for the common assets since almost all of them were included in the SPL at that time. Over the years, most of the assets included in the SPL, for accommodating the necessary changes, were variable and product-specific assets.

Regarding this law, a similar behavior could be identified in the MC SPL assets. To support the Conservation of Familiarity law, we were looking for assets that have a stationary growth over the years. The relative growth in module count for common, variable, and product-specific assets have a stationary behavior by applying the KPSS Test. Thus, the *Conservation of Familiarity* law was supported within this project.

Unfortunately, as shown in Figure 21, all assets at MC SPL have intersections among their limits, thus, we could not conclude which asset had the biggest relative growth.

The *Relative Growth in Module count* dependent variable is shown in Figure 22 for the FC SPL. We can notice a significant growth in the first months of 2010 for product specific assets. For common and variable assets we could notice an stationary behavior over the months.

The relative growth in module count for common and variable assets could not be evaluated because of their stabilization over the months at FC SPL. Product-specific assets have a stationary behavior by applying the KPSS Test. Thus, the *Conservation of Familiarity* law was supported only within the product specific assets.

Unfortunately, as shown in Figure 23, all the assets at FC SPL have intersections among their limits, thus, we could not conclude which asset had the biggest relative growth.

FC has a small amount of modules for common and variable assets, which resulted in their stabilization over time. However, we could realize that both MC and FC SPLs indeed care about

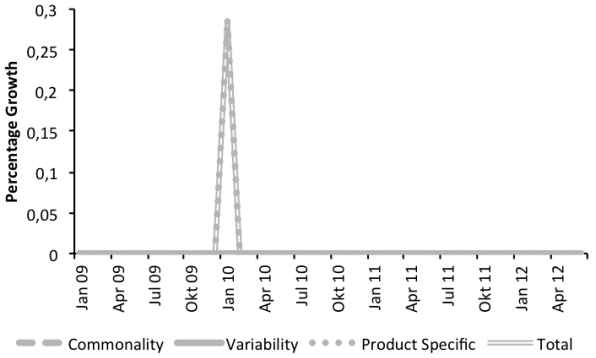

Figure 22: Relative Growth in Module Count at FC SPL.

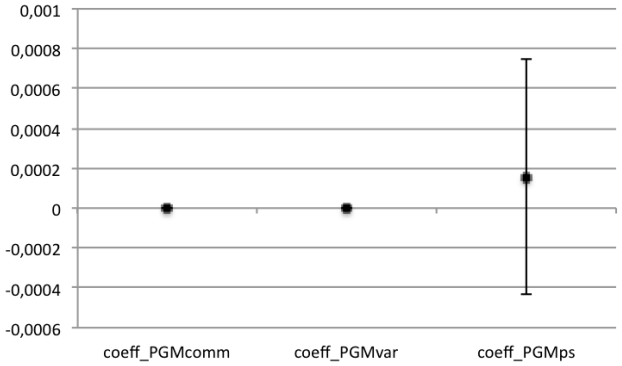

Figure 23: Confidence Intervals for the Regression Coefficients (RGM) at FC SPL.

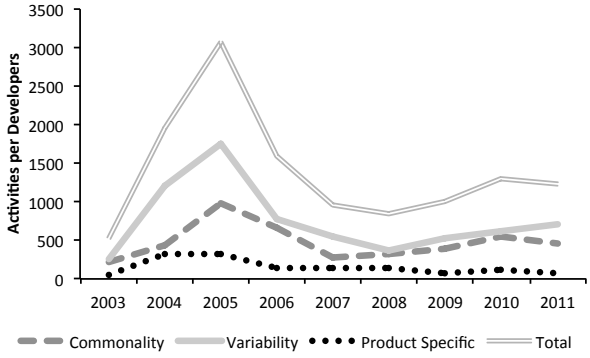

Figure 24: Number of Activities per Developer at MC SPL.

