



UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TESIS

**APROXIMACIÓN DE LA AUTOMATIZACIÓN DE LA SÍNTESIS DE
MODELOS CARACTERIZADOS EN LÍNEAS DE PRODUCTO**

PRESENTA

Catalina Calderón García

**PARA OBTENER EL GRADO DE MAESTRÍA EN CIENCIAS CON
OPCIÓN A LA COMPUTACIÓN**

TUTOR

Dr. Ángel Eduardo Muñoz Zavala

COMITÉ TUTORAL

Dr. Francisco Javier Álvarez Rodríguez

Dra. Eunice Esther Ponce de León Sentí

Aguascalientes, Ags., a 13 de mayo de 2015



UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES
CENTRO DE CIENCIAS BÁSICAS

CATALINA CALDERON GARCÍA
ALUMNA DE LA MAESTRIA EN CIENCIAS
CON OPCIÓN A LA COMPUTACIÓN,
P R E S E N T E.

Estimada alumna Calderón:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o caso práctico titulado: **“APROXIMACIÓN DE LA AUTOMATIZACIÓN DE LA SINTESIS DE MODELOS CARACTERIZADOS EN LINEAS DE PRODUCTO”**, hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 23 de marzo de 2015
“SE LUMEN PROFERRE”
EL DECANO

M. en C. JOSÉ DE JESÚS RUIZ GALLEGOS

c.e.p.- Archivo.
JJRG,yscd





UNIVERSIDAD AUTONOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruíz Gallegos
DECANO DEL CENTRO DE CIENCIAS BASICAS

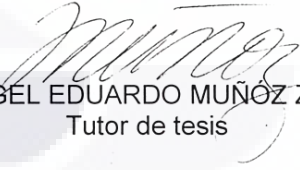
PRESENTE

Por medio del presente como Tutor designado de la estudiante **CATALINA CALDERÓN GARCÍA** con ID **159787** quien realizó la Tesis titulada: **APROXIMACIÓN DE LA AUTOMATIZACIÓN DE LA SÍNTESIS DE MODELOS CARACTERIZADOS EN LÍNEAS DE PRODUCTO**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que ella pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE
"Se Lumen Proferre"

Aguascalientes, Ags., a 20 de marzo del 2015


DR. ANGÉL EDUARDO MUÑOZ ZAVALA
Tutor de tesis



c.c.p. - Interesado
c.c.p. - Secretaría de Investigación y Posgrado
c.c.p. - Jefatura del Depto. de Ciencias Computacionales
c.c.p. - Consejero Académico
c.c.p. - Minuta Secretario Técnico



UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruíz Gallegos
DECANO DEL CENTRO DE CIENCIAS BÁSICAS

PRESENTE

Por medio del presente como Asesor designado de la estudiante **CATALINA CALDERÓN GARCÍA** con ID **159787** quien realizó la Tesis titulada: **APROXIMACIÓN DE LA AUTOMATIZACIÓN DE LA SÍNTESIS DE MODELOS CARACTERIZADOS EN LÍNEAS DE PRODUCTO**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que ella pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

ATENTAMENTE

"Se Lumen Proferre"

Aguascalientes, Ags., a 20 de marzo del 2015

DR. FRANCISCO JAVIER ALVAREZ RODRIGUEZ
Asesor de tesis

c.c.p. - Interesado
c.c.p. - Secretaría de Investigación y Posgrado
c.c.p. - Jefatura del Depto. de Ciencias Computacionales
c.c.p. - Consejero Académico
c.c.p. - Minuta Secretario Técnico





UNIVERSIDAD AUTÓNOMA
DE AGUASCALIENTES

FORMATO DE CARTA DE VOTO APROBATORIO

M. en C. José de Jesús Ruíz Gallegos
DECANO DEL CENTRO DE CIENCIAS BÁSICAS

P R E S E N T E

Por medio del presente como Asesor designado de la estudiante **CATALINA CALDERÓN GARCÍA** con ID **159787** quien realizó la Tesis titulada: **APROXIMACIÓN DE LA AUTOMATIZACIÓN DE LA SÍNTESIS DE MODELOS CARACTERIZADOS EN LÍNEAS DE PRODUCTO**, y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia, me permito emitir el **VOTO APROBATORIO**, para que ella pueda proceder a imprimirlo, y así como continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E

"Se Lumen Proferre"

Aguascalientes, Ags., a 20 de marzo del 2015

DRA. EUNICE ESTHER PONCE DE LEÓN SENTI
Asesor de tesis



- c.c.p. - Interesado
- c.c.p. - Secretaría de Investigación y Posgrado
- c.c.p. - Jefatura del Depto. de Ciencias Computacionales
- c.c.p. - Consejero Académico
- c.c.p. - Minuta Secretario Técnico

AGRADECIMIENTOS / ACKNOWLEDGEMENTS

I want to say thanks to my tutors who encourage me to start this research work which was a dream for a long time. Thanks for your patience and support during the difficult moments.

A special recognition goes to the CONACYT for the support provided to fund this research work. Thanks to Mexican community who by paying taxes supports the labor of CONACYT in our country. Hoping that, this research will encourage others to create new ways of thinking on Software Engineering to propel technology and jobs in México.

Finally, thanks to my family, what can I say? All of you are the best of my life. Thanks for your unconditional support ever. I love you guys.

DEDICATORIAS / DEDICATORIES

To my father who was my first teacher, and to, his dream to be a researcher...

To my mum and her dream to turn me into an educated woman...

To my family...

Para mí familia...



INDICE GENERAL / TABLE OF CONTENTS

INDICE DE TABLAS / TABLE LIST	2
INDICE DE FIGURAS / TABLE OF FIGURES	3
ABBREVIATIONS	4
RESUMÉN	5
ABSTRACT	6
Chapter 1. Introduction	7
1.1 Contribution and scope of this research work.....	7
1.2 Motivation and Context.....	9
1.3 Organization of this thesis.....	10
Chapter 2. Background	11
2.1 The SEI framework for Software Product Line	11
2.2 Mobile User Interfaces Families	16
2.3 Automated treatment of feature models	18
2.4 Evolutionary Computation	19
2.5 Software Automation and SPLE	30
Chapter 4. Study case: Configuration of a SPL for user interfaces at mobile devices domain	34
4.1 The study case	34
4.2 The MAPDS prototype	36
Chapter 5. Results and Evaluation	58
5.1 Evaluation	58
CONCLUSIONS	64
GLOSARY	65
REFERENCES	66
ANNEXS	70

INDICE DE TABLAS / TABLE LIST

Table 1. Car Configurations..... 16

Table 2. Feature set for Lyker scale question36

Table 3. Set of Android elements most common used to build MUIs extracted from
Android Community [7]40

Table 4. Results from experiment 159

Table 5. Results from experiment 2.....60



INDICE DE FIGURAS / TABLE OF FIGURES

Figure 1. Reuse History: From Ad-Hoc to Systematic [5]12

Figure 2. The three essential activities for software product line [5]13

Figure 3. A simple feature diagram [6].....15

Figure 4. A minimal *mobile user interface* in Android.16

Figure 5. A basic MUI and its feature diagram18

Figure 6. Shows the CGA workflow22

Figure 7. Encoding of a feature model, approach introduced by ETHOM [1].....24

Figure 8. Chromosome called SurveyComponent125

Figure 9. Encoding representation of natural number into a binary array27

Figure 10. Example of one-point crossover introduced by ETHOM [1].....28

Figure 11. Relations (or- and alternative) with a single child repairing [1]29

Figure 12. Cross-tree constraints between features with parental relationship repairing [1]29

Figure 13. Contradictory or redundant cross-tree constraints repairing [1]29

Figure 16. XML code and its MUI37

Figure 17. The class interaction on MAPDS tool38

Figure 18. MAPDS *mobile user interface*.39

Figure 19. Relation between the ETHOM chromosome and the MUI code based on XML41

Figure 14. Feature model of SuervyComponent145

Figure 15. MUI feature model proposed on this research work and its translation to ETHOM [1] chromosome46

Figure 20. A FM Individual and its derived products.....56

Figure 21. Sample of MUIs with similar fitness and different XML code.....63

ABBREVIATIONS

ADT – Android Development Tools

AC - Automation Concept.

AI - Artificial Intelligence.

AIT - Artificial Intelligence Techniques.

EA - Evolutionary Algorithm.

Framework 5.0 - Framework for Software Product Line Practice, Version 5.0.

MAPDS - Mobile Android Product Derivation Software

SE - Software Engineering.

SPL – Software Product Line.

SPLE - Software Product Line Engineering.

RESUMÉN

El desarrollo de software se enfrenta al avance de la producción en masa derivada del crecimiento de las sociedades de conocimiento que demandan productos tecnológicos de mayor calidad y menor costo. De este modo, el paradigma de la Línea de Producto (SPL) se está convirtiendo en el campo de batalla para los desarrolladores de hoy y el futuro, dado su enfoque de desarrollo basado en la reutilización de un campo específico de dominio.

Teniendo en cuenta que la producción en masa se apoya en la estandarización y automatización de procesos de fabricación pensar en el desarrollo de software en escenarios automatizados bajo estas condiciones requiere un esfuerzo obligado y no ha sido suficientemente explorado hasta ahora.

