Software Product Lines in Value Based Software Engineering

Maria Teresa Baldassarre, Danilo Caivano, Giuseppe Visaggio*
Department of Informatics, University of Bari – RCOST Bari
{baldassarre, caivano, visaggio}@di.uniba.it
** contact author*

Objective: Evaluate the value of a product line in terms of maintainability, extensibility and configurability with refer to the interested stakeholders: customers, maintainers, producers.
Rationale: There are values that customers constantly require in a modern software application. Some of these values are supported by product lines. Nevertheless, in the industrial and scientific communities the conjecture that customer values clash with those of producers/maintainers is diffused.
Design of Study: we have designed and carried out a case study in an industrial context on an ongoing project to verify the validity of a product line in creating value for stakeholders. So data was collected as the project was being executed along a nine month period. Then, descriptive statistics and hypothesis testing were carried out.
Results: experience acquired during the execution of an industrial project has allowed the authors to point out the differences between program families and software product lines. Also, the case study has shown how product lines contribute to stakeholder value proposition elicitation and reconciliation.
Conclusions: This study has represented a first step towards analyzing the value that product lines represent for various stakeholders.

Value Based Software Engineering; Software Product Lines; Product Families; Industrial Case Study;

1. INTRODUCTION

In literature much confusion is generated between the two concepts of: Software Product Lines [5] and Program Families [4]. A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. On the other hand, a program family is a set of programs that share a common set of properties and then determine the specific properties of the individual family members. In this sense, a first aim of this paper is to point out the importance of the difference between the two approaches for software development through an experience that the authors have made during the execution of an industrial project.

In the past years, most of the software engineering research and practice has been carried out in a value-neutral setting, in which decisions made do not take into account the roles and value propositions of all the stakeholders (either they be users, acquirers, developers, maintainers, managers) involved in the process. In current contexts, where software engineering strongly influences system's cost, schedule and value, such an approach is no longer feasible. Important studies like the CHAOS report [2], carried out by the Standish Group, have attributed failure of many software projects to value oriented shortfalls. Consequently, software engineering principles and practices result being more difficult to share among different stakeholders if their value propositions are not considered. In this sense, Value Based Software Engineering (VBSE) plays an important role in integrating software system's stakeholder value propositions into all of its parts: definition, design, development and evolution.

In this work, the authors adopt the following definition of Value Based Software Engineering (VBSE): "the explicit concern with value concerns in the application of science and mathematics by which the properties of computer software are made useful to people" [1]. The stakeholders involved in a software engineering project are: Customers, Producers and Maintainers. If we were to consider the most important value propositions of these stakeholders, it would most likely arise that they are in conflict and must be in some way mitigated. Figure 1 represents part of the "spider web" [1] of the most frequent "model clashes" among stakeholders. So for example, customers desire many features, changeable requirements, easy availability and so on, but the aspects they give value to may clash with those of producers or maintainers. So for example, many features desired by the

customers clashes with the ease of meeting budget and schedule on behalf of producers who must take into account time restrictions, effort and available budget for satisfying customer requests.

Given these general considerations, a second aim of this study is to investigate the contribution that Product Lines can provide to stakeholder value proposition elicitation and reconciliation in the context of VBSE [3], with respect to the three quality characteristics of maintainability, configurability and extendibility.

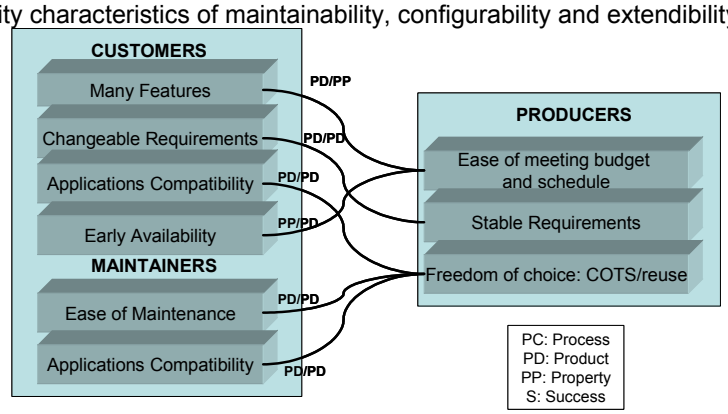

Figure 1: Value proposition Model-Clash spiderweb diagram

The investigation was carried out during the execution of an industrial project within an application domain that represented typical development issues of an enterprise application. Given the issues above, the enterprise desired adopting a product line approach. First, the confusion between product lines and program families highlighted weaknesses, which motivated improvements and led towards a correct interpretation and understanding of the product line concept and its benefits on a software system. The product line was defined taking into account value propositions of the involved stakeholders. The investigation has recently finished and data analysis is still being carried out, for this reason only part of the data have been analyzed and presented.

In the remaining part of the paper a brief description of product lines is given (section 2). We then illustrate the industrial case study in detail (section 3) and discuss results of data analysis (section 4). Finally conclusions are drawn.

2. BACKGROUND

In literature it is not seldom for confusion to be generated between the concepts of “product lines” and “program families”. Some authors state that they are the same. On behalf of the authors of this paper an important conceptual distinction must be made. Such distinction is made with respect to the chronology in literature of the definitions we refer to.

