

Quality in Development Process for Software Factories According to ISO 15504

Kenyer Domínguez

Decanato de Investigación y Desarrollo (DID)
Universidad Simón Bolívar. Apartado Postal 89000, Caracas 1080-A. Caracas - Venezuela.
kdoming@usb.ve

María Pérez, Luis E. Mendoza, Anna Grimán

Laboratorio de Investigación en Sistemas de Información (LISI),
Universidad Simón Bolívar. Apartado Postal 89000, Caracas 1080-A. Caracas - Venezuela.
movalles@usb.ve, lmendoza@usb.ve y agriman@usb.ve

ABSTRACT

Currently the concept of Software Factories (SF), where reuse plays a leading role, is being adopted. Due to the different approaches in this area, and although SF concept is not new in Software Engineering, it is still not mature enough to clearly identify the treatment of certain variables within the process. One of these variables is Quality. Therefore, this paper presents a historical review of the SF concept, proposes an ontology based on the most recent definitions and establishes a relationship between these concepts and the ISO 15504 standard, with the purpose of specifying the systemic quality in software developer companies that decide to implement an SF strategy.

Keywords: Software Factory, ISO 15504, ontology, reuse, software process quality.

1. INTRODUCTION

Given the demanding requirements as to meeting schedules and budgets that software developing companies go through constantly, the efficiency of the developing process has become a growing necessity. Nevertheless, this is a variable that does not necessarily favor effectiveness.

Consequently, reuse within the software production process has been center of interest long ago, without obtaining any real success. At present, developers create new systems and applications from scratch by using generic tools, inadequate processes and customized architectures; a task that is time consuming and can be the origin of some mistakes.

Being aware of this reality, some organizations have established different working strategies and have even gathered the best practices in this subject with the purpose of industrializing software developing. As a result, the concept of Software Factory (SF), where reuse plays a leading role, is currently being retaken. Due to the different approaches in this area, and although SF concept is not new at all in Software Engineering, it has not achieved the necessary conceptual maturity to clearly identify the treatment of certain variables within the process, one of them being Quality.

This paper proposes an ontology which helps understand the different meanings of the SF concept, as well as the characteristics, innovations, and implications of the most recent approaches in the SF Software Engineering field.

Mainly, the objective of this article is to present the correspondence between the ISO 15504 standard and the concepts identified in the SF ontology, with the purpose of estimating the presence of the software process quality in any organization that works according to the SF characteristics.

This paper is part of a research in progress whose objective is to estimate the systemic quality in software developer companies that decide to implement an SF.

The background of this research (section 2) is presented below, followed by the methodology used (section 3); a description of each one of the processes and concepts related to SF (section 4); the integrated conceptual model (section 5); and the comparison of the processes identified in an SF with those evaluated in the ISO 15504 standard (section 6). Finally, a number of conclusions and recommendations are proposed.

2. BACKGROUND

Great efforts have been made in Software Engineering in order to compensate for the lack of a well defined and well managed development process, which affects developers as well as users and customers because the product is not obtained within the desired time and the estimated costs. The application of an SF model is one of those attempts where code reuse, process specification, and the model-driven development play a leading role.

However, the term SF has already been used in Software Engineering. It was used for the first time between 1950 and 1960 in the United States and Europe to improve productivity and software development through standard tools and reuse of methods and components; then, between 1970 and 1980, computer manufacturers in Japan adopted the new concept and considerably improved software development [15].

In the 90's, substantial research in relation to the term SF was reported [15]. Later, Microsoft approach of the concept was announced attached to its product "Visual Studio Team System 2005" and its Microsoft .NET platform [17]. Almost by the same time, Fabri and other researchers [5] presented the techniques used for developing an SF in Brazil, where an SF based on Open Source is still being developed [18].

The most recent approaches found in the literature [5][8] represent an important background of the SF subject. A developer company without the following characteristics cannot be considered an SF: mass and large scale software production, task and control standardization, and work division and automation [5].

Furthermore, according to Humprey [9] and Pressman [19], the quality of the final product is strongly influenced by the quality of its development process, in agreement with the functional requirements explicitly established. These models were reviewed but the standard ISO 15504 [10][21] was taken as a basis for this research..

3. METHODOLOGY

In order to relate SF ontology with ISO 15504 [10][21], the Methodological Framework of the Information Systems Research Laboratory (LISI, according to its acronym in Spanish), at the Universidad Simón Bolívar, is used. This framework is inspired in the Action Research method [2], and uses the DESMET methodology [12] for the evaluation phase.

The Methodological Framework consists of 11 steps or activities grouped in 5 wider phases, which correspond with the phases of the Action Research Method: (1) Documental Research, (2) Background analysis, (3) Formulation of objectives and scope of the research, (4) Formulation of the research methodology, (5) Proposal of comparison of the processes identified in an SF with the processes evaluated in the ISO 15504 standard, (6) Context analysis, (7) Application of the DESMET method, (8) Evaluation through the method produced by DESMET, (9) Results analysis, (10) Definition of the scope of the next iteration and, (11) Conclusions and recommendations.

Due to presentation restrictions in this paper, the proposed steps are not detailed; nonetheless, the activities carried out in each one of them are emphasized below: (1) and (2): no ontology in this subject was found, but various SF models; (3) and (4): the objectives and scope of the research were specified within the limits of ISO 15504; (5): the construction of the ontology and its correlation with the standard selected is presented, this topic is dealt with in more detail throughout this paper; (6), (7) and (8): once the correlation is established, DESMET is used to validate it; this point constitutes the basis of the future steps of the present research; (9), (10) and (11): conclusions and recommendations are listed at the end of the paper.

4. SOFTWARE FACTORY PROCESSES

The search for effectiveness and the motivation generated by competitiveness have been a constant in any business. Software industry does not escape to this reality. As a consequence, organizations which develop software spare no effort to go from the traditional way of development to mass production; in this context a question appears: which are the typical activities in each case?

According to Coe [4], cited in [20], the software industry represents a continuum of activities located between two extremes: 1) the development of software customized to the needs of a determined customer, and 2) the process of packed software production. The steps in the first case are [4]: Problem identification ; Justification / Feasibility study; Analysis and Determination of Requirement; Design; Testing; Delivery / Installation and; Maintenance.