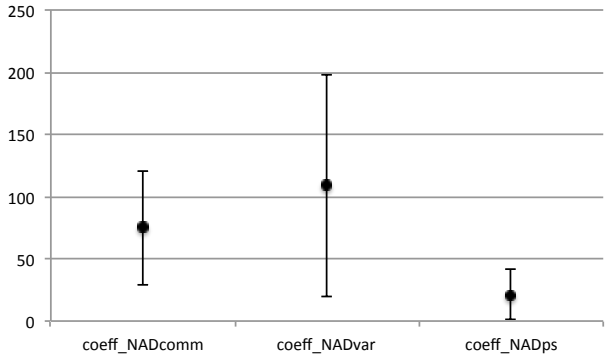

Figure 25: Confidence Intervals for the Regression Coefficients (NAD) at MC SPL.

the conservation of the familiarity law for product-specific assets. Thus, besides the small increase of product-specific assets, both SPLs handle in a constant manner the evolution of their modules. Moreover, the medical domain shows that it is also possible to handle in a smooth way the evolution of common and variable modules. Over time, this behavior tend to easy the SPL evolution because it may help in setting the goal of how many modules should the company create to keep its presence in the market over the year.

RQ6.) Is there a relationship between the *Conservation of Organizational Stability* law and the evolution of common, variable, and product-specific assets?

The total number of activities (adaptive, corrective, and perfective) per year divided by the total number of developers working on the activities, corresponding to dependent variable *Number of Activities per Developer*, is shown in Figure 24 for the MC SPL. The number of activities for the variabilities are bigger in the first years, however, it tends to decrease in the following years. This situation also happened within the common and product-specific assets of the product line.

To support the Conservation of Organizational Stability law, we were looking for assets that have a stationary growth over the years. Applying the KPSS Test in common, variable, and product-specific assets at MC SPL, the *Conservation of Organizational Stability* law was supported since they have a stationary trend, meaning that the conservation of organizational stability indeed exists.

Nevertheless, as shown in Figure 25, we could not conclude which assets at MC SPL had the biggest number of activities per developer, since the assets have intersections among their limits.

The dependent variable *Number of Activities per Developer* is shown in Figure 26 for the FC SPL. The number of activities for the variabilities are bigger in the first months, however, it tends to decrease in the following months. This situation also happened within the common and product-specific assets of the product line.

Applying the KPSS Test in common, variable, and product-specific assets at FC SPL, the *Conservation of Organizational Stability* law was supported since they have a stationary trend, meaning that the conservation of organizational stability indeed exists.

Nevertheless, as shown in Figure 27, we could not conclude which assets had the biggest number of activities per developer at FC SPL, since the assets have intersections among their limits.

All MC and FC assets supported this law. This happens because both companies ensure that their developers maintain the amount of work to be performed within the SPLs evolution. This result shows that regardless the domain, it is possible to maintain the right amount of work for each developer working

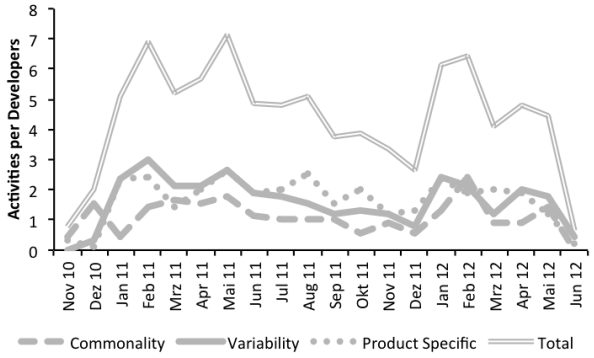

Figure 26: Number of Activities per Developer at FC SPL.

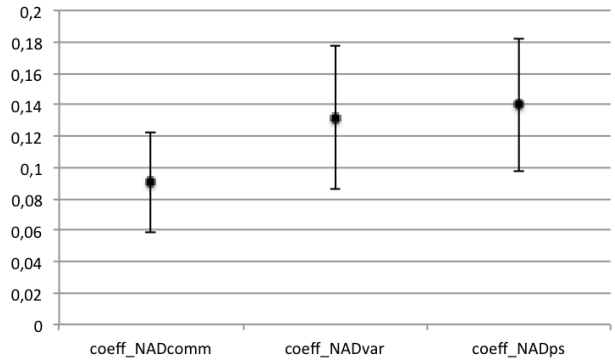

Figure 27: Confidence Intervals for the Regression Coefficients (NAD) at FC SPL.

in each kind of SPL asset.

The results of the KPSS test for MC and FC are shown in Table 5 and Table 6, respectively. Moreover, Figure 28 summarizes the results from both the linear regression analysis of the target SPL's assets and the hypothesis testing. We present next a comparison of the results from both empirical studies.

3.6. Comparison and Discussion of Results

After applying the KPSS statistical test in both empirical studies, we found that the results are one-third consistent. Next, we discuss the consistent and different results, according to Table 7.