In [4] Parnas wrote: “A **Program Family** is considered a set of programs to constitute a *family*, whenever it is worthwhile to study programs from the set by *first* studying the common properties of the set, and *then* determining the special properties of the individual family members”. In [5] a **Software Product Line** is: “a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way”.

A program family is usually developed without the intent of satisfying a common set of features required by the market. Moreover it can be seen as a software application that is tailored to customer requirements by setting parameters. In this way the same software application is customized to specific needs. An example is SAP/R3, a general purpose information system that can be tailored in their behaviour by setting a set of specific parameters. So, in a program family, the same system is adapted by setting parameters to customer requirements.

A software product line is characterized by a set of products that share a common core of features and are distinguished from one another by additional variant capabilities which satisfy the needs of a specific stakeholder. An example is given by the Windows Operating Systems. In this case, there are various final products: *Windows XP*, *Windows Me*, *Windows 2003 Server*, *Windows 4.0 NT*, etc. Each product is an independent system made up of common features, such as libraries, interfaces, algorithms, and basic services, shared among all the systems, and specific variant features that are characteristic of each application, for example *Windows 2003 Server* has capabilities for network domain management, user account and profile management, network monitoring, etc; *Windows XP for tablet* PCs has capabilities for touchscreen management as an input device; *Windows Mobile* has capabilities for interface management that differs from other products.

If the two concepts were the same, what need would there have been to use a different term (product line) in the late 90s to identify something that had already been introduced in the early 70s (product family) by Parnas. Wouldn’t it be like reinventing the wheel?

The experience acquired during the project has pointed out the importance of differentiating program families from product lines. Also, in spite of the importance of the presented issues for practitioners, literature provides little

references on the application of value to product lines [3]. This paper represents a step in this direction and illustrates an empirical investigation on an ongoing industrial project.

VBSE was firstly significantly addressed by Boehm [11]. The value-based movement became of high interest for the software engineering community during the 90's with the Win Win model which stressed the multi-stakeholder perspective and with contributions from empirical and evidence based software engineering, value based approaches, agile software development and risk driven approaches. In [1] Boehm revisits and puts together all the contributions.

3. CASE STUDY

LEGOM is an application of an Italian SME, that has been on the market for 3 years. It supports the legal office of a bank or an insurance company in managing risky credits. In the respect of the norms that regulate this application domain, each bank adopts a specific approach, based on past acquired experience, with their clients. For nationwide banks such decisions must adapt to the various territorial and economical contexts which have different cultures, and different manners for facing the initiatives to undertake. Many institutions also rely on external attorneys that adapt credit return procedures to their experience. So, in facing credit return issues there are common features applicable nationwide and variant aspects that apply only to local contexts. Experience is another variant factor.

In this context, banks and insurance companies are the *customers* of LEGOM, while SME's staff involved in the LEGOM development teams are the *maintainers* and *producers*. We have considered the spiderweb diagram in [1] as a starting point for identifying value propositions of stakeholders and have refined it into what appears in Figure 1 according to what stakeholders have pointed out during the project execution. Moreover, the values desired by customers were: *Many Features*, used for defining client characteristics and understanding how to enact credit return; *Changeable Requirements* because experience acquired in time may lead to modification of features; *Early Availability* because customers require that requested changes be usable in a short time. *Application Compatibility* in that the features of an application must be coherent with the social context they are used in and with the other applications they must interact with. On the other hand, maintainers desire *ease of maintenance* in satisfying maintenance requests in short time with low effort, and that the applications integrated in the software system are compatible with one another (*application compatibility*). Finally developers of a software system desire that the requirements of the system to develop are stable and not subject to change in short periods (*stable requirements*) so budget and schedule can be planned and respected (*ease of meeting budget and schedule*). Also, developers desire freedom of choice of COTS that should make up the final software system and the possibility of reusing existing code for developing requirements.

In order to face and overcome the clashes related to stakeholder value propositions, the SME migrated the LEGOM application into a product line [5, 6]. This was done through three steps:

1. analysis of existing versions in order to point out: a) invariant core and b) variant components
2. distinction of characterizing parameters for each version
3. formalization of the relation between characterizing parameters and variant components. Each combination between a set of characterizing parameters and variant components determines a profile, i.e. the applications of LEGOM implemented for customers.

After the migration, a case study was carried out in the SME in order to evaluate how the product line provided value proposition elicitation and reconciliation with respect to three quality characteristics of maintainability, configurability and extendibility. The case study was carried out on field as LEGOM was used by its customers and maintenance requests were forwarded during an observed time period of three trimesters. The relations between the stakeholder value propositions of Figure 1 and the three quality characteristics observed in the product line are pointed out in Table 1. Moreover for each quality characteristic, the table lists the clashes between stakeholder value propositions and motivates our conjecture on the how the quality characteristic measured in a product line reconciles the clash.