En este sentido, este trabajo de investigación contribuye a aproximar el concepto de automatización en Líneas de Producto de Software mediante la creación de una línea de Producto de Software (SPL) para interfaces de usuario móviles en el contexto de la utilización de técnicas de inteligencia artificial (AIT) como un estándar de automatización. Probándolo mediante un caso de estudio que utiliza la síntesis automatizada de modelos de característica basada en algoritmos evolutivos para lograrlo.

ABSTRACT

The software development is facing the advancement of mass production produced by growing of knowledge societies who demand technological commodities of higher quality and lower cost. Under this scenario, the paradigm of Software Product Line (SPL) is becoming the battleground for developers today and future due its reuse development approach based on a domain specific field that reduces the time and effort required to produce software products.

Lessons learned from other industries teach us, the mass production is effectively supported in the standardization and automation of manufacturing methods, so that, the thought of automated scenarios of software development is an effort required and not enough explored so far.

This research work contributes to approximate the concept of automation in the Software Product Line Engineering by creating a mechanism for automatic derivation of *Mobile User Interfaces* (MUI) under the context of adding Artificial Intelligence Techniques (AIT).

Although, the automated software development can be done by ad-hoc mechanisms, we do believe that by using AIT, the growth of automated-tools for software development may see itself benefited by the expansion of a scientific formal area that has provided successful automated approaches for many knowledge areas.

Chapter 1. Introduction

This chapter introduces to the reader a brief description about the main topic addressed in this research work about using Artificial Intelligence Techniques (AIT) for automation purposes on Software Product Line (SPL), the use of automated tools supported at Artificial Intelligence Techniques to approximate an *automation concept* on this key paradigm and how the motivation behind this research and the scope are successfully addressed at the end of the document.

1.1 Contribution and scope of this research work

This thesis promotes the use of AIT for *automatic product derivation* in Software Product Line (SPL). The model proposed is taking advantages of the automatic synthesis of feature models based on Evolutionary Computation (EC) [1] to achieve automatic product derivation of *mobile user interfaces* for applications in mobile computing.

The Software Product Line (SPL) is a paradigm for developing software systems based on components. A Software Product Line (SPL) differs from others software development paradigms as is sustained by [2] on the fundamental distinction of “*development for reuse*” and “*development with reuse*”.

At the SPL approach, requirements are obtained from a target domain and not from a specific problem, what introduced a novel approach where development is focusing to create assets. Samples of assets are software components, documentation, and others goods that can be part of a software product configuration as interchangeable modules.

The SPL approach basically introduces a change of strategy for building software from the strategic business meeting and the ad-hoc contract to developing based on a business domain with re-use purposes [2]. The business domain will lead the development efforts, so that, the assets created can be reused in many different products that satisfy requirements in same business domain. [2].

Problem Statement

Research in the AIT application to Software Engineering (SE) has grown enormously in the last two decades to produce a large number of projects and publications. However, most of the work carried out in this area is mainly targeted to the research community, what is producing a gap between research and practice that depicts an open problem [3].

Under this premise, our work adds to further exploration, where both disciplines, artificial intelligence and software engineering can synergize in order to contribute to close this gap. By taking advantages of the automatic synthesis of feature models based on EC [1] this research work introduces an approach for automatic product derivation in a real scenario. A study case based on automatic derivation of *mobile user interfaces* in *mobile devices* with Android operating system that serves to prove in terms of SPL practice, automatic product derivation and the efficiency of automation based on AIT in real practice also, in order to tackle these open problems:

- 1 - Provide an approximation to close the gap between research and practice of AIT applied to Software Engineering. [3]
- 2 - Providing an implementation for ETHOM [1] algorithm designed for optimizing computationally hard feature models using evolutionary algorithms.
- 3 – A contribution for the approximation of a formal *Automation Concept* (AC) in SPL which is based on the application of AIT.

The proof of concept and deliverable of this research work is a software prototype that implements an Evolutionary Algorithm (EA) to synthesize a software family of *Mobile User Interfaces* (MUIs). As other software development approaches, SPL can be assisted by AIT for automation purposes, so that, this work will be a reference for further applications of SPL in real life.

The scope of this dissertation

The scope of this work is to provide a prototype for automatic product derivation for

producing user interfaces for applications in *mobile devices* with Android operating system.

In the SPL landscape, this research work will deliver an approach and a software tool for automatic product derivation, from the natural speaking requirements, where the interfaces are bounded to those user requirements needed to depict a Lyker-scalate screen for survey mobile applications.

The *automation concept* addressed here is one that emphasizes the adoption of AIT as *automatic mechanism* over ad-hoc approaches. Future work, it should be lead to automate other parts of the software development process in SPL using AIT.

1.2 Motivation and Context

A lot of research in terms of provisioning to the SE with *automatic mechanisms* based on AIT can be found on literature, some of them ad-hoc approximations. However in terms of automation, AIT has been proved to be a very efficient approach in a big bunch of disciplines. In more than three decades AIT has been growing in a very constructive way and seeing itself benefited from the contribution of many scientific communities worldwide.

So the belief that by providing AIT mechanisms to the *automatic mechanisms* in SPL, beyond that these have also proven to be highly effectives for optimization and automation, ensures that, today and in the future, there is a structured methodological framework that supports *automatic mechanisms*, in contrast to the ad-hoc automation approaches. It is one of the reasons that encourage this work.

A lack of AIT targeted to specific developing frameworks or paradigms also depicts an open research field on software development engineering, due to the most of the work published in this field are mainly targeted to the research community. The research work is driven by the specific AIT used rather than the supported software engineering frameworks [3]. In this way, more effective research on this field should be leaded to a specific software development framework [3], so that, these endeavors might be part of software engineering specific practices, and hence, to be adopted in real life scenarios.

Under this context, the present research work is addressing this issue by

contributing to the formal approximation of an *automation concept* on one of the most challenging software development paradigms, the Software Product Line.

The expectations behind of automating the software process development is to reduce time to market and improve the quality of the final software product, what is crucial, given the last software systems are highly complex compared to their ancestors. These systems deal with a lot of complexity that includes and it is not limited to fancy user interfaces, remote services, security concerns and more complex infrastructures [2].

1.3 Organization of this thesis

Chapter 2. Provides an overview of the main subject addressed on this research work in order to facilitate the comprehension to the reader.

Chapter 3. The related research work is surveyed in this chapter.

Chapter 4. Deploys the prototype and/or proof of concept proposed by our work.

Chapter 6. Discusses the results gathered through the study case during the test cycle in order to confirm the correctness of this model.

Chapter 7. Conclusions and Future Work from our perspective is claimed here.

Chapter 2. Background

This chapter provides a review of the main topics addressed on this research work about Software Product Line and Automatic Product Derivation using Evolutionary Computation. In order to understand how the ETHOM [1] algorithm was customized for creating the prototype in the *mobile domain* introduced by this research which emphasizes the use of *automatic tools* assisted by Artificial Intelligence.

2.1 The SEI framework for Software Product Line

The idea behind a software product line is to exploit the communalities and tackle variability of a set of products related on a domain field to obtain a standard set of products, called *software family* [2]. Under this context, the domain is exploited until derive a set of software products or systems that share common features.

In the 1970s the concept of “product families” in which the SPL is supported was introduced by Parnas [4] to tackle variability in non-functional characteristics. However until 1990 and after a set of experiments supported mostly by European governments the concept was fully accepted, leading the paradigm of software development up to systematic approach [2]. Figure 1 below shows the history of the developing paradigms until now days.

Reuse History: From Ad Hoc To Systematic

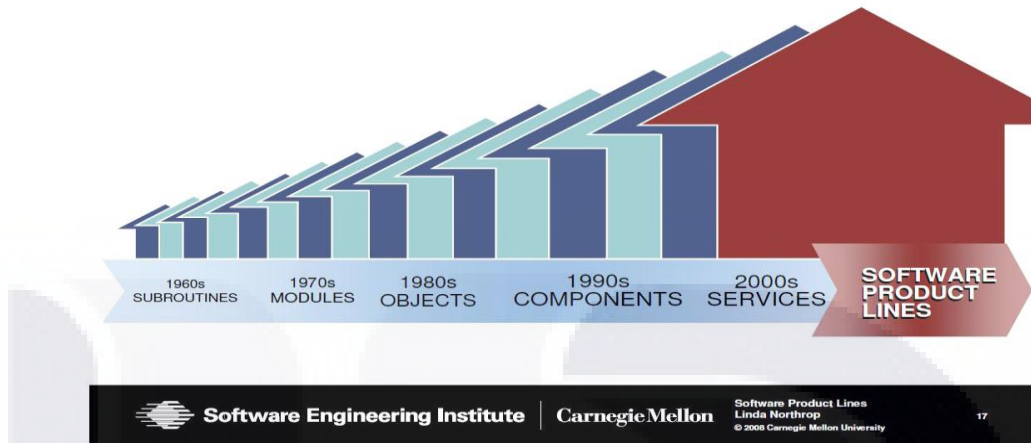


Figure 1. Reuse History: From Ad-Hoc to Systematic [5]

The Framework for Software Product Line practice is a property of the Software Engineering Institute of Carnegie Mellon University and it was introduced in the 1990s, after more than one decade, it has extended its large and respected knowledge base, thanks to the biggest practitioners community on SPLE, until deriving five refined versions of its respected framework which is supported by thousands of success cases for many different industries. A complete documentation of the framework, tools and industry cases can be found on its website.