3.6.1. Consistent Results

Comparing both empirical studies, one law (*conservation of organizational stability*) was completely supported for all assets (commonalities, variabilities, and product specific), as shown in Table 7.

- The *Conservation of Organizational Stability (NAD)* law. For this law, our findings shown that common, variable, and product specific assets have a stable behavior over the years for both empirical studies. Thus, the number of activities per developer over the time for both companies are constant, supporting this law.

Moreover, since the number of activities in all assets per developers keeps constant, these companies avoid a frequent problem of overloading the developers with extra tasks, smoothing the SPL evolution.

3.6.2. Partially Consistent Results

We also had partially consistent results for some assets concerning two laws (*continuous growth* and *conservation of familiarity*).

- The *Continuous Growth (LOC)* law. We could support the *Continuous Growth (LOC)* law in both empirical studies for common and variable assets, meaning that these assets keep growing in LOC over the years. Moreover, according to the first study at MC, variable assets had more LOC than other assets to support all the product configurations (Figure 9).

The increase of LOC for common and variable is an evidence that a control under the growth of these assets should exists to avoid a future increase within the SPL complexity.

- The *Conservation of Familiarity (RGM)* law. Product specific assets for this law also had a consistent result. Both results showed a stationary trend in the relative growth of module count over the time. Thus, this law was supported for product specific assets within both empirical studies.

3.6.3. Partially Different Results

We also had partially different results for some assets concerning the *continuous growth* and *conservation of familiarity* laws.

- The *Continuous Growth (LOC)* law. Product specific assets had a difference in the results for both empirical studies according to this law. With the MC empirical study, this law was not supported. However, it was supported for product specific assets at the second empirical study in the FC. Thus, the number of LOC's for product specific assets within FC increased over the time being also responsible for the higher number of LOC's within this SPL compared to other assets (Figure 11). This can also brings some increase in their complexity in a near future. Thus, in order to avoid such increasing, a standard behavior within an SPL should be a refactoring of product specific assets into common or variable ones.

- The *Conservation of Familiarity (RGM)* law. According to this law, the system growth should be constant. This could be confirmed for common and variable assets of the first empirical study. Hence, there is no relative growth in module count over the time. However, common and variable assets of the second study at FC could not be evaluated through the KPSS statistical test since the data were constant over the time. Thus, we need more studies to support this law for SPL common and variable assets.

3.6.4. Differences in Results

Three of the laws (*continuous change*, *increasing complexity*, and *declining quality*) had differences within all the assets (common, variable, and product specific) according to the results of both empirical studies, as shown in Table 7. Next, we discuss the differences.

Table 5: KPSS Test and Hypotheses Results for MC.

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
RGM	0.0508	0.1000	Do Not Reject H_0	0.0725	0.1000	Do Not Reject H_0	0.0491	0.1000	Do Not Reject H_0
NAD	0.0927	0.1000	Do Not Reject H_0	0.0911	0.1000	Do Not Reject H_0	0.0783	0.1000	Do Not Reject H_0

H_0: Stationary; H_1: Trend. The gray shading represents the supported laws/assets for this empirical study. RGM and NAD should be stationary to support Lehman's Laws of software evolution.

Table 6: KPSS Test and Hypotheses Results for FC.

Variable	Commonalities			Variabilities			Product-Specific		
	KPSS Test	p-value	Decision	KPSS Test	p-value	Decision	KPSS Test	p-value	Decision
NA	0.0776	0.1000	Do Not Reject H_0	0.0916	0.1000	Do Not Reject H_0	0.1578	0.1000	Do Not Reject H_0
LOC	1.0470	0.0100	Reject H_0	0.5130	0.0387	Reject H_0	1.3532	0.0100	Reject H_0
NCLOC	0.2041	0.1000	Do Not Reject H_0	0.0854	0.1000	Do Not Reject H_0	0.1402	0.1000	Do Not Reject H_0
NCM	0.1041	0.1000	Do Not Reject H_0	0.0851	0.1000	Do Not Reject H_0	0.1547	0.1000	Do Not Reject H_0
RGM	-	-	-	-	-	-	0.1277	0.1000	Do Not Reject H_0
NAD	0.0776	0.1000	Do Not Reject H_0	0.0916	0.1000	Do Not Reject H_0	0.1578	0.1000	Do Not Reject H_0

H_0: Stationary; H_1: Trend. The gray shading represents the supported laws/assets for this empirical study. RGM and NAD should be stationary to support Lehman's Laws of software evolution.