Quality characteristic	**Clash of stakeholder values**	**Relation**
Maintainability	Changeable Requirements & Stable Requirements	Maintainability assures that changes, due to changeable requirements desired by customers, are localized in small parts of the variant components and do not impact on the remaining modules of the product.
	Ease of Maintenance & Freedom of Choice	Developer's reused software may not be easy to maintain causing a model clash in that each time a feature or requirement must be modified it is necessary to verify and validate the correctness of the changes keeping in mind the relations among the numerous components. We hypothesize that in a product line the integrated components favour maintenance interventions.
Extendibility	Early Availability & Ease of meeting budget and	The customer's desire of having the software available in short time clashes with developer's budget and schedule. We hypothesize that a product line allows to

	schedule	develop features that can be used by customers and gradually extend them.
Configurability	Many Features & Budget Schedule	An application with many features makes production costly and risks to be unable to meet budget and schedule restrictions. Configurability verifies that the product line has a very large set of available features manageable according to customer needs. This overcomes the clash with budget and schedule.
	Applications Compatibility & Freedom of Choice	The developer's choice of COTS or reused components may be incompatible with customers and maintainer's other applications because configuration management is costly and may be unreliable due to the large amount of components to manage. We hypothesize that in a product line choice of software components is compatible with existing applications.

Table 1: Relations between Stakeholder Values and Quality Characteristics

In the following, details of the case study planning and enactment are provided. Note that, given the characteristics of a case study carried out on field, the level of control is not the same as for an experiment in vivo. However, authors will provide a description as detailed as possible to allow other interested practitioners and researchers to perceive the aspects and lessons learned by carrying out the study.

3.1. Definition of the Case Study

The case study aimed at assessing value elicitation of stakeholders involved in the LEGOM product line in terms of maintainability, configurability, and extendibility of the application. So, three research goals have been defined:

RG1: *Analyze* the maintenance process of the LEGOM product line; *For the purpose of* evaluating it; *With respect to* **maintainability**; *From the point of view of* maintainers; *In the context of* an Italian SME

RG2: *Analyze* the maintenance process of the LEGOM product line; *For the purpose of* evaluating it; *With respect to* **configurability**; *From the point of view of* maintainers; *In the context of* an Italian SME

RG3: *Analyze* the maintenance process of the LEGOM product line; *For the purpose of* evaluating it; *With respect to* **extendibility**; *From the point of view of* maintainers; *In the context of* an Italian SME

These goals have been assessed during three trimesters of execution of the case study, according to the maintenance requests made by stakeholders/customers of the product line. For clearness, a maintenance request has been classified as corrective (for eliminating defects of the system in use); adaptive (for adapting existing features of the application to customer needs), evolutional (for adding new capabilities to the application). The application domain of LEGOM is given by the body of knowledge related to legal procedures in credit or insurance institutions.

3.2. Planning

This phase is described according to the guidelines in [7] i.e. variables selection, selection of subjects, hypothesis formulation.

3.2.1. Variables Selection

The dependent variables express the quality characteristics previously mentioned. They have been detailed and measured through a set of metrics and are specified in the following.

Maintainability Indicator (I_M):

Metric	***Definition***
Number of Reworks (NR)	Number of times it was necessary to iterate the maintenance process to overcome the defects encountered following to a maintenance request (MR). The NR is considered with respect to the overall MR and to the specific types (corrective, adaptive, evolutional): $MR_{CORR,}$ MR_{ADEG}, MR_{EVOL}
Maintenance Requests (MR)	The set of modification requests made by the customers of LEGOM to the producers/maintainers of the product line during the case study observation period. The MR are considered with respect to their totality and to the specific types. (MR_{CORR}, MR_{ADEG}, $MR_{EVOL)}$.

Maintainability Indicator (I_M)	This indicator is measured as the ratio between the number of reworks for satisfying a maintenance request and the overall number of requests made. It is evaluated along the observation periods, as maintenance requests are made during the use of LEGOM on behalf of its customers. Also, it is evaluated with respect to the requests made and to the types of maintenance requested (corrective, adaptive, evolutional). Let MR_i be the i-th maintenance request, for i=1…N and N=total number of requests along the observation period. Let NR_i be the number of reworks carried out to satisfy the i-th maintenance request. The maintainability index for the i-th MR is defined as: $\boxed{I_M(MR_i) = \frac{\sum_{j=1}^{i} NR_j}{i}}$, and with respect to each type of MR we define: $I_{M\_CORR}(MR_{CORR\_t}) = \frac{\sum_{j=1}^{t} NR_{CORR\_j}}{t}$, $I_{M\_ADEG}(MR_{ADEG\_k}) = \frac{\sum_{j=1}^{k} NR_{ADEG\_j}}{k}$, $I_{M\_EVOL}(MR_{EVOL\_s}) = \frac{\sum_{j=1}^{s} NR_{EVOL\_j}}{s}$

Extendibility ($I_{E/C}$ and I_{CM}): it is evaluated through two indicators.