<http://www.sei.cmu.edu/productlines/tools/framework/>

According to SEI community a right practice of SPL involves the synergy of three essential activities that are shaping on the Configuration Management (CM) of the Software Product Line depicted in figure 2.



Figure 2. The three essential activities for software product line [5]

The SPL practice consists in the mining of pre-existing assets (core asset development essential activity), by using these assets (product development) and under an organizational strategy (management essential activity) the software systems can be deployed. Samples of assets are software components, related documents and/or other sort of configurable sub-products that can add value to the final software system [5].

The core asset development activity refers the processes of creating and/or re-using assets and *production constraints* along with the *product constraints* to establish a *production strategy*, resulting of this, is to define the *production capability* of the software product line (SPL) which is composed by the core asset base and the production plan [5].

Thus, a new product endeavor requires of a new requirements analysis that leads with gathering the specific product features and product constraints. These constraints affect the design of the core assets during, for example, the architecture definition and the component development [5]. To alleviate the task of manage product requirements and constraints, some techniques are suggested by the framework, one of them is the FODA method below described.

FODA method

Every software product can be described in terms of their features. The FODA

method introduced by SEI community [6] describes a methodology to depict software systems in terms of these features where features are defined as "prominent or distinctive user-visible aspects, quality, or characteristics of a *software system* [6]". In terms of modularity, features also can be thought as components of a product, which provide a specific functionality and have a well-defined purpose. By this way, a family of related software products can be depicted by the features (components) of the set, constraints and dependences between them into a feature model [6]. Then the feature model contains all the configurable products that the software product line can produce.

Domain analysis and *Feature modeling* are two techniques that complement each other. They are part of the FODA method. Feature modeling proposes the exploration of communalities of a number of software products that co-exists in a well delimited domain to form a software family. A new software product can be added into the family by the exploration of its intrinsic communalities and the alignment of its variability to the context domain of the family in question [6].

FODA also introduced the graphical representation of a feature model known as feature diagram. The goal of feature diagrams is to provide a useful tool to analyzing and designing feature models. The basic notation of a feature diagram depicts relations between parents and children, as follows. [6]

Basic feature diagram notation

Basic models are known for being able to represent the following relationships among features:

- *Mandatory*. This relation exists when a child feature must be included in all products where the parent feature is shown [6].
- *Optional*. The child feature can or cannot be included in products which its parent feature appears in [6].
- *Alternative*. Only one feature of this set can be selected to be part of the product [6].
- *Or-Relation*. One or more features from a set can be inserted in the products in

which its parent feature is appearing [6].

A basic feature model can also depict constraints between features like this below.

- *Requires*. The inclusion of one feature implies the inclusion of another different feature. The difference from mandatory relations and this remains in that, does not exist a parental relation and sometimes the features in question are at the same level in the feature tree [6].

- *Excludes*. A feature excludes another one. It means, feature A and feature B cannot exist in the same product [6].

A simple sample of a feature model provided by FODA is depicted on figure 3, it describes a car in terms of its components and shows the relation between them in order to illustrate the basic notation of feature models.

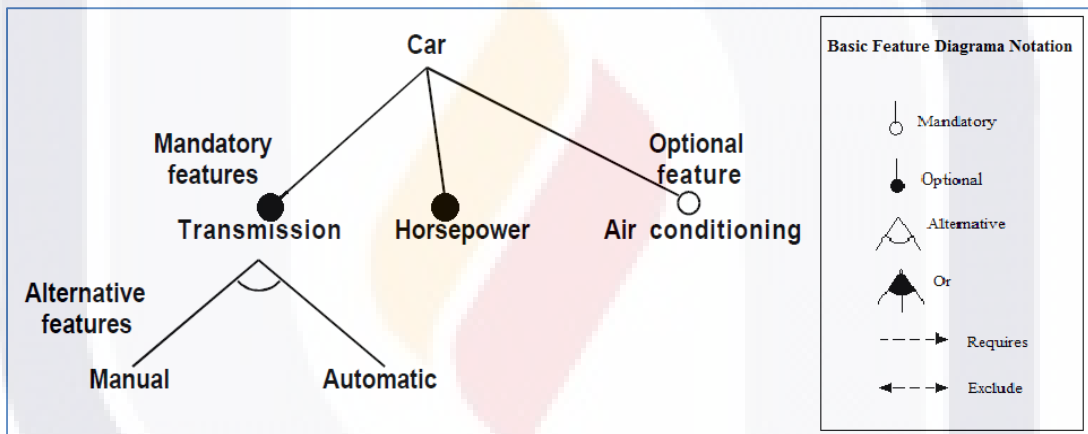


Figure 3. A simple feature diagram [6]

From the feature diagram on figure 3, four different cars configurations can be obtained on Table 1. The feature diagram is a tool widely adopted in the product configuration field in general due its feasibility for depiction of product families.

Table 1. Car Configurations

<i>ITM</i>	<i>Car 1</i>	<i>Car 2</i>	<i>Car 3</i>	<i>Car 4</i>
1	Automatic Transmission	Automatic Transmission	Manual Transmission	Manual Transmission
2	HorsePower	HorsePower	HorsePower	HorsePower
3	Air Conditioning		Air Conditioning	

2.2 Mobile User Interfaces Families

Now think about the user interfaces domain for applications in *mobile devices* applications. Into this domain, a lot of features like color, fonts, images, display layouts, and other functionalities on screen come into play for creating a *Mobile User Interface* (MUI).

Thousands and thousands of features, their dependencies and constraints work to create rich MUIs for millions of users worldwide. So that, the modeling of a software family of MUIs is a real challenge that starts by defining the boundaries for delimiting without sacrificing to functionality.

A minimal MUI can be built by picking at least four features from the set of features of the domain: one lay-out to organize the elements on screen, a user command bar, a text label and a background color, this mini-screen is showed on figure 4 below.

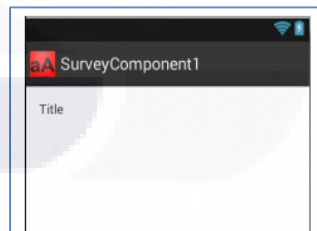


Figure 4. A minimal *mobile user interface* in Android.

Android provides two different approaches for developing user interfaces for *mobile devices*: Java code and the language descriptor XML.

The XML coding approach provides a way to develop *mobile user interfaces* by

focusing only in the elements on the screen leaving the functionality of them for specific java classes. Each XML label depicts a graphic widget in the user screen, the properties associated to each XML label are used to configure the visible and not visible features of each widget, for example, <textview> element has a property named color and the <checkbox> element in the line of code below will depict a checkbox button.

By using XML and the combination of elements on screen the design of a MUI is separated from most specific code for focusing only on design. The interfaces created under this specification, are isolated in a piece of XML code in a XML file, which make them reusable for other Android applications. In this way the *mobile user interface* can be consumed as an isolated component.

From this standpoint, a *mobile user interface* (MUI) is a component which belongs to one or many Android applications. In same way, a MUI is comprised of many sub-components, the XML labels or elements in it. Therefore, the XML elements are features of the MUI and they can be described into a feature model that depicts the MUI, as it is showed on figure 5 and its feature diagram representation.

Following this path, the domain of the *mobile user interfaces*, in terms of this research, was bound by the elements of XML in ANEX A which is the set of elements suggested by Android to build *mobile user interfaces* [7] . So that, feature models in this domain can be depicted using the elements of android XML, the constraints and dependences found in the Android Community website [7] . Figure 5 shows a sample of a small family of MUIs for a survey interface.