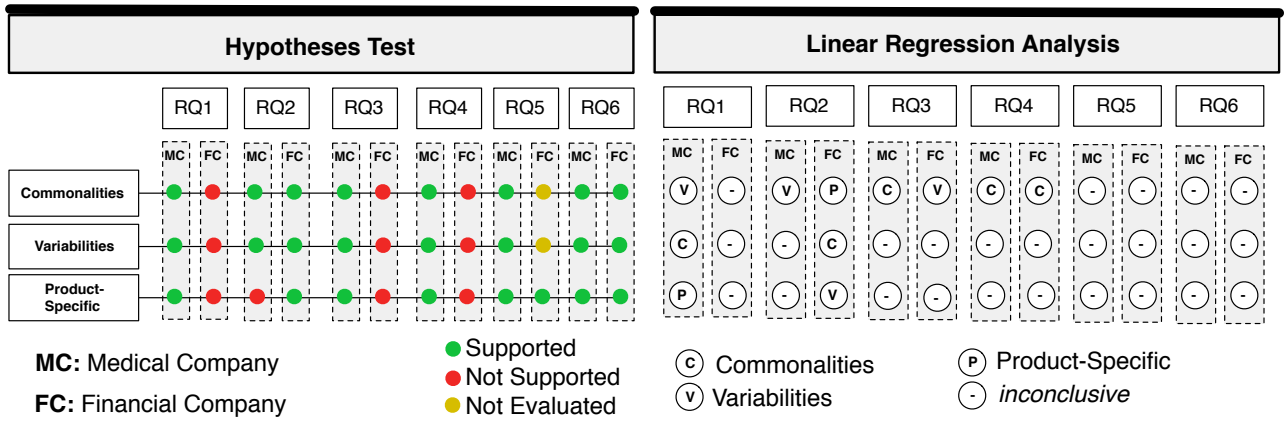

Figure 28: Summary of the case studies results.

Table 7: Consistent/Different Results from the Empirical Studies.

Dependent Variable	Law	Commonalities	Variabilities	Product Specific
NA	Continuous Change	Different	Different	Different
LOC	Continuous Growth	Consistent	Consistent	Different
NCLOC	Increasing Complexity	Different	Different	Different
NCM	Declining Quality	Different	Different	Different
RGM	Conservation of Familiarity	Different	Different	Consistent
NAD	Conservation of Organizational Stability	Consistent	Consistent	Consistent

- The *Continuous Change (NA)* law. This law was completely supported for all assets in the first empirical study at MC. Hence, the number of activities for MC is increasing over the time. It is interesting to check that the number of activities for the medical domain are increasing, thus the need to evolve the SPL according to the user needs or due a changing environment indeed happens. On the other side, neither asset supported this law within the second empirical study at FC. In the financial domain, the need to evolve the SPL according to the user needs does not happen because of the maturity in the domain, where less changes are needed according to user needs.

For the first study, we could identify that variable assets have the greater number of activities (Figure 5). The number of activities for variable and product specific assets in the second study seems to be apparently higher than activities for common assets, however, we could not verify which asset had more activities, since they had intersections among their limits, as shown in Figure 7. Nevertheless, we should consider the number of activities for variable assets, since it was the biggest number within the first study and it was also high for the second one. This happens to keep updated the variabilities in the SPL, allowing the configuration, and customization of several products.

- The *Increasing Complexity (NCLOC)* law. This law was completely supported for all assets in the first empirical study at MC. Hence, the number of corrections per LOC in MC is increasing over the time. On the other side, neither asset supported this law within the second empirical study at FC.

Moreover, the complexity was higher for common assets within the first empirical study (Figure 13), and it was higher for variable assets within the second empirical study (Figure 15). The complexity for these assets is higher because of the following SPL behaviors: common assets should support all the products from the SPL (McGregor, 2003) and; *"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"* (Deelstra et al., 2004).

- The *Declining Quality (NCM)*. This law was completely supported for all assets in the first empirical study at MC. Thus, the number of corrections per module count within MC is increasing over the time. Nonetheless, neither asset supported this law within the second empirical study at FC.

For both empirical studies, the evidences shown that common assets had more declining quality than others (Figure 17 and Figure 19). Since commonalities have to support all the common assets from the SPL products, their quality also should rise to improve the whole SPL quality.