Similarly, Coe identified the sequence of steps for the production process of packed software [4]: Development; Testing; Duplication of disks; packing and manuals; Sales and marketing; Distribution; and Support.

According to Rojas [20], the similarities of the two versions offered by [4] are more than those perceived at the first sight, because "*Product Development*" includes a complexity that can be disaggregated to favor understanding. In this sense, SF proposes the typical processes of Analysis, Design, Implementation, and Testing, but oriented toward the Problem and Solution Domain, thus encouraging reuse with emphasis on system architecture.

In order to develop any software product it is necessary to determine the technical, operational, and economic development feasibility. The difference lies in the fact that in the customized software development there is a single customer, whereas in the production of packed software, not only the diversity and scope of the needs of the potential customer, but also market size and location, marketing and commercialization channels have to be considered to estimate return on investment before any decision regarding development can be made [20].

In the case of customized software there are Maintenance strategies that are offered to the customer, whereas packed software includes a Support Process, which must be handled with more detail because it will not always be carried out by the same person. In some organizations there are departments dedicated exclusively to provide customer support or assistance, whereas software Maintenance is direct responsibility of the developers.

In addition, in the development of packed software a process subsequent to product development is to be taken in consideration: Delivery Process. In the customized software case, software delivery is personalized and the setting-up process can be performed by an employee of the company: In the packed software case, an additional component is needed for installing and uninstalling. This executable is neither included in the requirements nor depends on the system functions; however, it must be easy, simple, and fast to use, because the customer could not have the technical knowledge required to install or uninstall the purchased software.

Furthermore, in the case of the packed software production, testing phases have to be designed with the participation of volunteers who can subject the software to the rigor of use and offer the results of this experience. In the case of customized software, testing can be restricted to the ambit of the developer and the customer company [20].

5. ONTOLOGY FOR SF

According to authors, an SF must be flexible and capable of: producing various types of products, implementing software engineering concepts, and analyzing, projecting, implementing, developing, and improving systems. As it was already mentioned, the term SF is not new, but, most of the different versions of the SF concept agree that it comprises the mass production of a Family of Products (belonging to a determined Solution Domain) that comply with certain common requirements (belonging to a determined Problem Domain) that are established by a group of people involved [5]. These people are not necessarily the developers, but also costumers, users, providers, and even the software architects. Figure 1 represents the first

level of concepts based on the reviewed background.

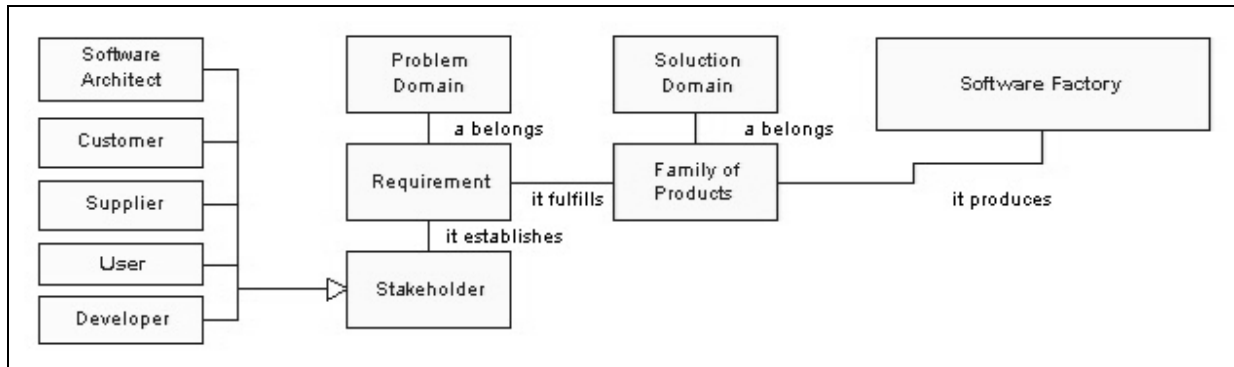


Figure 1. Elements involved in the SF definition

On the other hand, the term SF is defined as a ‘**Software Product Line**’ (SPL) that produces tools, processes, and content by using a **template** based on a given **schema**, with the purpose of automating the development and maintenance of variations of an architectonic product by means of adaptations, assembly, and configuration of components based on frameworks supported by an **environment** [8].

5.1 Software Product Line

A production line can be understood as a group of products that are closely related (by its functionality), that are sold to the same group of costumers, that are marketed through the same type of distribution, or fall within the same range of prices [6]. This concept has also its origin in the traditional industry, but is applicable to software development and its use is not very innovative. Perhaps the use of the SPL concept is known due to its high generality degree, being applicable to both large as well as small organizations.

Most organizations produce similar family of systems that differ from each other by certain characteristics, but broadly speaking, three general activities that can be applied in any situation have been detected [3]. For these authors, the three activities are closely related: progress in one of the activities necessarily implies the progress of the others, therefore, in fact, it cannot be said that there is an order among them, but due to space reasons, each one will be described individually and in a sequential way.

- **Core Assets Development:** This activity is focused on the production of general resources for any product. The scope of the product line, production plan, components, architecture, requirements, as well as restrictions, styles, patterns and frameworks, among others, are described here. The inventory of pre-existent assets is also configured, which is later named Object Library in SF.
- **Product Development:** This activity consists of assembling the assets developed or stored in the previous activity to create products according to market requirements; however, this process is rarely sequential. Product creation implies a continuous relationship with scope, production plan, and core assets.
- **Management:** Management plays a critical role in the success of the production line. The activities must have resources assigned, which should be coordinated and overseen. Management, at the technical as well as the organizational level, must be strongly associated to the effort of the SPL. However, management should not only be directed inwards, but should also consider the relationship with suppliers and consumers. This aspect is later called Supply Chain by SF.

On the other hand, Fabri and his colleagues [5], based on the model of SF presented by [1], propose a model that separates the production of core assets or components and the software production process; this author names the management process as "Organization of the work". This model comprises two significant activities: Software Production and Components Production.