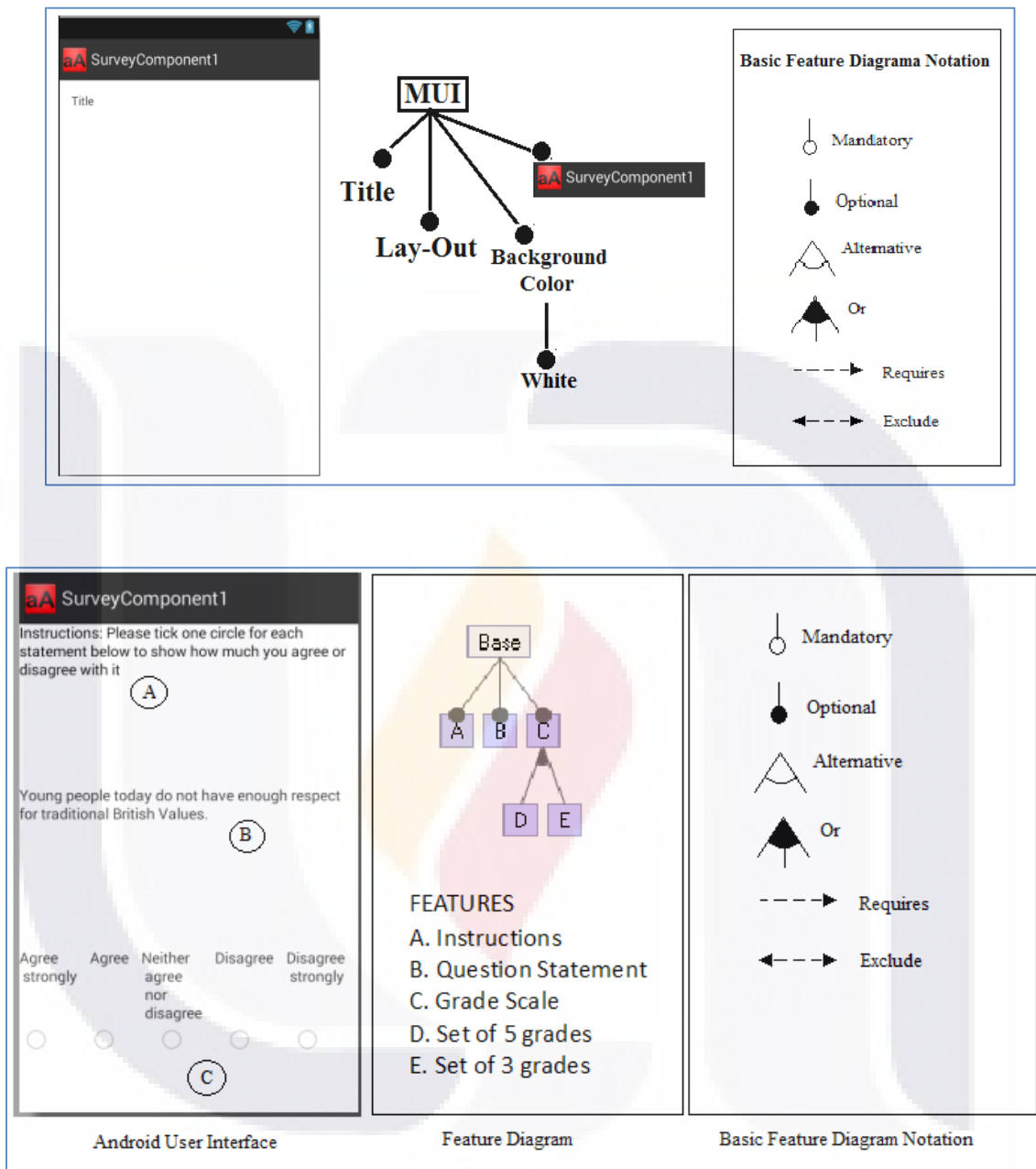


Figure 5. A basic MUI and its feature diagram

2.3 Automated treatment of feature models

Software systems and product configuration in general are more complex every day,

due the number of features in products is growing [8]. So the task to create and edit feature models to depict product families and the analysis of variability to introduce new products into the family is a rough task for domain designers [8]. However to deal with the problem, computer-aided extraction mechanisms have been introduced by researches to alleviate the task and reduce the amount of time it takes to domain designers [1]. These mechanisms have shaped a formal research area known as automated treatment of feature models [9].

By producing nodes and building relations between them randomly, is possible to create features models which are later treated to make them syntactically correct [1]. Propositional logic (PL), description logic, constraint programming, graph transformations, metaparadigm approaches and other not classified techniques have been used to model the problem to create automatically feature models (FM) [9].

In 2012 evolutionary computation was introduced as a solution to create high-complex feature models to test performance in software analysis tools, the approach was introduced by ETHOM [1] a novel algorithm based on Evolutionary Computation (EC). This research takes advantages of this approach to create *automatic derivation of mobile user interfaces* for Android applications.

2.4 Evolutionary Computation

The Artificial Intelligence comprised a set of computational mechanisms that are trying to re-create the natural process of how the live beings learn and surviving in nature. By the imitation of the nature processes, Artificial intelligence is providing with a kind of syntactic intelligence to the computer algorithms and reasoning. Artificial intelligence is a combination of mathematics, computing and engineering to build “intelligent” entities that is successfully contributing to solve a multidisciplinary set of problems [10].

The artificial intelligence is an empiric science that is growing mainly due to it is providing solutions in affordable computational times to complex problems in comparison to some mathematical approaches. A sample of this is Evolutionary computation (EC), which proposed a computer-based model to solve combinatorial optimization problems by using computational models that emulate biology evolutionary processes, such as natural

selection, survival of the fittest and reproduction as the fundamental components of such computational systems [11].

EC was inspired from the power of natural evolution of the diverse species in nature studied by Darwin, Lamarck and Mendel. Darwin proposed the "natural selection of the fittest" [12] while Mendel proposed "corpuscular theory of heredity" [13] and Lamarck proposed "Inheritance of acquired characteristics" [14]. By the work of these authors, the phenomenon of natural evolution can be described by the chances that an individual has to survive and multiplying in a particular environment. The most capable individuals will have more chances for reproduction and therefore to inherit genetic material to their offspring, in this way generations are improved, it means evolved [15].

Offspring, reproduced from two parents contain genetic material of both. The individuals who inherit the bad characteristics are weak and lose the battle to survive leaving the best adapted individuals to survive, analogous, best solutions, those which are nearest to the optimal will survive in terms of EC [15].

The idea of automated solving problem using natural selection can be tracked to the 40's with the introduction of the Turing machine in 1948 [16]. Turing proposed "genetic or evolutionary search". In 60's Bremermann performed experiments to optimize using evolution and recombination and then two different schools emerged: evolutionary programming and genetic algorithm, both in USA. At the same time Evolution strategies were proposed in Germany. But till 90's, the schools were viewed as different approaches of the same subject which was re-named Evolutionary computation [10].

The first formal evolutionary algorithm named Canonical Genetic Algorithm (CGA) was proposed by Holland [17] who based on the work of Fraser and Bremermann in 60's extended the genetic algorithm in combinatorial problem solving.

A genetic algorithm simulates genetic systems in order to model genetic evolution; here the characteristics of individuals are expressed using genotypes. The algorithm introduced by Holland implements the basic steps of an EA: A bit-string representation for the chromosome, proportional selection to select parents, one-point crossover to produce offspring and uniform mutation. These concepts will be introduced ahead in this chapter.

Evolutionary Algorithm

The Evolutionary algorithm (EA) is a general computational algorithm to solve *optimization problems* based on the empiric method proposed by Evolutionary Computation [15].

A well-known sample of optimization problem is the “Travelling Salesman” [18], which refers a problem of minimization. The goal is to find the shortest route that makes possible to a Salesman for visiting the more cities. The problem has been treated as a combinatorial problem, however, once the number of cities grows the computational times increase dramatically to set this problem as an NP-hard. From perspective of EC, good solutions has been found in affordable times by using evolutionary strategies like genetic algorithms.

The *optimization problems* are such where it is needed finding the maximum or minimum solution for an objective function in a solution space by doing local search [15]. So that the solution to a problem is depicted in terms of finding the most suitable solution for an objective function. This function is a $f : S \rightarrow R$ where S depicts a subset of real numbers and the goal is to find a x_0 such that $f(x_0) \leq f(x)$ for all x in S, in case of “minimization” or, such that $f(x_0) \geq f(x)$ for all x in S for maximization objectives.

General speaking, EA can be structured in six parts [15]: **encoding, fitness function, initialization, selection, reproduction and stop condition**. The first two are pre-set conditions which describe the computational structure to depict the candidate solutions and how to measure its fitness regards to an optimal solution. The four parts remaining are the logical structure of the algorithm and will be executed for same. For purposes of this research work we have included mutation and restriction techniques as suggested practices to add variation and avoid convergence. Figure 6 shows the operational workflow of an EA.