Metric	*Definition*
Number of Created Components ($\#C_{cr}$)	Number of new components that are created following to a MR
Number of Eliminated Components ($\#C_{el}$)	Number of components that are eliminated following to a MR
Number of Modified Components ($\#C_{mo}$)	Number of components that are modified following to a MR
Extendibility Indicator ($I_{E/C}$)	Is given by the ratio between the total number of components created and eliminated for satisfying the i-th MR, and the effort required for carrying out the request. It is calculated with respect to the MR made during the entire observation period. It is calculated as: $I_{E/C}(MR) = \frac{\#Ccr + \#Cel}{effort}$
Modifiability Indicator (I_{CM})	Is given by the ratio between the total number of components modified for satisfying the i-th MR, and the effort required for carrying out the request. It is calculated with respect to the MR made during the entire observation period. It is calculated as: $I_{CM}(MR_i) = \frac{\#Cmo}{effort}$

Configurability (**I_{Ci}**, **Nr.Profiles**): evaluated through two indicators

Metric	*Definition*
Number of Applications (#App)	Number of applications in use that have been defined through the LEGOM product line. Each application is a specific configuration of the product line.
Indicator of Components Usage (I_{Ci})	It is given by the ratio between the number of occurrences of a variant component Ci in a specific application (configuration) in use, of the LEGOM product line and the total number of existing applications. In particular, let C_i be the i-th variant component (for i=1..M) of a total of M components that make up the variant part of the LEGOM product line. Let A_j be the j-th application of the product line (for j=1…K), made up of a total of K applications. The following table can be defined:

components/applications	A1	A2	Aj	…	AK
C1	X		X		X
C2					
Ci		X	X		
….					
CM		X			X

The intersection (i-j) indicates that the i-th component is part of the configuration that characterizes the j-th application of the LEGOM product line. So, if we define:

$c_{ij} = \begin{matrix} 1 if(i,j)="X" \\ 0 if(i,j)="-" \end{matrix}$ and $I_{Ci} = \frac{\sum_{i=1}^{K} C_{ij}}{K}$ for i=1…M

the indicator is calculated for each component. (for example, in the table above I_{C1} = 3/K)

Number of Profiles (#Pr)	The total number of configurations of customer parameters. This metric identifies the types of customers that use a product of the product line.

3.2.2. Selection of Subjects

The subjects involved in the case study were on one hand, producers/maintainers of the SME with experience in development and maintenance of LEGOM and on the other domain experts. The latter define the various profiles and represent LEGOM customers. They are the stakeholders that make maintenance requests as an application of the product line is used.

3.2.3. Hypothesis Formulation

Operationally, the research goals have been evaluated through the metrics listed above. The baselines for the measures were specified by the SME's quality team according to their experience acquired in previous projects and the past performances achieved in the enterprise. So, the measures were familiar within the enterprise. In the following we briefly motivate their values.

- *Maintainability (I_M)*: for this indicator the expectation is that a system is more maintainable as the number of reworks for a MR decreases. So, a low number of reworks is most likely related to a localized modification. This indicator is expected to be less than 0.6, i.e. no more than three reworks for each five maintenance requests were to be carried out (3 reworks / 5 MRs = 0,6 as shown in the tables in section 3.2.1). These considerations are also valid for the specific types of MRs (I_{M_CORR}, I_{M_ADEG} and I_{M_EVOL}).
- *Extendibility*: $I_{E/C}$ evaluates the effort spent for extending or reducing functionalities of the product line, i.e. the degree with which the SME's developers are able to add or eliminate software components following to a MR. On the other hand, I_{CM} refers to the effort required for modifying components of the product line impacted by a MR. If the MR doesn't require creation/elimination (or modification) of components, $I_{E/C}$ (respectively, I_{CM}) will be zero, otherwise its value will increase as the effort for creating/eliminating (or modifying) components decreases. The baseline for this indicator is of 0,1, i.e. we expect that each created/eliminated (or modified) component does not require more than 10 person days (1 component / 10 person days = 0,1 as appears from the table in section 3.2.1.). The closer this indicator is to "1", the less the effort needed for creating/eliminating or modifying the components.
- *Configurability*: represents the degree to which each component is used within the applications of the product line. Overall, we expect that the indicator of components usage tends to increase as the number of applications increase. This would mean that the products of the product line and the components of each product are used. Each time I_{Ci} is greater than 75%, the component becomes a *core* part of LEGOM.

Hypothesis testing has been carried out to answer the research goals. The hypotheses have been tested with respect to the MRs received during the three observation trimesters. They are defined with respect to the indicators of the three quality characteristics previously mentioned:

- $\mathbf{H_{i0}}$: There are no statistically significant differences in the values of the indicator I_i between the trimesters of observation
- $\mathbf{H_{i1}}$: There are statistically significant differences in the values of the indicator I_i between the trimesters of observation

Where the set of indicators $I_i = \{I_M, I_{M\_ADEG}, I_{M\_EVOL}, I_{M\_CORR}, I_{E/C}, I_{CM}, I_{Ci}\}$, as shown in Figure 2. Also, as it arises from the arrows in the figure, tests have been carried out *between* trimesters to identify significant differences of the same indicator from one observation period to another.

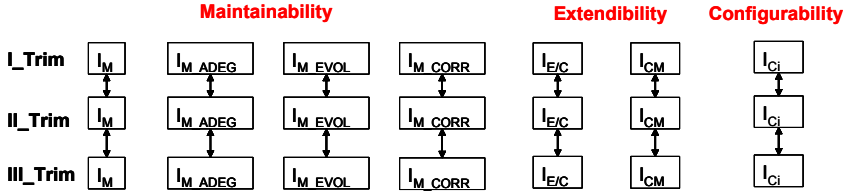

Figure 2: Hypotheses testing

The indicators were expected to achieve the baseline values. However it is also reasonable for a slight decrease in values over time, due to constant maintenance the system is subject to. So, if on one hand, over time the components benefit from test carried out following to maintenance, leading to less effort for satisfying other MRs of the same type, on the other, as stated by the laws on software maintenance/evolution [8, 9], it is inevitable for a maintained software system to degrade in time. This means that, as the product line increases in terms of products and components, its complexity increases and so does the effort for satisfying MRs. Overall, we expected that such degradation occurred slowly, thanks to the product line structure of the application.