The conception of SF by [8] is a little wider since it includes modeling of the application domain. SF is based

exactly on SPL, and shares activities with the model by [5], such as Analysis, Design and Implementation, but in the model by [8], these are oriented for a product line. The assets in the model of [8] are those that [5] denominates components. This last model also contemplates the possibility of selecting the tools which are more adequate for each domain, as well as the option of personalizing them.

It is worth noting the similarity between Fabri's SF model [5] and the SF template proposed by [8], since both models were published in similar dates but using different bibliographical sources, which reflects the current interest in improving the development processes to produce higher quality systems. Once the SPL concept and its role within the SF (Figure 2) are known, the rest of the elements which make it up can be defined: schema, template and environment.

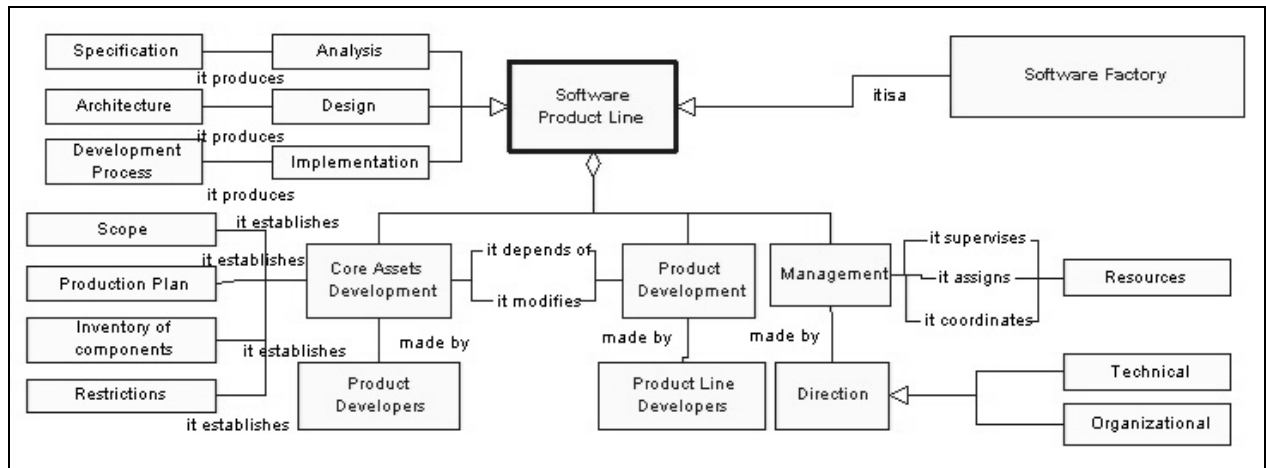


Figure 2. Concepts of Software Product Line

5.2 Schema, Template and Environment

An SF is an SPL that configures developing tools with guides and content packages that are carefully designed to build specific types of applications [7]. These same authors establish that an SF contains three key ideas: a schema, a template and a development environment.

- **A schema** can be thought of as a recipe. It is a list of ingredients such as previous projects, source code directories, SQL files, and configuration files. The schema establishes which *Domain Specific Languages* (DSL) should be used and describes how models based on those DSL can be transformed into codes or other artifacts. The schema describes the architecture of the product line and the relationships between the components and frameworks involved.
- **A template** is like a package or supermarket bag which contains all the ingredients listed in the schema. The template provides the patterns, languages, libraries, guides, examples, specific tools like DSL graphical publishers, scripts, style sheets, and other ingredients necessary to build the product.
- **A development environment** is like the kitchen where the meal is prepared. High level tools like *Microsoft Visual Studio Team System* offer a work environment suitable to build a family of products. These high-level tools facilitate the configuration, adaptation, and assembling of components.

Furthermore, this analogy can be exemplified by relating the products to plates served in a restaurant; and the stakeholders to the customers who order plates from the menu. A product specification is like an order of a plate with some extras [7].

Product developers are like the cooks that prepare the plates described in the orders and who can modify certain ingredients. The product line developers are the chefs that decide what is to appear in the menu, and the ingredients, processes and kitchen appliances that are to be used.

These three elements (schema, template and environment) are shown in Figure 3 and constitute the technological bases of an SF [8].

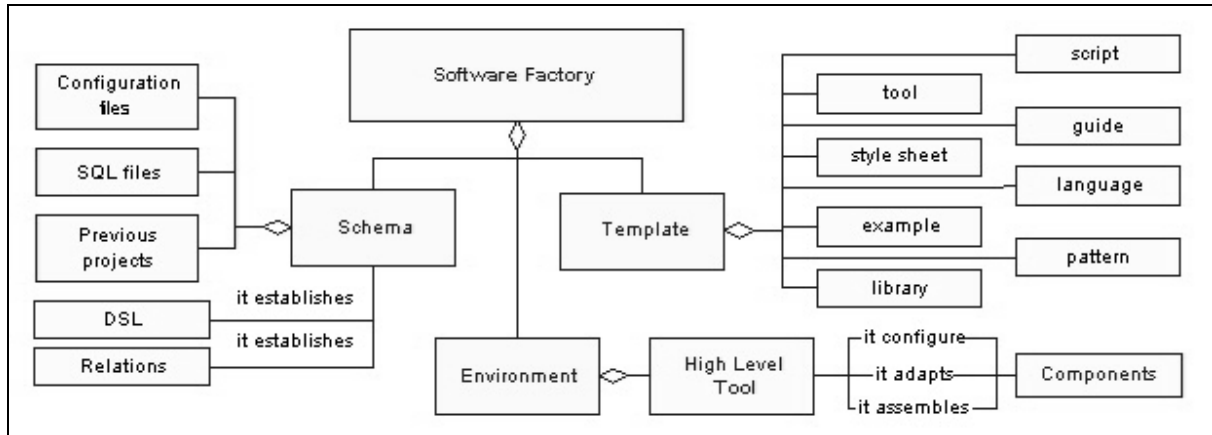


Figure 3. Technological elements that conform a SF

The term SF comprises not only technological aspects; next certain concepts of economy applied to the software development are defined, which also must be present in all company that decides to adopt this trend.