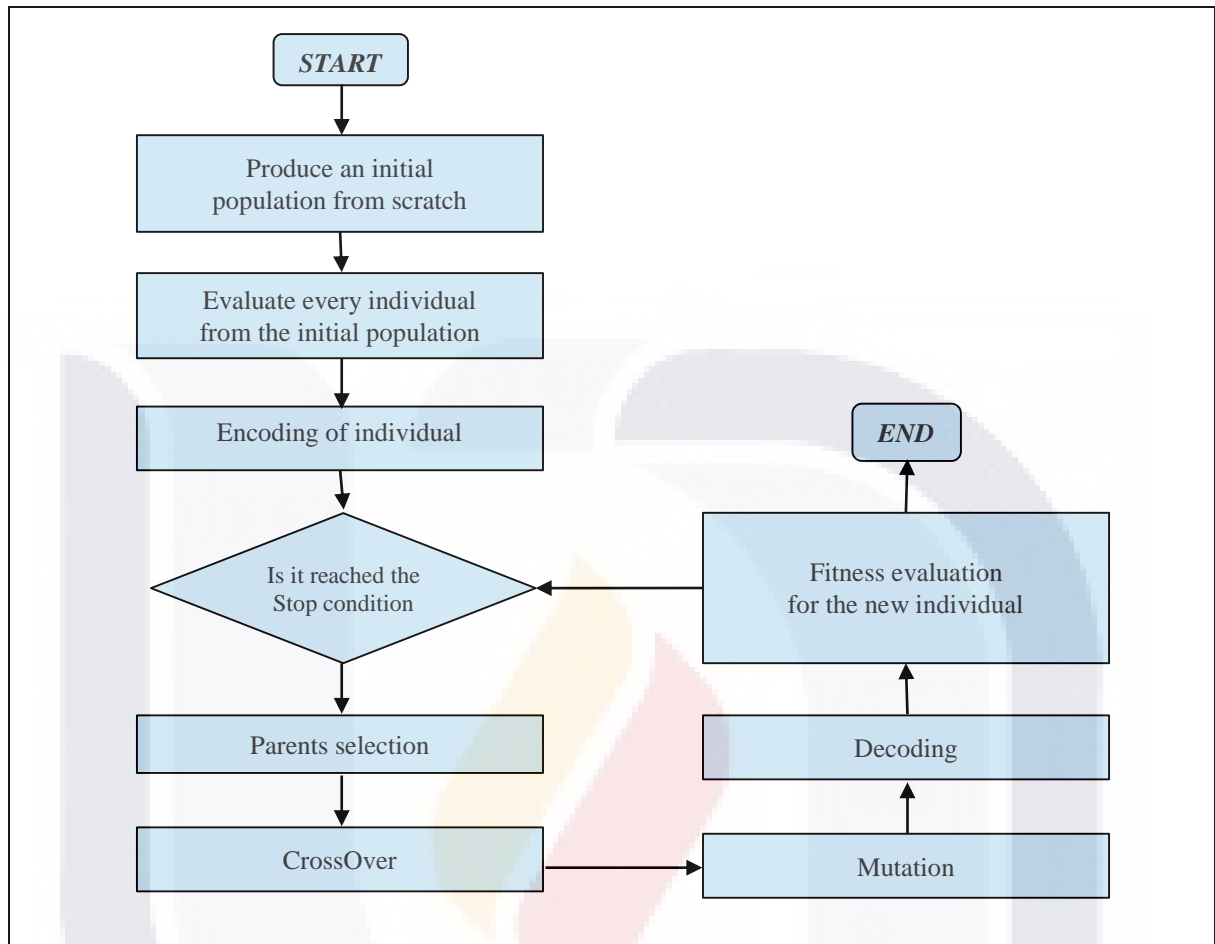


Figure 6. Shows the CGA workflow

This research work treats the programming of a *mobile user interface* as a combinatorial optimization problem where the goal is to design and code the most suitable user interface that meets user & aesthetic requirements, by using the minimum number of code lines. As we know this is a complex and rough task which starts by designing a set of candidate *mobile user interfaces* which will be evaluated by the effort required by a programmer for coding and testing. At the end, those interfaces with the minimum parameter will be picked.

Initial Population

The initial population is a set of candidate solutions to an optimization problem, created randomly from scratch. The initial population for the *mobile user interfaces* scenario above is set by a number of *mobile user interfaces* that meet user requirements and programming language restrictions.

Encoding a chromosome

Each solution in the initial population is known as the individual and the data structure that describes the individual is called chromosome [15]. Arrays of data and tree-like structures are the most used to depict chromosomes [11]. The most basic chromosome form is the bit-string structure that represents the binary form of a natural number [11]. This approach is gathered from the genetic encode of an AND of an individual which depicts a solution in terms of a computational data structure, that facilitates to an EA for applying operations over the individual like cross-over and mutation, this is described ahead [15].

The ETHOM [1] algorithm proposed the representation of a chromosome for a feature model by using two arrays of data. The first one that keeps the FM tree structure and a second one for describing relations of exclusion and/or inclusion. In figure 7 the ETHOM chromosome is depicted.

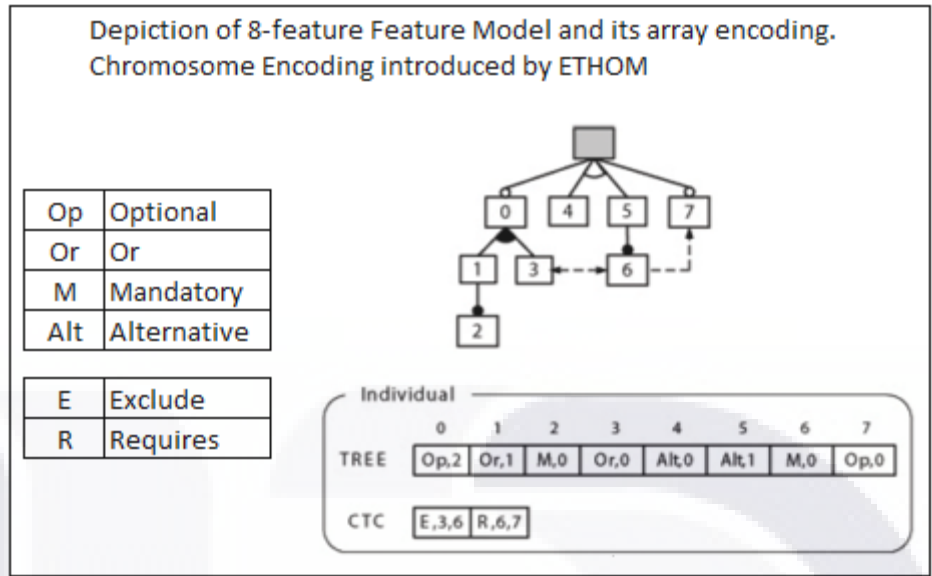


Figure 7. Encoding of a feature model, approach introduced by ETHOM [1]

For purposes of this research work, each feature in the ETHOM chromosome is linked to a specific functionality in the screen of the *mobile user interface*, such that, the ETHOM chromosome can provide support for modelling features models of MUIs families.

Fitness Function

In order to find best solutions (chromosomes) for an optimization problem, the individuals need to be evaluated to select those that are nearest to the optimal. Given EA is an empiric approach, non-deterministic methods are not available to find suitable solutions, so that, the only way is by measuring the quality of the candidate solutions and to compare [15].

Under this approach, metrics are gotten by the fitness function which measures strength of an individual to survive in a specific environment [11]. So, a fitness function is an objective function which is particularly useful to qualify how good the solution represented by a chromosome is, by grading the chromosome representation to a scalar

value.

The fitness function evaluates a candidate solution from a pool of possible solutions produced in a random basis or by evolutionary operators in a particular problem. The function returns the fitness of an individual [15]. ETHOM [1] proposed the fitness function in terms of the number of backtracks it takes for a software tool to traverse the feature tree, in order to find those feature models with the lees number of backtracks. The approach is discarded for this research work due it is not applicable.

The fitness function introduced by this research work is one that is capable to find the feature models to derive the *mobile user interfaces* with the least number of code lines that meet user requirements and programming constraints. This function is widely discussed on Chapter 4. In figure 4 below, it can be seen the mapping of MUI in the ETHOM chromosome.

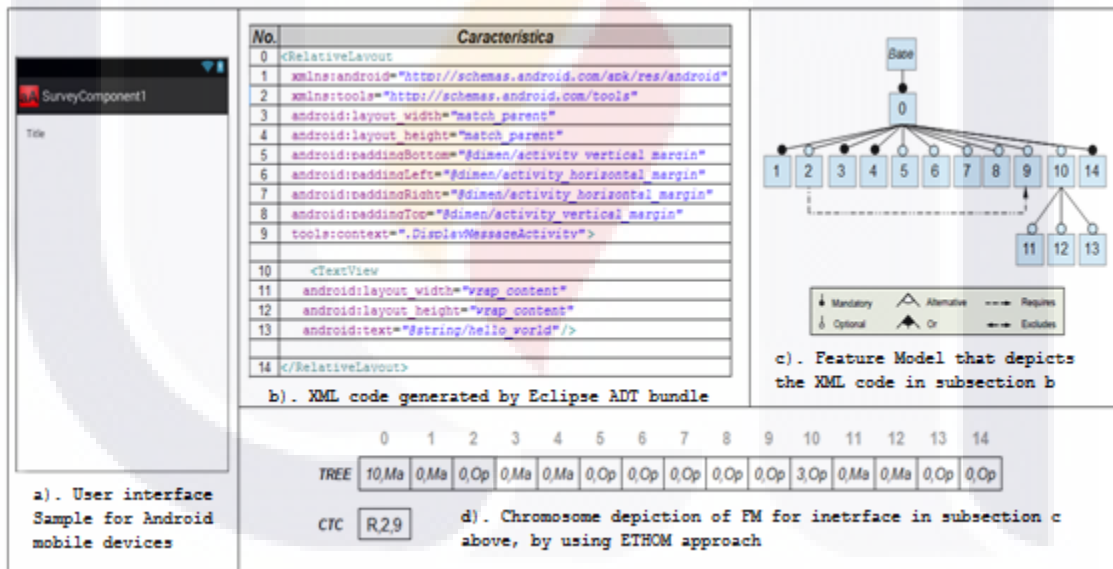


Figure 8. Chromosome called SurveyComponent1

Selection and crossover operators

Selection operators are techniques to choose best fit individuals in order to emphasize better solutions for reproduction and avoiding that the best individuals dominate

the landscape [11]. Once the fittest individuals are selected, these are subject of a reproduction process determined by the type of the crossover operators [11].