3.7. Descriptive Statistics Results

To complement our statistics analysis, we also performed a descriptive statistics analysis of the results according to each dependent variable, grouped by the asset type (common, variable, and product-specific).

Commonalities for MC (Table 8) presented homogeneity only for LOC dependent variable (CV=11.7%), representing a low dispersion among the data (CV < 30.0%). Results also shown negative skew for NA and LOC dependent variables. At FC, the coefficient of variation for commonalities was constant to the RGM dependent variable (Table 9). Moreover, all dependent variables had a positive skew.

Variabilities for MC (Table 10) also shown homogeneity only for LOC dependent variable (CV=22.3%), representing a low dispersion among the data (CV < 30.0%). Results also shown that only RGM and NAD dependent variables had a positive skew. At FC, the coefficient of variation for variabilities was constant to the RGM dependent variable (Table 11). Moreover, the LOC dependent variable presented a low dispersion (CV=20.4%) and it was the only variable with a positive skew.

Product-specific for MC (Table 12) presented a low dispersion among the LOC data (CV=26.2%). Results also shown that only NA and LOC dependent variables had a negative skew. At FC, the coefficient of variation for the RGM dependent variable presented a high dispertion among the data (CV=640.3%), see Table 13. LOC presented a low dispertion among the data (CV=5.9%). Only LOC and RGM dependent variables presented a positive skew.

We could notice that, for all dependent variables, the descriptive statistics results (Minimum, 1st Quartile, Median, Mean, 3rd Quartile, Maximum, and Standard Deviation) have differences between MC and FC. Comparing the sizes, the results for FC were smaller than the MC ones. The mean of this difference was 99% for NA, 95% for LOC, 55% for NCLOC, 93% for NCM, 100% for PGM, 100% for NAD. Next, we present the threats to validity of our studies.

4. Threats to Validity

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

External Validity threats concern the generalization of our findings. We analyzed SPL from two specifics domains: the medical and the financial ones. There are requests for changing that are specific to these domains such as the need of changing because of a new government law (medical domain). Hence, it is not possible to generalize the results to other domains than the ones investigated here. However, the results maybe considered to be valid in the medical and financial domains.

These empirical studies involved two SPL companies. Thus, it needs a broader evaluation in order to try to generalize the results. However, this was the first step where LL were evaluated within two industrial SPL from different domains in which a long historical data was made available for the 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 1997-2011 (MC) / 2009-2012 (FC). Others were evaluated using the period between 2003-2011 (MC) / 2010-2012 (FC). Also, the requests from the bug tracking system were used in the same way no matter of their quality, duplication or rejected requests. Moreover, the size in modules of the systems were not the same. Nevertheless, the available data are meaningful because we could deal with industrial SPL projects with long periods of historical data, where statistical methods could be successfully applied.

Table 8: Descriptive Statistics Results for MC - Commonalities.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	219	984	1957	1603	2241	2355	773	48.2%	-0.493
LOC	716179	1049827	1049827	1005341	1049827	1049827	117399	11.7%	-1.945
NCLOC	0.000	0.000	0.001	0.001	0.001	0.001	0.000	51.2%	0.359
NCM	30	105	152	151	164	301	77	51.2%	0.359
RGM	0.000	0.000	0.000	0.022	0.000	0.333	0.086	387.3%	3.133
NAD	219	326	439	480	560	984	236	49.0%	0.879

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

Table 9: Descriptive Statistics Results for FC - Commonalities.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	4	8	9	10.42	13.5	22	5	44.8%	0.544
LOC	21429	21429	24472	34691	59537	59537	16451	47.4%	0.784
NCLOC	0.000	0.000	0.000	0.000	0.000	0.001	0.000	67.1%	1.338
NCM	4	8	9	9	12	20	4	41.2%	0.737
RGM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	NaN	NaN
NAD	0	1	1	1	2	2	1	44.8%	0.544

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

Table 10: Descriptive Statistics Results for MC - Variability.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	257	1750	2750	2350	3117	3495	1.078	45.9%	-0.700
LOC	1084158	3710162	4231216	3834242	4251868	4302112	854609	22.3%	-2.221
NCLOC	0.000026	0.000192	0.000224	0.000209	0.000258	0.000319	0.000092	44.0%	-0.672
NCM	4	30	34	33	41	50	14	43.8%	-0.639
RGM	0.000	0.000	0.038	0.208	0.127	2.250	0.572	274.6%	3.013
NAD	257	537	623	754	778	1750	461	61.2%	1.007