5.3 Economic Concepts

The concepts of **Marketing Channels** or **Supply Chain** are used by any organization that produces goods or provides services. The software industry does not escape from this reality but so far it has used this concept only to market the product already developed, but not as a part of the development process. A marketing channel takes the goods from the producers to the consumers, overcoming time gaps, market and possession [13]. An SF can be segmented both vertically as well as horizontally to delegate responsibilities to outside providers, thus creating Supply Chains for the development of a software product [8].

Another concept used by SF is **reuse economy**. Reusing solutions of a common sub-problem in a specific domain can reduce the total cost of solving multiple problems in the same domain [11] (cited in [8]). Furthermore, reuse can reduce the total time to market and improve the product quality. It is important mentioning the new connotation that is granted to this concept within an SF, where not only the reduction of time but also the cost of the project are taken into account, and even the return on the investment is improved, because when investing in architectural patterns, components and possible links in the supply chain, reuse makes it possible to see the financial results much faster, thus favoring the **scale and scope economies**.

These two concepts are frequently confused with each other. Although both reduce time and product costs, and improve quality producing multiple products instead of individual copies, there are considerable differences between them [8]. Scale economy offers the possibility of developing multiple identical instances starting from a simple design. On the other hand, scope economy emerges when multiple but similar designs and prototypes are produced collectively more than individually.

The main difference between scale and scope economy, in terms of SF, lies in the moment at which reuse appears. In the first type of economy, reuse appears after the product has been developed, whereas in the scope economy, reuse appears while the product is being developed. The relation between these concepts can be seen in Figure 4.

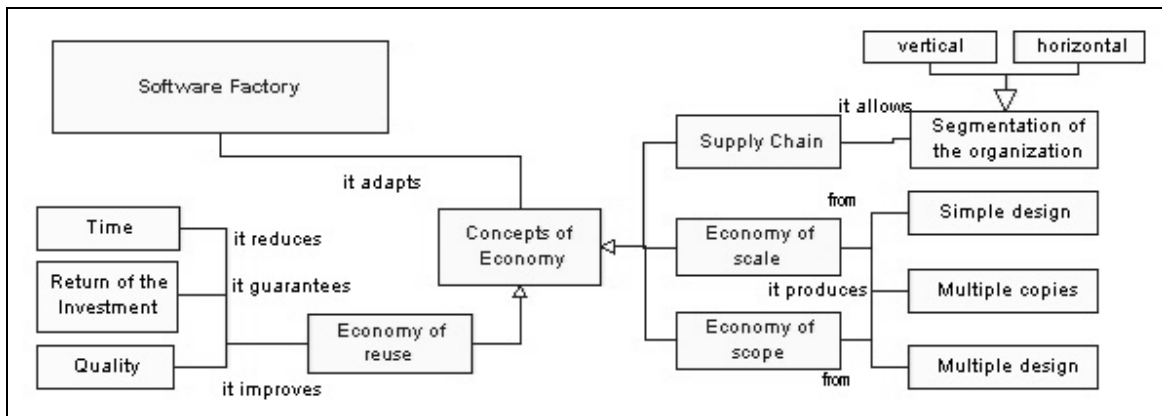


Figure 4. Economic Concepts of an SF

Both scale economy and scope economy, according to the SF conception, increase efficiency in the software development process because they are not limited to the development of one product at a time, but promote mass development, strongly supported by the concept of SPL. By improving efficiency, quality in the development process is therefore improved. Developing an SF implies the best practices of Software Engineering to be systematically applied [5]. Following, some of the best practices that this trend offers are shown.

5.4 Best Practices

SF is based on the convergence of key ideas in **Model-Drive Development (MDD)** and systematic reuse [8]. According to these authors, MDD reduces the problem of abstraction by using models only as documentation. This is one of the main practices proposed by SF because, according to these authors, models must be used for developing instead of documenting the product.

All these efforts are meant to go from the traditional way, based on a source code, to a new way of developing software products based on models, where previous attempts are increasingly reused. According to Pressman [19], the **Component-based Software Engineering (CBSE)** aims at finding a set of pre-built and standardized software components that fit in a specific architectural style for an application domain. This set of components constitutes the Core Assets Development of SPL.

On the other hand, a framework is a group of classes for a specific kind of software; it can be then inferred that a process framework is a group of sub-processes for a determined type of domain [11]. In addition, a **process framework** is a structure that organizes the micro-processes used to build artifacts that form part of the members of a family of products [7]. Developing a process framework for a line of software products makes sense because the cost can definitely be recovered with the production of many products.

These same authors affirm that a process framework essentially disaggregates a process into micro-processes attached to the development of various types of artifacts, including the description of the requirements necessary for such a goal. In these descriptions one can list considerations as diverse as key decision points; analysis of trade-offs associated to each point of decision, optional or required activities, and the results to be produced in each activity.

In this sense, according to [14], up to now the system architecture remained always in a sort of black box, being a secret known only by its authors. A good software architecture promotes the **reuse**, and makes it possible to obtain and maintain the intellectual control of the development as well as to manage its complexity and maintain system integrity, implicitly managing quality in the development. Figure 5 presents the involved concepts of the SF best practices.