ETHOM [1] algorithm incorporates two crossover operators: rank-based selection and tournament selection. Both techniques were successfully implemented to brew new features models solutions for optimization. The approach was implemented in the prototype that validates this research work.

The rank-based selection method starts aligning the individuals in decreasing order by using its fitness values, then a probability of selection is calculated using formula, individuals are picked using the selections probability in roulette spin. In this way individuals selected are independent of the absolute fitness values, the advantage is that the best ones will not dominate the selection process and it will avoid divergence [11].

In tournament selection a set of individuals is randomly picked from the population, then individuals are compared in order to find the best one of this group, the process is called tournament [11]. A tournament round is performed by each parent, a crossover for two parents will need tournament selection to be done twice [11].

To illustrate the crossover process, the figure 7 shows an initial population of random natural numbers, the chromosome to depict each natural number is using its binary representation in an array, this bit-string structure is the most basic form of a chromosome [11].

In the figure also we have applied two operators: crossover and mutation. For crossover, the bit-string structures are combined by selecting the first two bits of parent A and the last one of parent B for breeding a new individual. The fitness function is defined in terms of the scalar value of $f(x) = x$. If a minimal is searched, minor values are the best fitness ones, by the other hand if maximal is the optimization goal maximum values will be selected as the best fitness.

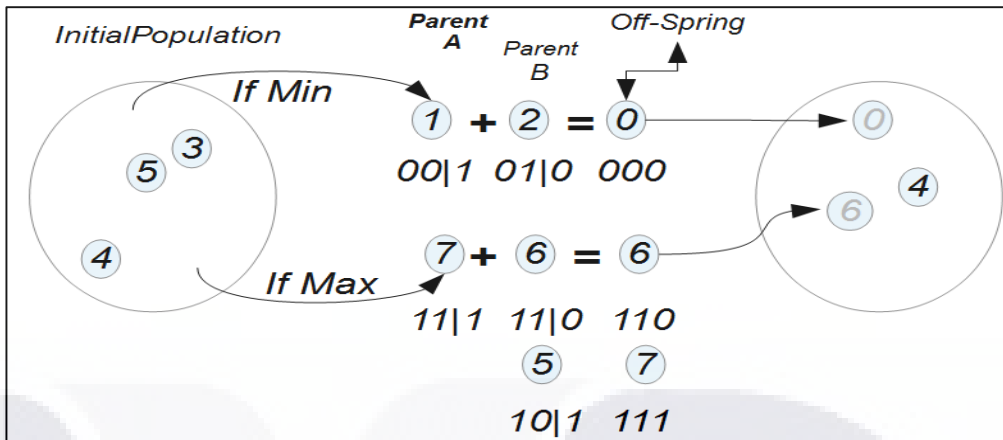


Figure 9. Encoding representation of natural number into a binary array

The sort of crossover defines how the parents are combined for breeding a new offspring. There are three types of crossovers categorized by the number of individuals that participate in the reproduction: *asexual*, when only one parent participates, *sexual*, two parents produce one or two more individuals and *multi-recombination* where more than two parents breed one or more offspring [11].

For combination, parents break into pieces, the new individual contains some parts of each parent. The point where the parents break is known as crossover point. These are the types of crossover points:

One-point crossover. Establishes a common point of crossover for both parents [11].

Two-point crossover. Allows break down the data array structures in more than two points [11].

"Cut and splice". Allows uncommon breaking points for both arrays [11].

Uniform Crossover and Half Uniform Crossover. The arrays are divided into multiples pieces in uncommon points and mixes [11].

Sexual reproduction on one-point crossover was implemented on the prototype of this research work as ETHOM algorithm does. In figure 9 is illustrated the process, the chromosome that depicts the feature model according to ETHOM [1] is comprised by two arrays, a random crossover point in each array must not be greater than the maximum size

array, then the left part of parent A and the right part of parent B are mix on the new individual.

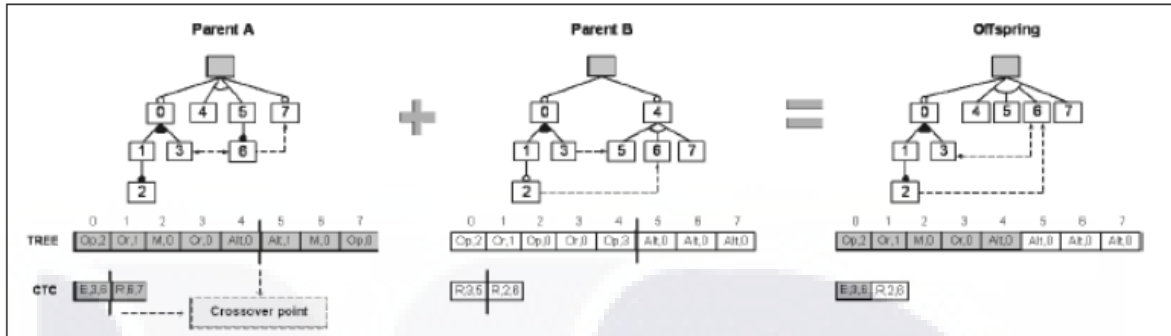


Figure 10. Example of one-point crossover introduced by ETHOM [1]

Mutation

The goal of mutation is to introduce new genetic material into the population, thereby increasing genetic diversity. It can be done by random changes on the genes of a chromosome [11]. The mutation operators are ad-hoc mechanisms regarding to the particular problem.

ETHOM [1] proposed four mutation operators, below:

- Operator 1 changes randomly the type of a relationship between parent and child [1].
- Operator 2 changes randomly the number of children of a feature in the tree [1].
- Operator 3 changes the type of a cross-tree constraint in the CTC array [1].
- Operator 4 changes randomly (with equal probability) the origin or destination feature of a constraint in the CTC array [1].

Constraints and infeasible individuals

In EC, the automatic generation of solutions by random methods, crossover or mutation can lead sometimes to produce infeasible individuals in a constrained solution space. The techniques to tackle infeasible individuals those who are violated at least one constraint, are three different approaches: rejection of the infeasible individuals, individual

repairing by solving the unconstrained problem using information of the feasible solution space, and discourage to the search mechanism to look for solutions on the unfeasible space like penalty functions and pareto ranking, it last use concepts from multi-objective optimization [11].

The repairing approach of the FM from ETHOM [1] was used for the proposed prototype by this research work. This approach avoids semantically redundant feature models. The mechanisms of repairing are below.

a).- Relationships (or- and alternative) with a single child feature are changed by optional relationships [1].



Figure 11. Relations (or- and alternative) with a single child repairing [1]

b).- Cross-tree constraints between features with parental relationship are removed [1].

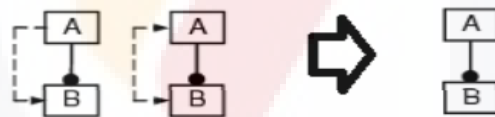


Figure 12. Cross-tree constraints between features with parental relationship repairing [1]

c).- Contradictory or redundant cross-tree constraints changed by implications [1].



Figure 13. Contradictory or redundant cross-tree constraints repairing [1]

Every MUI also must meet a set of requirements stated by a customer or end user. These requirements were thought as constraints for the chromosome. A pool of solutions

represents all possible combinations of MUIs that meet the user requirements, and to treat this problem, an ad-hoc mechanism in prototype section in Chapter 4 was added to the repairing techniques exposed here.

Survival and stop condition

Finally, offspring should consider a replacement policy and stop condition in order to avoid an infinite growing of the population. One approach is that the offspring may replace the worst parent as long as its fitness is better compared to the one which is replacing. This approach has been adopted by this work.

2.5 Software Automation and SPLE

The *automation concept* that this dissertation supports on SPL, it can be defined as the replacement of human hand by a software tool which is aided by intelligence artificial mechanisms and it can be tracked back to the research work of [19] at the 80's. . The software tool in question is a robot in terms of manufacturing

The approach of automation in Framework 5.0 is focused in the automatic product derivation. According to SEI “Automated derivation of products requires an explicit tool chain that begins with domain concepts and ends with executable code”... [5]

In this context, automatic product derivation proposes a way to create software product automatically from requirements. However, natural language speaking requirements cannot be exploited under this approach, requirements needs to be processed into a model that can be fully read by automated tools.

The different paths followed to addressed this task have been categorized by SEI [5] in terms of how well-defined is the semantic y the type of modeling language that is specific for creating the model in question. The five main approaches are specification-based, intelligent build, domain-specific languages (DSLs) and product generation, meta-modeling and frame technology”. These approaches are not at the same level of abstraction, and they are not mutually exclusive. [5]

Chapter 3. Related Work

This chapter reviews the research work on the main fields addressed in this dissertation. Cites the efforts for automating the software engineering process using AIT, its applicability on SPLE as well on the *mobile domain*. Also, the verification of the correctness of automated synthesis of feature models on *mobile domain* is surveyed here.

The efforts to improve the software engineering process started at the 1960's with the adoption of machine learning to create formal specification from informal requirements in natural language [3]. Later at the 80's a formal concept of software automation based on AIT is introduced in [19] and supported by the intensive work to automate the whole process of software developing. However this is not led to a specific software developing framework leaving these efforts to the research community only.