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

Table 11: Descriptive Statistics Results for FC - Variability.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	0	11	16	15.11	19	27	7	45.8%	-0.453
LOC	3782	3782	4865	4999	6395	6395	1018	20.4%	0.254
NCLOC	0.0000	0.0022	0.0031	0.0028	0.0036	0.0056	0.0013	47.9%	-0.273
NCM	0	2	2	2	3	4	1	47.4%	-0.256
RGM	0.000	0.000	0.000	0.000	0.000	0.000	0.000	NaN	NaN
NAD	0	1	2	2	2	3	1	45.9%	-0.453

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

Table 12: Descriptive Statistics Results for MC - Product-Specific.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	52	331	432	419.6	497	694	181	43.1%	-0.502
LOC	237865	989077	1005822	988482	1130335	1256781	259226	26.2%	-1.398
NCLOC	0.0000	0.0001	0.0001	0.0002	0.0002	0.0004	0.0001	62.8%	1.029
NCM	3	13	14	17	19	42	11	63.3%	1.041
RGM	0.000	0.000	0.000	0.244	0.183	2.000	0.526	215.3%	2.484
NAD	52	83	139	156	144	331	100	63.9%	0.817

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

Table 13: Descriptive Statistics Results for FC - Product-Specific.

Variable	Measures								
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max.**	**SD**	**CV**	**SV**
NA	1	12.5	17	15.74	19.5	24	6	39.3%	-0.892
LOC	145559	145559	151291	154929	159896	174229	9180	5.9%	0.850
NCLOC	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	38.5%	-0.812
NCM	0	1	1	1	1	1	0	38.3%	-0.951
RGM	0.000	0.000	0.000	0.007	0.000	0.286	0.045	640.3%	5.942
NAD	0	1	2	2	2	3	1	39.3%	-0.892

SD: Standard Deviation; **CV:** Coefficient Of Variation; **SV:** Skewness Coefficient

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 LL 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 (Kan, 2002), thus we use LOC since MC and FC only provided access to this information.

Conclusion Validity threats concern with issues that affect the ability to draw the correct conclusion about the outcome of the study. Indeed, not all of the laws were supported to all assets (common, variable, and product-specific) concerning the evaluated laws. However, most of the laws were supported according to the asset type within both empirical studies (twenty-four supported assets from a total of third-six assets).

5. 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*. Variable assets are responsible for the greater number of activities performed in the MC industrial SPL project and also it is one of the assets with more activities for the FC SPL industrial project. Practitioners should be aware of making modifications within those assets, since there are several constraints among them. According to the MC SPL, 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 and variabilities. This would be a typical SPL behavior. Also, another reason for increasing the number of activities on variable assets is that SPL needs more attention within the variabilities for the sake of reuse.

b. *Continuous Growth Law*. The increase of LOC for common and variable is an evidence that a control under the growth of these assets should exists to avoid a future increase within the SPL complexity. Also, we identified that variable and product-specific assets had the biggest growth. Practitioners should search among the variable and product specific assets, those that share behavior and can be transformed into common assets. Transforming variable and product specific assets into common assets will reduce their total growth and it also will reduce their complexity.

c. *Increasing Complexity Law*. Complexity within common assets is bigger than for other assets within the MC SPL. According to the FC SPL, variable assets had the biggest complexity. Practitioners should be aware of complexity in common and variable assets since they have to support all the products and variations from the SPL (McGregor, 2003; Deelstra et al., 2004). This makes any kind of change in common and variable assets to be considered as critical, since they may affect the whole SPL.

d. *Declining Quality Law*. The number of corrections per modules were higher for common assets in both studies. In fact, for these empirical studies we also have to consider the number of maintainers at both companies, which was a small number. Moreover, it is very important to control the quality of the SPL, mainly for commonalities. Since they have to support all the common assets from the SPL, their quality also should rise to improve the whole SPL quality.

e. *Conservation of Familiarity Law*. All assets from MC (common, variable, and product-specific) supported the conservation of familiarity law. Within the FC SPL, we could not evaluate this law for common and variable assets because of the data stabilization over the time. Hence, more studies are needed. However, product specific assets at FC also supported this law. In order to avoid the loss of familiarity with the SPL, practitioners should maintain the relative growth for all assets constant, avoiding the exponential growth.

f. *Conservation of Organizational Stability Law*. All assets support this law by keeping the same amount of work among the developers for both empirical studies. Practitioners should try to achieve a better management of the developers' work, without overloading them with unnecessary work.