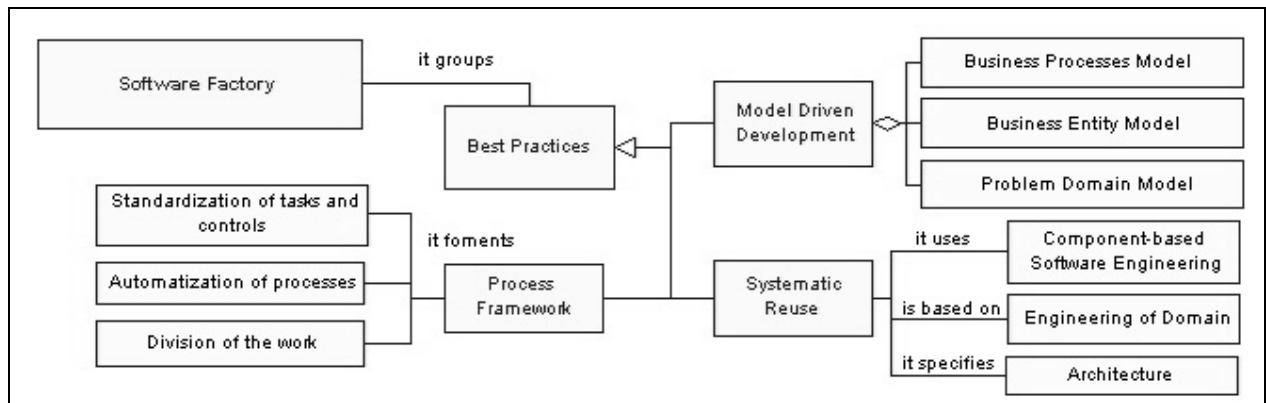


Figure 5. Best Practices in Software Development

5.5 Shared-concepts Model

Once described separately the aspects shared around SF, it is worth showing them in a unified model where new relationships, not shown previously, are presented. One of the main relationships is based on the fact that the economic aspects adapted by SF to software development are supported by the technological aspects; accordingly, the **economy of reuse** promotes the **systematic reuse**; the scale and **scope economies** favor the **Model Driven-Development**; and the **Supply Chain** must be coordinated by the **Management** of the SPL.

On the other hand, by presenting all models in a unified way, those recurrent concepts which serve as integrators between the different parts involved are outlined. In this way, it can be established that in an SF, the **Software Product Line** develops a group of products with similar characteristics thus conforming a **Product Family**. In order to develop these products a certain **schema** is followed that describes the system **architecture**, and also establishes the relationships between the **frameworks** used and the components to assemble. These **components** are registered in the **inventory of components** thus fomenting **systematic reuse**.

As it has been shown so far, the term SF, besides having multiple definitions, gathers a considerable number of concepts which should be identified to handle a general idea of its scope. Figure 6 shows a unified version of the proposed conceptual model, including the new relationships described before. The boxes do not imply any kind of order or hierarchy between concepts; they are used only to group similar concepts and to facilitate the reading of the model.

Although SF does not contribute new concepts to the software product development, the innovation consists of the grouping of previous concepts (SPL, MDD, systematic reuse and framework of processes), presented together, in a new attempt to go from the traditional to the automated production.

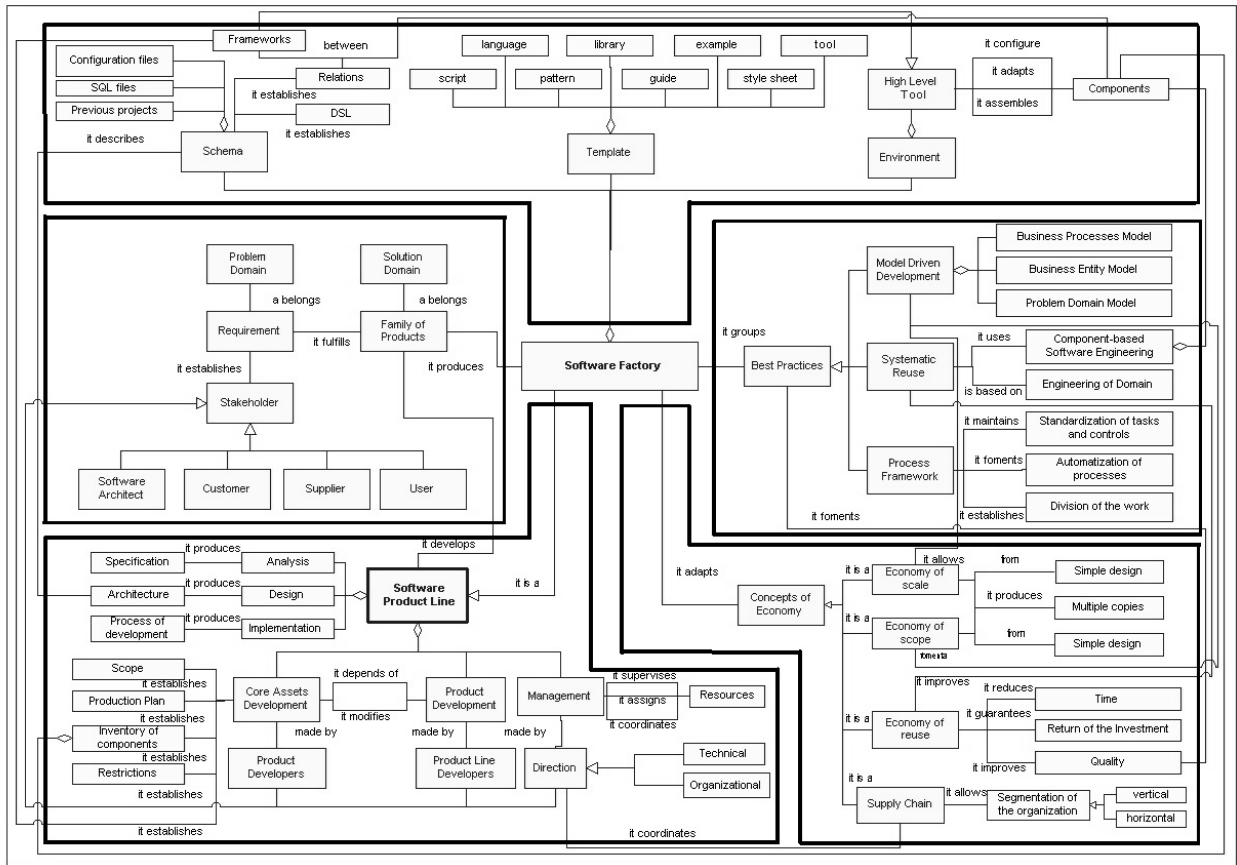


Figure 6. Conceptual model proposed for a Software Factory

6. PROCESSES IDENTIFIED IN AN SF vs. ISO 1504 PROCESSES

Many of the concepts associated to SF imply the execution of processes as well; this is the case of "analysis", "design" and "implementation" of an SPL. Therefore, it is necessary to group the concepts shown in Figure 6 in typical processes of an SF. Table 1 shows this relationship between processes of an SF and the aspects of the conceptual model.