3.3. Case Study Design

The case study is planned in three observation periods, each of three months. We will therefore refer to trimesters of observation: I_Trim, II_Trim, III_Trim. During each observation period, maintenance requests (MR) were made by the stakeholders (customers) of the various LEGOM products. So considering that MR={MR_1, …, MR_N} is the set of maintenance requests satisfied in an observation period, the case study design was defined as follows: :

I_Trim	II_Trim	III_Trim
Tr1 O1, Tr2 O2, …, Trk Ok	Trk+1 Ok+1, …, Trj Oj	Trj+1 Oj+1, …, TrN ON

Table 2: case study design

$\mathbf{T_{ri}}$	T_{rc} = treatment of corrective maintenance T_{ra} = treatment of adaptive maintenance T_{re} = treatment of evolutional maintenance
$\mathbf{O_i}$	Observation of the indicators according to the type of T_{ri}.

Table 3: classification of maintenance interventions

T_{ri} is a maintenance intervention that can be classified as in Table 3. Each time a maintenance request begins in the i-th trimester and ends in the i+1-th trimester, it is considered as part of the i+1 trimester.

3.4. Operation

After migrating the client-server version of LEGOM to a product line, it was installed and configured according to each customer profile. Each application was then put in practice. As each one was used, customers forwarded their MRs which were satisfied by the SME's producers/maintainers during the three observation periods. As MRs were made and satisfied, dependent variables were collected.

Note that, being the investigation a case study carried out on an ongoing project, it had a low level of control. In particular the MRs could not be decided previous to operation, rather they were defined as the LEGOM product line applications were put in practice. The strength, however, is given by the fact that the MRs are representative of real context needs.

T_{ri} is a maintenance intervention that can be classified as in Table 3. Each time a maintenance request begins in the i-th trimester and ends in the i+1-th trimester, it is considered as part of the i+1 trimester.

4. DATA ANALYSIS

The project has recently ended and data analysis is still in execution. Moreover, till now we have analyzed results on two of the indicators (maintainability and extendibility). We are not able to make any considerations on configurability of the LEGOM product line. So, this section includes our preliminary results on the analyzed data.

First of all descriptive statistics have been used for graphically representing collected data. We have used box plots for comparing the trend of maintainability and extendibility indicators between successive trimesters.

For what concerns hypothesis testing, from section 3.2.3. it can be seen that the test hypotheses compared more than two samples made up of observations coming from distinct treatments. For this reason we planned to use an ANOVA - ANalysis Of Variance test. This test verifies for significant differences between means of samples. However, since the samples were not normally distributed, a non parametric alternative to the ANOVA was adopted, i.e. Kruskal-Wallis. The Kruskal-Wallis ANOVA [10] by ranks test assumes that the variable under consideration is continuous and measured on at least an ordinal scale. The test assesses the hypothesis that the different samples in the comparison were drawn from the same distribution or from distributions with the same median. Thus, the interpretations of the Kruskal-Wallis test is basically identical to that of the parametric one-way ANOVA, except that it is based on ranks rather than means. For hypothesis testing an $\alpha = 5\%$ was considered.

4.1. Discussion of Obtained Results

Throughout the three observation periods, a total of 8 applications of the product line were implemented among 5 different customers who forwarded 63 MRs distributed throughout each trimester as described in Table 4. Figure 3 plots the trend of the indicators calculated according to the MRs received during the observation period.

	Corrective	Adaptive	Evolutional
I_Trim	3	6	11
II_Trim	2	5	15
III_Trim	1	7	13

Table 4: Distribution of MRs along the three trimesters

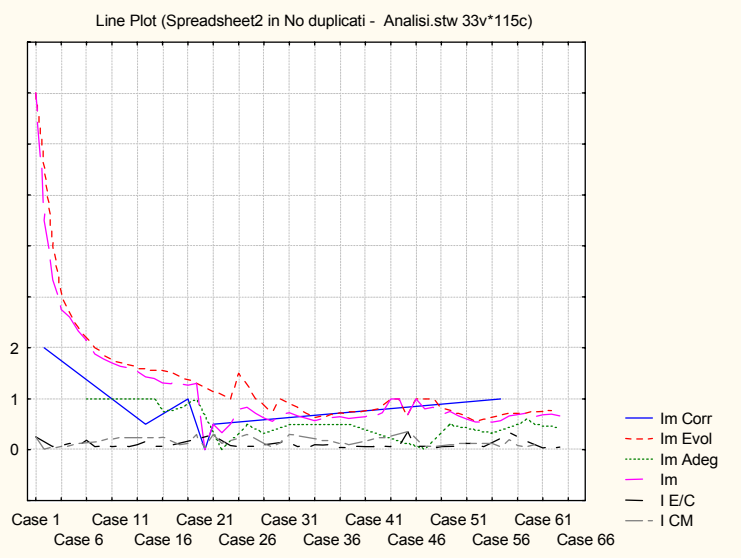

Figure 3: trend of indicators with respect to MRs during the three trimesters

It is evident how in I_Trim (case 1 – case 20) the data is quite discordant from the rest of the period. Such poor data was attributed to a confusion made between the two concepts of product families and product lines. Moreover, within the first trimester the difference was not yet acquired by the developers (producers & maintainers) of the SME, and what they thought was a product line version of LEGOM actually was a family of programs.