Based on the results of these empirical studies, 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 may be applied to check if the new change in the code increases or not the complexity of the SPL;

. *Improve the feedback.* The management team should be aware of the clients' needs in order to keep adding/removing functionalities within the SPL paying attention to not increase LOCs of the assets.

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

6. Conclusions and Future Work

Lehman's Laws of Software Evolution were published in the seventies and are still perceived in nowadays software evolution context. The investigation performed here with two industrial SPL projects shown that commonalities, variabilities, and product-specific assets seems to behave differently regarding evolution.

For instance, the law *conservation of organizational stability* was completely supported for all assets within both empirical studies. Two laws were partially supported for both studies: *continuous growth* - common and variable assets; and *conservation of familiarity* - product specific assets. Two laws had partially different results: *continuous growth* - rejected for MC product specific assets / supported for FC product specific assets; and *conservation of familiarity* - supported for MC common and variable assets / not evaluated for FC common and variable assets. And finally, the other laws/assets had differences among their results. Three laws had completely different results for all assets within both studies: *continuous change* - supported for all assets at MC / rejected for all assets at FC; *increasing complexity* - supported for all assets at MC / rejected for all assets at FC; and *declining quality* - supported for all assets at MC / rejected for all assets at FC.

Few results were consistent between both empirical studies. The benefits are twofold: the consistent results confirm once again the LL regarding SPL; and the different results indicate that more investigation is needed to better understand the reason why LL may not hold for SPL. However, we believe that the LL are also applicable in the SPL context, since most of the laws could be supported for most of the SPL assets of the two SPL industrial projects (twenty-four supported assets from a total of third-six assets).

In summary, the consistent results shown that SPL support the conservation of organizational stability law, however, there is a need to cope with the continuous growth, which may affect the whole SPL complexity and quality. According to the first study, all assets are changing over the time. However, there is an increasing of complexity and a decrease of quality over the years. Therefore, dealing with the complexity and quality in evolving an SPL needs special attention.

As future work, we would like to obtain more insights about the supporting of LL based on others SPL industrial projects. Moreover, we intend to explore more the similar/different behaviors of LL, according to the type of development: SPL and single system. Thus, we pretend to apply the same evaluation performed in this study within each single software (SPL product) of the SPLs. After that, the obtained results will be compared to identify the similar/different behaviors within single and SPL developments. Additionally, to deal with the declining

quality and increase complexity during the SPL evolution, we intend to propose guidelines. These guidelines may help during the whole SPL evolution starting from the SPL requirements up to the SPL tests, considering all LL of software evolution.

References

Ajila, S., Kaba, A., November 2004. 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.

Bailetti, A., Ajila, S., Dumitrescu, R., November 2004. Experience report on the effect of market reposition on product line evolution. In: IEEE Int. Conf. on Information Reuse and Integration (IRI). pp. 151 – 156.

Barry, E. J., Kemerer, C. F., Slaughter, S. A., January 2007. 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.

Basili, V. R., Caldiera, G., Rombach, H. D., 1994. Goal Question Metric Paradigm. Vol. 2. Wiley-Interscience New York, NY, USA, pp. 528–532.

Botterweck, G., Pleuss, A., 2014. Evolution of software product lines. In: Mens, T., Serebrenik, A., Cleve, A. (Eds.), Evolving Software Systems. Springer Berlin Heidelberg, pp. 265–295.

Carver, J., 2010. Towards reporting guidelines for experimental replications: A proposal. In: Proceedings of the 1st International Workshop on Replication in Empirical Software Engineering Research. International Conference on Software Engineering (ICSE) 2010. Cape Town, South Africa.

Clements, P. C., Northrop, L., August 2001. Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley.

Cook, S., Harrison, R., Lehman, M. M., Wernick, P., 2006. Evolution in software systems: foundations of the spe classification scheme. Journal of Software Maintenance 18 (1), 1–35.

Deelstra, S., Sinnema, M., Nijhuis, J., Bosch, J., September 2004. Cosvam: a technique for assessing software variability in software product families. In: IEEE Int. Conf. on Software Maintenance (ICSM). pp. 458 – 462.