Three broad areas of AIT have most impacted the software engineering process: searching and optimization, fuzzy and probabilistic methods for reasoning under uncertainty and, machine learning which includes classification and prediction [20].

In software engineering, the efforts can be classified in three branches.

Probabilistic Software Engineering which refers to the treatment of noisy and incomplete information using fuzzy and probabilistic methods, some samples are Bayesian probabilistic reasoning to model software reliability and the stochastic nature of human behavior.

Classification, Learning and Prediction for Software Engineering which refers to the increasing interest in modeling and predicting software costs as part of project planning, traditional machine learning techniques such as artificial neural networks, case based reasoning and rule induction have been used for software project prediction.

Search Based Software Engineering, in this classification, the software engineering problems are treated as optimization problems that can be solved with computational search, some applications are led to requirements and design, maintenance, and testing [20].

In the SPL field the efforts for automating are focused on the automatic product derivation [5] which have driven the automated analysis of feature models to become a

well-established research area that have attracted more attention in the last two decades [9] given the feature model is one of the most used approaches to model variability and communality in SPL. The automated analysis of feature models from AIT perspective has been treated for ETHOM [1] a novel algorithm based on EC which has proved being more effective to process and analyzing hard-computationally feature models on an affordable machine cost.

This evolutionary algorithm demonstrates to be more efficient than random search in finding input combinations for maximizing or minimizing execution times to synthesize feature models with functional and non-functional features. The goal of ETHOM [1] is to provide an *automatic mechanism* that produces hard-computationally feature models for testing purposes. So in this way, the feature models produced by ETHOM [1] have been proved to be more effective for finding processing errors on FM analysis tools compared to those feature models created randomly.

Given the novelty of this work its applicability is scarce in literature, however, other application of ETHOM was found on reverse engineering, this work proposed by the ETHOM authors is based on the statement that a “SPL does not need to be created from scratch”, the features of pre-existing products can be fed to the ETHOM algorithm in order to create a new SPL. This is particularly useful due to the labor of starting a new SPL can be error prone [21].

The efforts for automating in SPL based on artificial intelligence are mainly targeted to the proper selection of features from the stakeholders concerns and the functional/non-functional requirements selection, techniques like Hierarchical Task Network (HTN) [22], fuzzy logic [23], Analytical Hierarchy Process (AHP) and Fuzzy Cognitive Maps (FCM) [24], genetic algorithms [25] were used.

Another effort for automation in SPL show successful ad-hoc mechanisms in the labor of automatic support for product derivation of software for electronic devices [26], model-driven techniques for automatic product derivation on mobile software [27] that takes advantages of the conditional compilation and aspect-oriented programming, and automated testing [8].

Also, some prototypes and tools to support automation on SPL without AIT can be found on [28] that uses rational rose to derive products from UML multi-model views. In [29] a prototype for automatic analysis of the security requirements and [11] where automatic derivation is treated using a repository approach [30].

The adoption of AIT techniques in SPL is scarce, but a trend for synergizing, product configuration and SPL, promises benefits in the long run to the automation on SPL assisted by AIT due to product configuration is another well-defined research area with more than 30 years of application in many different industrial domains and of course a long history on the successful application of AIT for automation purposes [8].

The automatic product derivation based on SPL which is the subject of this research work for specific *mobile android applications* is non-existent in literature, but frameworks with automatic code generation and tools for automated android software development with no SPL support can be found, from XML to java codes by eclipse plug-ins [31].

Trends that adopt AIT in *mobile domain* only can be traced to contextual adaptive user interfaces [32] that implement machine learning.

As we demonstrated at this literature review, the state of the art in SPL support and automatic product derivation for mobile software are scarce in literature, approaches based on AIT were not found. Only ETHOM [1] provides a mechanism for automatic product derivation in SPL that implements AIT but it have not been tried on mobile software domain, which is the reason to integrate it into our proof of concept.

Chapter 4. Study case: Configuration of a SPL for user interfaces at mobile devices domain

This chapter introduces the study case and the prototype created to support the theoretical basis sustained by this research work. It explains the functionality and construction of the prototype and its applicability in the *automatic derivation of mobile user interfaces* for survey applications.

4.1 The study case

According to the World Bank in 2012, there were about six billion mobile subscriptions in use worldwide, about three quarters of the world's people now have access to a mobile phone. In some developing countries, more people have access to a mobile phone than a bank account, electricity or even clean water. Mobile communications now provide great opportunities to advance human development, from basic access to information and education health to make cash payments to encourage participation of citizens in democratic processes, etc [33]. Along with the technological mobile base growth, the increasing of functionality of Smart phone devices in recent years have given rise to an explosion in developing applications for this sector. This trend obliges to the researches on software automation to innovate in this field.

The prototype introduced in next sections is a proof of concept to support the automation concept in SPL assisted by AIT. It performs automatic product derivation through domain analysis proposed by FODA method and claimed as a good practice in Framework 5.0 in order to achieve an approximation between practicing of SPL discipline in real life and AIT. The prototype proposes a method to derive software products for electronic devices by exploding the Android platform. This prototype turns around of an evolutionary algorithm developed in java, which by means of automatic synthesis of feature models introduced by ETHOM [1] produces feature models which are exploited by the tools to the get the product configurations.

The case of study that supports the user requirements which shape the *mobile user*

interfaces derived under this approach, it is based on the domain of applications for Mobile Data Collection or Mobile Surveys which is a growing trend of data collection where a survey form, application or collection tool is on a mobile device such as a Smartphone or a Tablet and by using an online platform data gathered can be retrieved and processed [34]. This data collection trend is growing due low costs and short times needed for applying a survey compared to traditional methods [34].

Apart from the high mobile phone penetration, *mobile devices* and their applications offer innovative paths to collect data regardless of time and location of the inquired [35], further advantages expected are quicker response times and the possibility to reach previously hard-to-reach target groups [36]. Another advantage of using a *mobile device* for data collection is to gather information more discreetly if needed papers. For example, an observation survey could be performed on a smart phone or tablet instead of a clipboard or stack of survey papers [35] [36].

The first online commercial surveys platforms that provides mobile capabilities and total customer business integration through SOA are appearing into market in order to deliver on-line survey services, under this context, our contribution is to provide *mobile user interfaces* which can be reused as components inside of On-line Survey Data Collection Platforms.

By other hand, in order to demonstrate the applicability of the model and the tool introduced by this work, every *mobile user interface* must meet a set of requirements stated by a customer or end user. These requirements were thought as constraints for the chromosome. A pool of solutions represents all possible combinations of *mobile user interfaces* that meet the user's requirements which are obtained through the selection of the requirements set provided on table 2 below. This feature set is extracted from those suggested by Android Community [31] and it depicts the basic set of features requested to create *mobile user interfaces*, the set also compounds our feature domain as it is explained ahead, so that, a user screen for a Lyker-scale basic question [37] over a *mobile device* can be shaped into the set of features of Android.

Table 2. Feature set for Lyker scale question

Feature	UI Component Description	SurveyComponent1
A.	Instructions	✓
B.	Question Statement	✓
C.	Grade Scale	✓
D.	Set of 5 grades	✓
E.	Set of 3 grades	
F.	Question number	
G.	Button Bar	
H.	Comments text box	
I.	Disclaimer Legend	
J.	Background Image	

From model above, many *mobile user interfaces* will be automatic derivatives using this approach from XML Android tags and the natural speaking requirements on table 4. Initially MUIs for questions with multiple choice of three and five grades will be produced.

In the next paragraphs the *Mobile Android Product Derivation Software* (MAPDS) tool is explained to prove the correctness of the model introduced.

4.2 The MAPDS prototype

The MAPDS prototype is an object-oriented software tool developed on Java SE 7, as a standalone application it can be run on any Windows 7 desktop machine. MAPDS takes a subset of XML labels into the set of Android labels as an input parameter and turns them into XML *mobile user interfaces* that can be run in any Android device like figure 16.

```

<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="#c0c0c0">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />

</RelativeLayout>

```

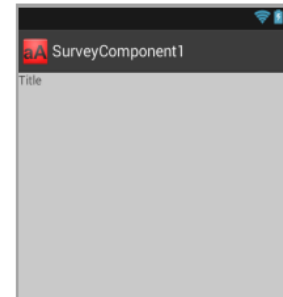
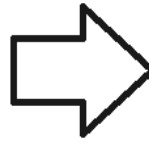


Figure 14. XML code and its MUI

Any MUI created by MAPDS can be re-used in more complex Android mobile applications due to the XML support provided by Android which allows to the screen functionality to be encapsulated.

A run cycle of MAPDS provides a set of *mobile user interfaces* that are constrained by a set of user requirements which were described in the study case section above table 2.

The tool is comprised by four classes: MainFrame, ATM, FeatureModel and Configurator. They interact each other to provide with an interface to the final user which allows selecting the configuration parameters of the evolutionary algorithm to perform the synthesis of feature models and the set of their derived products.

The ATM class deploys the evolutionary algorithm engineering to select the best feature models individuals. FeatureModel class is the feature model template to instantiate the FM individuals (objects) that will comprise the FM population. Mainframe class deploys the *mobile user interface* to interact with, it instantiates the ATM class to start the automatic process development. The figure below shows how these classes interact, each class has a well-defined goal and function explained ahead.