This confusion led to a product line difficult to manage, with 50 variant conditions, many of which binary. So the total number of possible products that could be defined was outrageous (2^{50}). Such a situation presents many issues:

- it is difficult to specify a product line, and validate the correctness of the specifications;
- with such a high number, it is impossible to test all the resulting products;
- maintenance requires a lot of effort;
- collaboration with customers and stakeholders for identifying the most suitable product for their profile is onerous.

As the application was migrated to what developers thought was a product line, the error made was that characterizing parameters referred to both variant features (software product line requested by customers) and variants of software components (flexibility desired by designers). The distinction of these two concepts then suggested to produce a product line with a reasonable amount of products during the specification phase, and in the design phase components were identified. After this improvement intervention, the data in the following two trimesters (II_Trim and III_Trim) reflected our expectations.

Since the data in the first trimester were biased, they were excluded from further data analysis. For space reasons we will not report the data related to this trimester. Consequently, hypothesis testing was carried out on the remaining two trimesters, and a Mann-Whitney U test [10] was carried out instead of a Kruskal Wallis ANOVA as planned in the case study design, because the treatments were two and no longer three. This test assumes that the variable under consideration was measured on at least an ordinal scale. The interpretation of the test is essentially identical to the interpretation of the result of a *t*-test for independent samples, except that the U test is computed based on rank sums rather than means. The U test is the most powerful (or sensitive) nonparametric alternative to the *t*-test for independent samples.

4.1.2. Hypothesis Testing

Maintainability: This indicator is expected to be less than 0.6, i.e. no more than three reworks for each five maintenance requests were to be carried out. These considerations are also valid for the specific types of MRs as specified in 3.2.3. Given the low number of corrective MRs, the sample was considered too small to be significant for testing. Below, box plots and Mann Whitney tests carried out on each of the maintainability indicators are reported.

In general, the indicators satisfy the expected baselines in time. Also, it seems that I_{M_ADEG} and I_{M_EVOL} are quite stable between II_Trim and III_Trim as it appears in Figure 4 and

Figure **5**. The following considerations can be made:

- the difference between adaptive and evolutional maintenance can be attributed to the fact that it is reasonable for an adaptive MR, implying modification of existing features, to require less rework than an evolutional maintenance request, which involves enriching the product line with new features;
- in case of adaptive maintenance, modifications impact features that developers are familiar with. They are supported by documentation and test cases for regression tests after integrating the component in the product line.

- in case of evolutional maintenance, the risk is that requested functionalities are not part of the enterprise assets, and are therefore more difficult to understand. In each case, statistical tests, in accordance to our expectations, do not point out significant differences.

The mean values of I_{M_ADEG} are of 0,36 and 0,4 respectively in II_Trim and III_Trim; for I_{M_EVOL} they are 0,8 and 0,78. Such difference is more evidenced in the I_M indicator because it groups all the types of MRs and better points out the consequences of software degradation following to maintenance interventions. In fact, in Figure 6 there is a more sensible difference between II_Trim and III_Trim, due to the fact that all types of requests are considered.
For this reason the differences are also statistically significant, as shown in Table 7. Then again, this is a calculated risk when a product line approach is used, where addition of new features and the consequent evolution of the entire system may be more costly than modification of existing ones because the first must be designed following the product line approach and safeguard the current components. The mean values of the maintenance indicator in the two trimesters are respectively of 0,6 and 0,71. The increase in the second trimester is attributable to the reasons discussed above. So, given the baseline for this indicator (0.6), we can consider the RG1 achieved. Discussion is done in the next section.

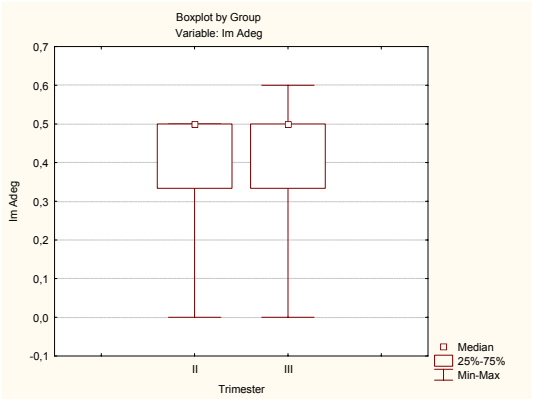

Figure 4: Box Plot for Adaptive Maintenance Indicator (I_{M_ADEG}) in II_Trim and III_Trim