Processes identified in an SF		Concepts in the Conceptual Model
Product Development	Analysis	Problem Domain, Solution Domain, Domain Engineering, Requirements, SPL Analysis
	Design	Model-Driven Development, Product Family, SPL Design, Architecture, Economy of Scope and Economy of Scale.
	Implementation	Systematic Reuse, Economy of Reuse, Core Asset Development, Product Development, Components and development of the template.
	Testing	Restrictions and schema of the SF
Support	Delivery and Distribution	Time of delivery, Supply Chain
	Consulting	Relations with stakeholders
Marketing	Collateral Development	Production Plan, Tools of high level
	Sales	Return on the Investment, Economy of Reuse, Supply Chain
Management	Supervision and Evaluation	Scope, Process Framework, Technical and Organizational Management, Development of the Environment, Economy of Reuse, relationships with stakeholders and the activities of Management.
	Quality assurance	Best Practices , standardization of tasks and controls

Table 1. Relationship among the unified processes and the own concepts of an SF

It is worth mentioning that the concepts included in the Conceptual Model can be used by several SF processes; this is the case of Supply Chain that is within the Delivery and Distribution Process, but also within the Sales Process. Other example is the Economy of Reuse concept that is used in the Implementation

Process and also in the Management Process, since the reuse can be expressed as a code and components or as processes and controls.

The grouping shown in Table 1 was made matching the description of the concepts. Furthermore, the denominations of the unified processes also are related with the packed software processes [4]. Below in Table 2 the relationship between the packed software processes is shown.

Steps of the production process of packaged software [4]	SF unified processes
Development	Product Development
Testing	Testing, within Product Development
Disk duplication, Packing and Manuals	Collateral development within Marketing
Sales and Marketing	Marketing
Distribution	Delivery and Distribution, within Support
Support	Support

Table 2. Relationship among the SF unified processes and the steps of the Production process of packed software

It is worth emphasizing that the Management process (non included in the steps of the Production process of the packed software) was added to the SF unified processes, because it is one of the concepts involved in all SF. With the aim of going more in detail, Table 3 shows the activities of each one of the typical processes in SF according to [5] and [8].

Process	Sub-process	Activities	
Product Development	Analysis	SPL Analysis, Scope of the Solution Domain, Elicitation of requirements.	Scope of the Problem Domain, Business Case Analysis, Evaluation of the scope.
	Design	SPL Design, Product Design, Development of the Architecture	Definition of Processes of product development , Modeling Functionalities
	Implementation	Specification of the product, assemble, review, store y distribute components, automate the development process	Configuring the software, Development of executables, Provisioning of assets, Specification of the environment
	Testing	Design and development of the test and corrections assets, evaluate the product, compare results	Execution of tests of software and components, Acceptance tests
Support	Delivery and Distribution	Pack and release the product, Development of installers	Mechanisms of systematic update, Distribution of components
	Consulting	Levels of services, repair and support within the guarantees	Customer support
Marketing	Collateral Development	Generation of Manuals, Promotions and publicity	Planning offers of products, Documentation
	Sales	Customer Relationship, partners and suppliers, To Calculate Profitability.	Defining channels and policies of commercialization
Management	Supervision and Evaluation	To use historical data, To maintain communication with the members of the process, To define the equipment and the division of the work	Verification of deadlines, Establishing terms and conditions, Managing each process. Establishing tools, resources and processes.
	Quality assurance	To improve the services, To optimize the production	Quality Management

Table 3. Specification of the activities contained in the processes identified in an SF

On the other hand, the Systemic Quality of an organization is the reflection of the quality of its software products and the quality of their development process (design quality and manufacturing) [16]. Since SF establishes changes in the development process rather than in the product to be developed, the comparison will take into account the ISO 15504 standard.

The ISO 15504 standard consists of 3 levels: Categories, Processes, and Base Practices. In Table 4 part of the structure of this standard is shown, without the base practices.

Categories	Process	
Customer-Supplier (CUS)	CUS 1. Acquisition CUS 2. Supply	CUS 3. Requirements Elicitation CUS 4. Operation
Engineering (ENG)	ENG 1. Development	ENG 2. System and software maintenance
Support (SUP)	SUP 1. Documentation SUP 2. Configuration management SUP 3. Quality assurance SUP 4. Verification	SUP 5. Validation SUP 6. Joint review SUP 7. Audit SUP 8. Problem resolution
Management (MAN)	MAN 1. Management MAN 2. Project management	MAN 3. Quality Management MAN 4. Risk Management
Organization (ORG)	ORG 1. Organizational alignment ORG 2. Improvement process ORG 3. Human resource management	ORG 4. Infrastructure ORG 5. Measurement ORG 6. Reuse

Table 4. Categories and Processes of ISO 15504 standards

Entering a lower level, some processes in Table 4 possess sub-processes. It is convenient to show these sub-processes, because they will be the main point of comparison between the ISO 15504 standard and the SF. Table 5 shows the sub-processes of ISO 15504.

Process	Sub-process	
CUS.1. Acquisition	CUS.1.1. Acquisition preparation, CUS.1.2. Supplier selection	CUS.1.3. Supplier Monitoring, CUS.1.4. Customer Acceptance
CUS.4. Operation	CUS.4.1. Operational use	CUS.4.2. Customer support
ENG.1. Development	ENG.1.1. System requirements analysis and design ENG.1.2. Software requirements analysis ENG.1.3. Software design	ENG.1.4. Software construction ENG.1.5. Software integration ENG.1.6. Software testing ENG.1.7. System integration and testing
ORG.2. Improvement process	ORG.2.1. Process establishment ORG.2.2. Process assessment	ORG.2.3. Process improvement

Table 5. Processes and sub-processes of the ISO 15504 standard

Each process or sub-process has associated a set of Base Practices which can be reviewed in ISO 15504 (1999). Based on the specifications of these Processes and their associated Base Practices, and, comparing them with the sub-processes or activities of the unified processes shown in Table 3, it can be observed that many of these sub-processes are already considered in the current version of the ISO 15504 standard. Specifically for 16 of those 33 sub-processes, the ISO 15504 standard has incorporated Base Practices with similar objectives.