Godfrey, M. W., German, D. M., 2014. On the evolution of lehman's laws. Journal of Software: Evolution and Process 26 (7), 613–619.
URL `http://dx.doi.org/10.1002/smr.1636`

Godfrey, M. W., Tu, Q., 2000. Evolution in open source software: A case study. In: IEEE International Conference on Software Maintenance (ICSM). IEEE Computer Society, Washington, DC, USA, pp. 131–142.

Gonzalez-Barahona, J. M., Robles, G., Herraiz, I., Ortega, F., 2014. Studying the laws of software evolution in a long-lived floss project. Journal of Software: Evolution and Process 26 (7), 589–612.
URL `http://dx.doi.org/10.1002/smr.1615`

Gupta, A., Cruzes, D., Shull, F., Conradi, R., Rønneberg, H., Landre, E., August 2010. An examination of change profiles in reusable and non-reusable software systems. J. Soft. Maint. and Evo.: Research Practice 22, 359–380.

Israeli, A., Feitelson, D. G., March 2010. The linux kernel as a case study in software evolution. Journal of Systems and Software 83, 485–501.

Jedlitschka, A., Ciolkowski, M., Pfahl, D., 2008. Reporting experiments in software engineering. In: Shull, F., Singer, J., Sjbøerg, D. I. K. (Eds.), Guide to Advanced Empirical Software Engineering. Springer London, pp. 201–228.

Kan, S. H., 2002. Metrics and Models in Software Quality Engineering, 2nd Edition. Addison-Wesley, Boston, MA, USA.

Kemerer, C., Slaughter, S., July/August 1999. An empirical approach to studying software evolution. IEEE Trans. on Soft. Eng. (TSE) 25 (4), 493 –509.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., Shin, Y., 00 1992. 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 (1-3), 159–178.

Lehman, M., September 1980. Programs, life cycles, and laws of software evolution. Proceedings of the IEEE 68 (9), 1060 – 1076.

Lehman, M. M., Ramil, J. F., Perry, D. E., 1998. On evidence supporting the feast hypothesis and the laws of software evolution. In: Proceedings of the 5th International Symposium on Software Metrics. IEEE Computer Society, Washington, DC, USA, p. 84.

Lehman, M. M., Ramil, J. F., Wernick, P. D., Perry, D. E., Turski, W. M., 1997. Metrics and laws of software evolution - the nineties view. In: Proceedings of the 4th International Symposium on Software Metrics. IEEE Computer Society, Washington, DC, USA, p. 20.

Lientz, B. P., Swanson, B. E., 1980. Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations. Addison-Wesley.

Lotufo, R., She, S., Berger, T., Czarnecki, K., Wkasowski, A., 2010. Evolution of the linux kernel variability model. In: Int. Conf. on Software Product Lines (SPLC). Springer-Verlag, Berlin, Heidelberg, pp. 136–150.

McCabe, T. J., 1976. A complexity measure. IEEE Transactions on Software Engineering (TSE) 2 (4), 308–320.

McGregor, J. D., 2003. The evolution of product line assets. Tech. rep., Software Engineering Institute, CMU/SEI-2003-TR-005.

Mende, T., Beckwermert, F., Koschke, R., Meier, G., April 2008. Supporting the grow-and-prune model in software product lines evolution using clone detection. In: European Conference on Software Maintenance and Reengineering (CSMR). pp. 163 –172.

Mens, T., Demeyer, S., 2008. Software Evolution. Springer.

Neves, L., Borba, P., Alves, V., Turnes, L., Teixeira, L., Sena, D., Kulesza, U., 2015. Safe evolution templates for software product lines. Journal of Systems and Software 106, 42–58.

Oliveira, R. P., Almeida, E. S., Gomes, G. S. S., 2015. Evaluating lehman's laws of software evolution within software product lines: A preliminary empirical study. In: Proceedings of the 14th International Conference on Software Reuse (ICSR), Miami, FL, USA. pp. 42–57.

Schmid, K., 2002. A comprehensive product line scoping approach and its validation. In: Proceedings of the International conference on software engineering (ICSE). New York, NY, USA, pp. 593–603.

Svahnberg, M., Bosch, J., 1999. Evolution in software product lines: two cases. Journal of Soft. Maint. and Evo.: Research and Practice 11 (6), 391–422.

Xie, G., Chen, J., Neamtiu, I., September 2009. 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.

Yan, X., Su, X. G., 2009. Linear Regression Analysis: Theory and Computing. World Scientific Publishing, River Edge, NJ, USA.