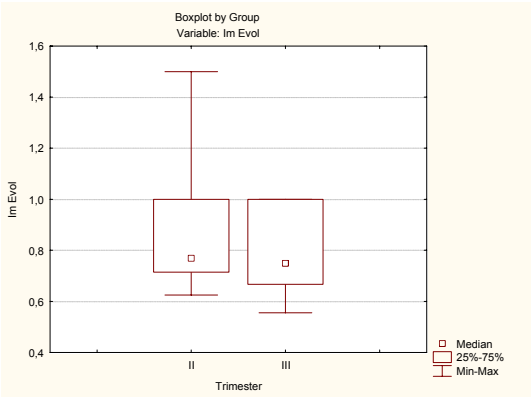

Figure 5: Box Plot for Evolutional Maintenance Indicator (I_{M_EVOL}) in II_Trim and III_Trim

variable	Mann-Whitney U Test (Spreadsheet2 in Analisi Ease.stw) By variable Trimester Marked tests are significant at p <,05000 Rank Sum II	Rank Sum III	U	Z	p-level	Z adjusted	p-level	Valid N II	Valid N III	2*1sided exact p
Im Adeg	30,50000	47,50000	15,50000	-0,324799	0,745333	-0,348095	0,727769	5	7	0,755051

Table 5: Mann-Whitney test for I_{M_ADEG}

variable	Mann-Whitney U Test (Spreadsheet2 in Analisi Ease.stw) By variable Trimester Marked tests are significant at p <,05000 Rank Sum II	Rank Sum III	U	Z	p-level	Z adjusted	p-level	Valid N II	Valid N III	2*1sided exact p
Im Evol	232,0000	174,0000	83,00000	0,667947	0,504168	0,674063	0,500272	15	13	0,524956

Table 6: Mann-Whitney test for I_{M_EVOL}

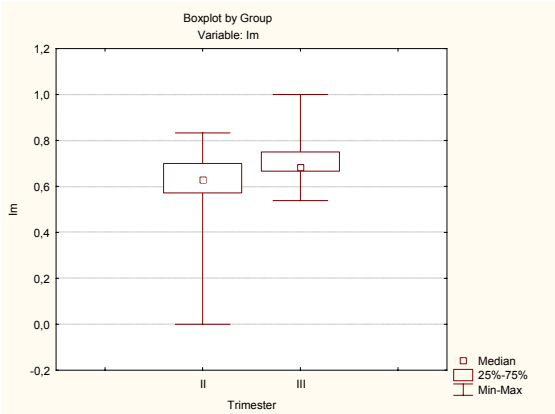

Figure 6: Box Plot for Maintenance Indicator (I_M) in II_Trim and III_Trim

variable	Mann-Whitney U Test (Spreadsheet2 in Analisi Ease.stw) By variable Trimester Marked tests are significant at p <,05000 Rank Sum II	Rank Sum III	U	Z	p-level	Z adjusted	p-level	Valid N II	Valid N III	2*1sided exact p
Im	403,0000	543,0000	150,0000	-1,96801	0,049067	-1,97316	0,048478	22	21	0,049777

Table 7: Mann-Whitney test for I_M

Extensibility: The baseline for this indicator is of 0,1, i.e. we expect that each created/eliminated (or modified) component does not require more than 80 person hrs. Below, the box plots for both indicators are reported, together with the Mann Whitney test results.

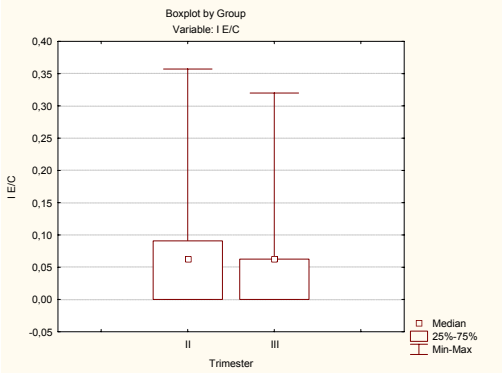

Figure 7: Box Plot for $I_{E/C}$ in II_Trim and III_Trim

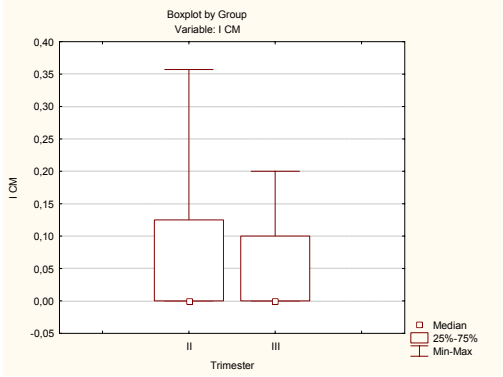

Figure 8: Box Plot for I_{CM} in II_Trim and III_Trim

	Mann-Whitney U Test (Spreadsheet2 in Analisi Ease.stw) By variable Trimester Marked tests are significant at p <,05000									
variable	Rank Sum II	Rank Sum III	U	Z	p-level	Z adjusted	p-level	Valid N II	Valid N III	2*1sided exact p
I E/C	518,5000	427,5000	196,5000	0,838228	0,401903	0,860545	0,389489	22	21	0,405763

Table 8: Mann-Whitney test for $I_{E/C}$

	Mann-Whitney U Test (Spreadsheet2 in Analisi Ease.stw) By variable Trimester Marked tests are significant at p <,05000									
variable	Rank Sum II	Rank Sum III	U	Z	p-level	Z adjusted	p-level	Valid N II	Valid N III	2*1sided exact p
I CM	501,0000	445,0000	214,0000	0,413040	0,679577	0,449272	0,653236	22	21	0,691465

Table 9: Mann-Whitney test for I_{CM}

From Figure 7 and Figure 8 it appears that these two indicators are similar with respect to II_Trim and III_Trim. In fact the mean values for $I_{E/C}$ are respectively 0.08 and 0.058, while for I_{CM} they are 0,07 and 0,049. Furthermore, the differences between trimesters of each sample are not statistically significant. The following considerations can be made:

- although the complexity of the system inevitably changes in time due to maintenance interventions, creation/elimination and modification of components are not significantly impacted;
- this was attributed to the product line structure given to LEGOM.