Table 6 below shows the correlation among the processes of an SF and the sub processes (characteristics) of the current version of the ISO 15504 standard. The new sub-characteristics or metrics to be developed are also specified.

According to the legend, the activities or sub-processes marked as (NBP) imply new metrics to be included into the ISO 15504 standard. Similarly, those marked as (NSP) imply new sub-process within the current structure of the model.

Process	Sub-process	Activities
Product Development	Analysis (covered by <i>ENG.1.1.</i> and <i>ENG.1.2.</i>)	- SPL Analysis (NBP) - Scope of the Solution Domain (NBP) - Elicitation of requirements (covered by CUS3) - Scope of the Problem Domain (NBP) - Business Case Analysis (NBP) - Evaluation of the scope (covered by ENG1)
	Design (covered by <i>ENG.1.3.</i>)	- SPL Design (NBP) - Product Design (covered by ENG1.3) - Development of the Architecture (covered by ENG1.3)

		<ul style="list-style-type: none"> - Definition of Processes of development of the product (<i>covered by ENG1</i>) - Modeling Functionalities (<i>NBP</i>)
	Implementation (<i>covered by ENG.1.4.</i>)	<ul style="list-style-type: none"> - Specification of the product (<i>NBP</i>) - Assemble, review, store y distribute components (<i>covered by ORG9</i>) - Automate the development process (<i>NBP</i>) - Configuring the software (<i>covered by ORG9</i>) - Development of executables (<i>covered by ORG9</i>) - Provisioning of assets (<i>NBP</i>) - Specification of the environment (<i>covered by ORG7</i>)
	Testing (<i>covered by ENG.1.6 and ENG.1.7</i>)	<ul style="list-style-type: none"> - Design and development of the test and corrections assets (<i>covered by SUP4</i>) - Evaluate the product (<i>covered by SUP5</i>) - Compare results (<i>NBP</i>) - Execution of tests of software and components (<i>covered by SUP4</i>) - Acceptance tests (<i>covered by SUP6</i>)
Support	Delivery and Distribution (<i>NSP within CUS2</i>)	<ul style="list-style-type: none"> - Pack and release the product. (<i>NBP</i>) - Development of installers (<i>NBP</i>) - Mechanisms of systematic update (<i>NBP</i>) - Distribution of components (<i>NBP</i>)
	Consulting (<i>covered by ENG2</i>)	<ul style="list-style-type: none"> - Levels of services, repair and support within the guarantees (<i>covered by ENG2</i>) - Customer support (<i>covered by SUP6 and CUS4</i>)
Marketing	Collateral Development (<i>NSP within CUS2</i>)	<ul style="list-style-type: none"> - Generation of Manuals (<i>NBP</i>) - Documentation (<i>covered by SUP1</i>) - Promotions and publicity (<i>NBP</i>) - Planning offers of products (<i>NBP</i>)
	Sales (<i>NSP within CUS2</i>)	<ul style="list-style-type: none"> - Customer Relationship, partners and suppliers (<i>covered by CUS1</i>) - Defining channels and policies of commercialization (<i>NBP</i>) - Calculating Profitability (<i>NBP</i>)
Management	Supervision and Evaluation (<i>covered by ORG8</i>)	<ul style="list-style-type: none"> - Using historical data (<i>NBP</i>) - Defining the equipment and the division of the work (<i>covered by ORG3</i>) - Maintaining communication with the members of the process (<i>covered by MAN2</i>) - Verification of deadlines (<i>NBP</i>) - Establishing terms and conditions (<i>covered by CUS2</i>) - Establishing tools, resources and processes (<i>covered by MAN1</i>) - Managing each process (<i>covered by MAN2</i>)
	Quality assurance (<i>covered by SUP3</i>)	<ul style="list-style-type: none"> - Improving the services (<i>NBP</i>) - Optimizing the production (<i>NBP</i>) - Quality Management (<i>covered by MAN3</i>)
Legend: NBP= New Best Practice NSP= New Sub Process		

Table 6. Correlation of the Processes identified in an SF with the Processes and Base Practices of ISO 15504

It can be observed that there are activities typical of an SF that were already contemplated by ISO 15504, but there are also some others that result new for this model, mainly those related to Marketing Processes. Below in Table 7 the percentage of SF activities that are covered by the ISO 15504 standard are presented.

Process	Sub Process	Number of activities	Number of activities covered by ISO 15504	Percentage (%)
Product Development	Analysis	6	2	33.33
	Design	5	3	60
	Implementation	7	4	57.14
	Testing	5	4	80
Support	Delivery and Distribution	4	0	0
	Consulting	2	2	100
Marketing	Collateral Development	4	1	25
	Sales	3	1	33.33
Management	Supervising and Evaluation	7	5	71.42
	Quality assurance	3	1	33.33

Table 7. Percentage of the activities typical of an SF covered by ISO 15504

The processes that present a higher coverage for the ISO 15504 characteristics, in decreasing order, are: Consulting Process (100%), Testing Process (80%), Design Process (60%), Supervising and Evaluating

Process (71.42%), and Implementation Process (57.14%). However, there are processes that have a very low percentage of coverage, as in the case of the Delivery and Distribution Process (0%), which reflects that in the ISO 15504 standard the fact of developing a system for a specific customer (customized software) is considered, but not in the case of multiple costumers (packed software). On the contrary, the Design Process, in spite of having 80% coverage, does not contemplate one of the main characteristics of an SF, where models must be used as inputs for the implementation and not as simple documentation.

Furthermore, the fact that some processes of the SF are considered by the ISO 15504 standard does not imply that they are completely evaluated. There can be ISO 15504 base practices that cover a large part of the activities of some process of the SF, but that do not include specific details. For example, the activity "Defining the equipment and division of the work" is covered by process "ORG.3. – Human resource management" in the ISO 15504 standard, but its Base Practices are oriented to hire personnel, but not contemplate the possibility that the developer could be hired by other company.

It is also observed that the processes of "Delivery and Distribution", "Collateral Development" and "Sales" are not considered in the present version of the ISO 15504 standard and can be added like new sub-processes within the Supply Process (CUS2). All these set of comparisons become an opportunity to improve the ISO 15504 standard.

8. CONCLUSIONS AND FUTURE WORKS

SF implies not only technological but also economic changes, therefore, any company that decides adopting an SF as its basis to produce software should consider certain economic concepts that were not involved before in this production sector . This paper organizes the main knowledge around the Software Factory concept, using an ontological approach.

As a result of the first statement, the reuse and the development of several products in parallel must be the fundamental characteristics in a company that decides to make the transition towards a Software Factory. However, it is relevant to outline that a company that covers only a part of the identified characteristics (standardization, reuse, MDD, etc.) could not be considered an SF; in this sense all the concepts described in our ontology should be covered to achieve this category.

In this paper, a relationship between SF processes and the ISO 15504 standard is also presented, determining their presence (by percentage) or absence respect to the standard base practices. As a result we obtained that the ISO 15504 standard has Base Practices incorporated which cover less than fifty percent of the SF sub-processes. For those practices proposed for SF that are not considered by the standard we have proposed a way to include them without affecting the conceptual structure.

This research reflected that none SF model or concept offered a way for making the development process operative, and that is the reason for adopting the ISO 15504 standard. It is expected that this work could serve as the basis for the development of an instrument that allows considering systemic quality in an organization with SF characteristics. In addition, in spite of having found several models of SF, no ontologies were related to this subject; therefore this research offers the first attempt in the development of an ontology for SF.

In future works, the use of processes of software like Unified Process (RUP), Catalysis, XP and others, within SF will be considered, but this point goes beyond the scope of this research, where the purpose was to specify the systemic quality in SF following the ISO 15504 standard.

ACKNOWLEDGEMENTS

This research was financed by the Fondo Nacional de Ciencia, Tecnología e Innovación (FONACIT) of the Bolivarian Republic of Venezuela, through the project S1-2000000437.

REFERENCES

- [1] V. Basili, G. Caldiera y G. Cantone “A Reference Architecture for the Component Factory”. ACM Transaction on Software Engineering and Methodology. Vol 1. n° 1. pp 53-80. January, 1992.
- [2] R. Baskerville. “Investigating Information Systems with Action Research”. Association for Information Systems. 1999.
- [3] P. Clements y L. Northrop. “Software Product Lines. Practices and Patterns”. SEI Series in Software Engineering. Addison Wesley. Second Edition. Boston, USA. pp. 29-31. 2001.
- [4] N. Coe. “Emulating the Celtic tiger? A Comparison of the Software Industries of Singapore and Ireland”. Singapore Journal of Tropical Geography, 20(1). 1999.
- [5] J. Fabri, A. Presende, L. Begosso, A. L’Erário, F. Fujii y M. de Paula. “Techniques for the Development of a Software Factory: Case CEPEIN-FEMA”. 17th International Conference Software and Systems Engineering and their Applications. ICSSEA 2004. Paris, November 30 – December 3. 2004.
- [6] F. García, J. Barras, M. Laguna y J. Marqués. “Líneas de Productos, Componentes, Frameworks y Mecanos”. Informe Técnico, Departamento de Informática. Universidad de Salamanca. 2002.
- [7] J. Greenfield, y K. Short. “Moving to Software Factories”. White Paper, updated on July 15, 2004, taken on January 2005 at <http://www.softwarefactories.com/ScreenShots/MS-WP-04.pdf>
- [8] J. Greenfield, K. Short, S. Cook, y S. Kent. “Software Factories. Assembling Applications with Patterns, Models Frameworks and Tools”. Wiley. First Edition. 2004.
- [9] Humphrey, W. “Introduction to the Personal Software Process”. Addison Wesley Longman, Inc., Cambridge, Massachusetts, 1997.
- [10] International Organization for Standardization (2004) Software Process Assessment, TR 15504, on-line, WG 10: Software Process Assessment, at <http://www.sqi.gu.edu.au/spice/>
- [11] I. Jacobson, M. Griss y P. Jonsson. “Software Reuse: Architecture, Process and Organization for Business Success”. ACM Press, 1997.
- [12] B. Kitchenham. “Evaluating Software Engineering Methods and Tool. Part 1: The evaluation context and Evaluation Methods”. Software Engineering Notes, 21(1). 1996.
- [13] P. Kotler. “Dirección de Marketing”. La edición del milenio. Prentice Hall. 2001.
- [14] P. Kruchten. “The Rational Unified Process. An Introduction”. Third Edition. Addison Wesley Longman, Inc. 2003.
- [15] N. Lim, S. James, y F. Pavri. ”Diffusing Software-based Technologies with a Software Factory Approach for Software Development. A Theoretical Framework”. Proceeding of the 22nd Information Systems Research Seminar in Scandinavia (IRIS22): Enterprise Architectures for Virtual Organisations. 7-10 August, Keuruu, Finland. 1999.
- [16] L. Mendoza, M. Pérez, and A. Grimán. “Prototipo de Modelo Sistémico de Calidad (MOSCA) del Software”. Computación y Sistemas. Vol. 8, pp. Pp. 196 - 217. 2005
- [17] Microsoft Prensa. “Microsoft aumenta su ecosistema de socios alrededor de Visual Studio 2005 Team System”. Tomado en Febrero de 2004 de <http://www.microsoft.com/latam/prensa/2004/Octubre/VisualStudio.asp>
- [18] OXE Open Source Software Factory, “Open Experience Environment”. Taken in Julio from 2005 of <http://php.cin.ufpe.br/~oxe/>

- [19] R. Pressman. "Ingeniería del Software. Un enfoque práctico". Quinta Edición. Mc Graw Hill. 2002.
- [20] D. Rojas. "Aproximación a la Industria del Software en el Estado Lara". Universidad Centroccidental Lisandro Alvarado, Decanato de Ciencias y Tecnología. Barquisimeto, Venezuela. 2003.
- [21] SEI - Software Engineering Institute. "ISO/IEC 15504 - An Emerging Standard on Software Process Assessment". 1998