Considering the expected baseline for these indicators (0.1), we can consider the RG2 achieved. Discussion is demanded to the next section.

5. CONCLUSIONS

This paper has focused on investigating a product line approach and the value generated for its stakeholders (Customers, Maintainers, Producers). The investigation was carried out through an industrial case study in an Italian SME. The aim of the case study was to: migrate an existing application for legal office management of banks and insurance companies (LEGOM) to a product line architecture; evaluate the resulting architecture together with the processes used for its development and evolution with respect to three quality characteristics (Maintainability, Extendibility, Configurability) of interest for the LEGOM stakeholders.

Data analysis has pointed out elements of interest:

- it is important to distinguish between software product lines and program families in that they differently impact on consequent budget and schedule restrictions. We have acquired such differences during the execution of the industrial project. In particular, the data collected in the first trimester, when such difference was not clear to the producers and maintainers of LEGOM impacted on the collected indicators. Moreover, in this period the data collected with refer to the three quality characteristics were useless and not representative of the value of the Product Line for its Stakeholders. The confusion generated made the MRs difficult to satisfy;
- for what concerns maintainability, in a product line values such as freedom of choice may lead to many products. However, being a product line, changes impact on few components. If such components are part of the variant assets, they refer to the application that is used by a specific customer; if the components are part of the core assets, the maintenance is carried out only once and is extended to all the products used by all customers;
- for what concerns extendibility, since applications of a product line are a combination of core and variant assets, each time a new product is requested it can be chosen among existing components (in this case

extensibility consists in configuring a new application by using existing components) or may require development/choice of new ones (in this case, extensibility will require more effort and costs which will most likely be mitigated by the fact that in the future, the same requests may be made by other customers).

While we have obtained results from data analysis on maintainability and extensibility indicators, we have not analyzed data on configurability yet. However, it is the case to comment on our expectations and conjecture. In a product line, an application has a specific configuration of common and variant parts. New configurations are characterized by the same core of components and by a choice of variant components that belong to the product line assets and that are documented. So we expect configurability to also be mitigated in a product line.

So, if the product line approach is correctly enacted, it is beneficial with respect to traditionally designed applications. A product line structure allows to face requirements changeability with less effort in that each customer is associated to a profile which leads to the most suitable application for their needs. Since each profile is traceable and may change in time, producers/maintainers are able to face maintenance requests with reasonable effort and identify components that should be created, eliminated or modified. So, overall, a system designed following a product line approach and subject to continuous maintenance is less prone to degradation symptoms in time.

This study has represented a first step towards analyzing the value that product lines represent for various stakeholders. It has validated our conjecture that a properly designed product line assures the values desired by customers in an enterprise application. Moreover, a product line mitigates the clashes between these values and those of producers and maintainers.

In spite of the weaknesses of a case study, given the nature of the investigation, it encloses many benefits: the study was carried out during an ongoing project, on a real product line application used by customers; maintenance requests expressed actual requirements that customers had with respect to the application configured to their specific needs; results apply to a real context.

REFERENCES.

[1] Biffl S., Aurum A., Bohem B., Erdogmus H., Grunbacher P.(2006) *Value based Software engineering,* Springer Verlag, Berlin.

[2] CHAOS Report (1995), The Standish Group, (www.standishgroup.com).

[3] Faulk S.; Harmon R., Raffo D. (2000), Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering, *Software Product Lines: Proceedings of the First Software Product Line Conference (SPLC1)*. Denver, Colorado, August 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 205-224..

[4] Parnas D.L., (1976), On the design and development of program families, IEEE Transactions on Software Engineering, Vol.SE-2 pp 1-9.

[5] Clements,Paul & Northrop, Linda. *Software Products Lines: Practice and Patterns*. Boston, MA: Addison – Wesley, 2002.

[6] CMU/SEI-2001-TR-001, ESC-TR-2001-001 (2001), Product Line Analysis: A Practical Introduction, available at: http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr001.pdf

[7] Wohlin C, Runeson P, Host M, Ohlsson M.C., Regnell B, Wesslèn A, (2002) *Experimentation in Software Engineering*, Kluwer Academic Publishers.

[8] 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, *Proceedings 4th Int'l Metrics Symposium.*

[9] Lehman M.M. (1978), Laws of Program Evolution-Rules and Tools for Programming Management, *Proceedings of the Infotech State of the Art Conference*, Pergamon Press, April, pp.1-25

[10] Marasculio L.A., Serlin R.C., (1988), *Statistical Methods for the Social and Behavioral Sciences*, W.H. Freeman and Company, New York, USA.

[11] Boehm B., (1981), *Software Engineering Economics*, Prentice Hall.