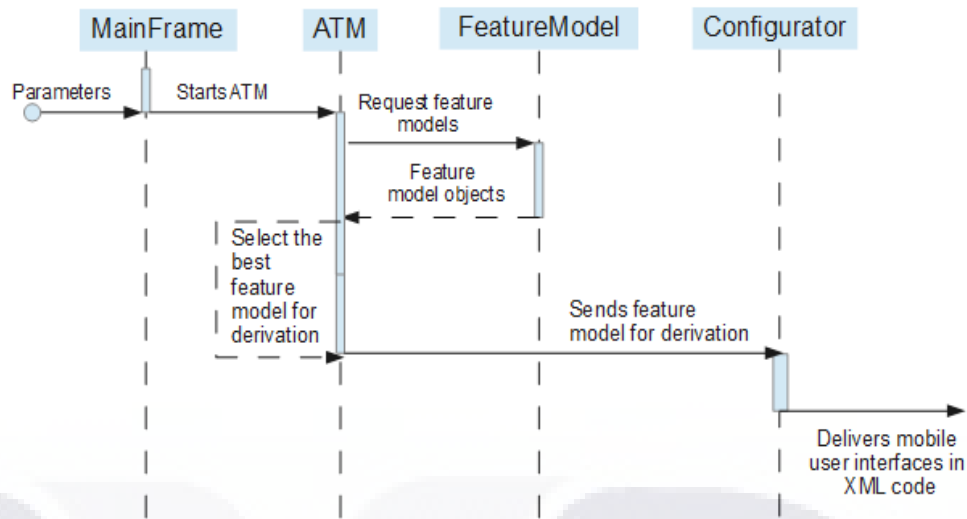


Figure 15. The class interaction on MAPDS tool

The *mobile user interface* of MAPDS provides the selection of input values for the evolutionary algorithm embedded, the user can choose from seven parameters to choose and create combinations.

Selection strategy parameter defines how the individuals will be selected for cross-over. At this stage only the one-point strategy is available to pick from the screen.

Mutation probability will define the percentage of individuals to be mutated from the initial population in terms of an EA.

Size initial population parameter sets the size of the population.

#Executions fitness function set the stop condition for this EA.

Infeasible individuals parameter defines whether the infeasible individuals will be replaced or repaired and finally *Features* parameter will establish the size of the features models.

Screen in figure 18 shows the configuration display to interact with. Once the parameters are selected, by clicking “Make Calculations” button, the MUIs are produced into the application directory and ready to be re-used in an Android application by adding the file into the /res directory in the Android *.apk project.

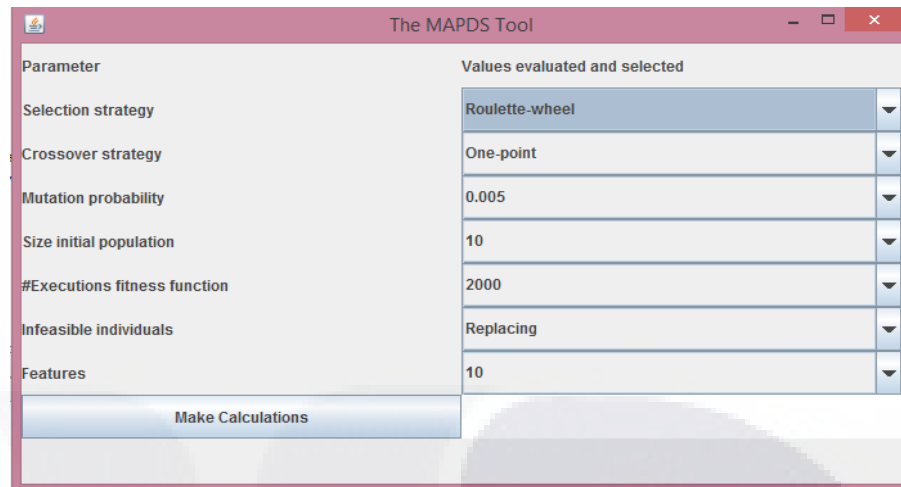


Figure 16. MAPDS *mobile user interface*.

Currently the prototype derives *mobile user interfaces* in XML code. Programming *mobile user interfaces* for *mobile devices* using Android can be coded through Java and XML languages, our prototype builds *mobile user interfaces* in XML, given the modularity of XML tags to represent on-screen functionality, a label in terms of this research represents a unit of functionality and looks like a building block, these blocks are the assets of our product line and it use them to configure software products (*mobile user interfaces for mobile devices*), such that a *mobile user interface* is a combination of XML tags that are there because of a required configuration.

The tool then takes the set of XML tags provided by Android Community [7] for building *mobile user interfaces* in table 2 and the properties, as the set of features which features models will be interpreted with.

Table 3. Set of Android elements most common used to build MUIs extracted from Android Community [7]

1	<RelativeLayout></>	22	<FrameLayout></>	43	<ImageSwitcher></>
2	<GridView></>	23	<TableLayout></>	44	<AdapterViewFlipper></>
3	<Button></>	24	<TableRow></>	45	<StackView></>
4	<EditText></>	25	<ListView></>	46	<TextSwitcher></>
5	<CheckBox></>	26	<ExpandableListView></>	47	<ViewAnimator></>
6	<RadioButton></>	27	<ScrollView></>	48	<ViewSwitcher></>
7	<ToggleButton></>	28	<HorizontalScrollView></>	49	<ViewFlipper></>
8	<Spinner></>	29	<SlidingDrawer></>	50	<requestFocus></>
9	<TextView></>	30	<TabHost></>	51	<View></>
10	<AbsoluteLayout></>	31	<TabWidget></>	52	<ViewStub></>
11	<ToggleButton></>	32	<WebView></>	53	<GestureOverlayView></>
12	<CheckedTextView></>	33	<ImageView></>	54	<TextureView></>
13	<Spinner></>	34	<ImageButton></>	55	<SurfaceView></>
14	<ProgressBar></>	35	<Gallery></>	56	<NumberPicker></>
15	<SeekBar></>	36	<MediaController></>	57	<ZoomButton></>
16	<RadioGroup></>	37	<VideoView></>	58	<ZoomControls></>
17	<Switch></>	38	<TimePicker></>	59	<DialerFilter></>
18	<RadioGroup></>	39	<DatePicker></>	60	<EditText></>
19	<EditText></>	40	<CalendarView></>	61	<TwoLineListItem></>
20	<AutoCompleteTextView></>	41	<Chronometer></>	62	<TextClock></>
21	<MultiAutoCompleteTextView></>	42	<AnalogClock></>	63	<LinearLayout></>

The ATM class

The instantiation of ATM class starts the initial population, which is stored in initialPopulation variable into the class. The initialPopulation variable is an array of FM class objects which are created by means of the instantiation of several feature model objects.

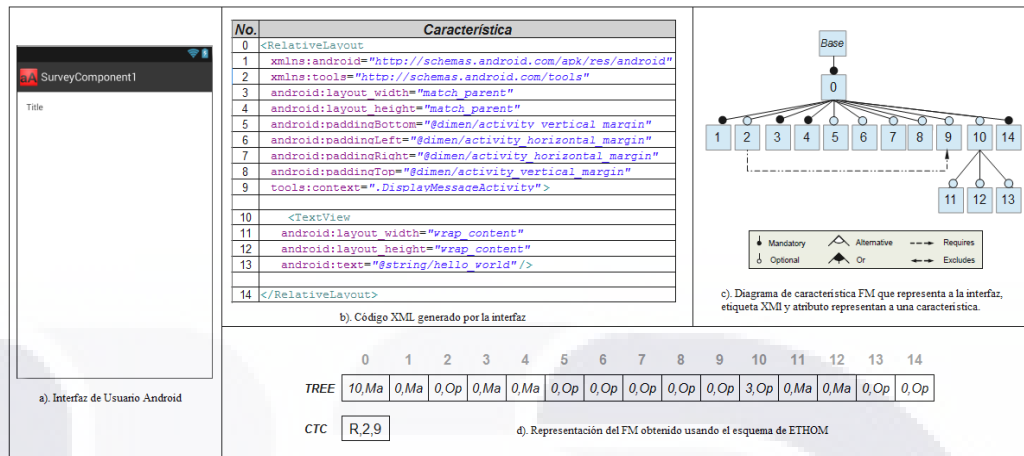


Figure 17. Relation between the ETHOM chromosome and the MUI code based on XML

The initial population is created by instantiating a number of FMs which are created randomly. The number of FM objects in the initial population is an input parameter of the *mobile user interface* MAPDS tool. It sizes the initialPopulation array, which will be fill by the initialPopulation() method in ATM class.

FM Class and the FM individuals

Each FM object is an individual created using the algorithm for the random generation of feature models proposed by [38], which describes a transversal-tree of nodes, where each node represents a feature on the feature model. Originally, the nodes are comprised by the child number and relation type of this node regards to its father, but for effects of this research it was added a number to identify the Android feature depicted by the node.

In the random approach to create feature models [38] built-in by ETHOM [1] to create the initial population, a node is randomly selected from the set of nodes and this is assigned a random number of children and a random relation with its father. All the actions to create the tree-like structure of a FM object are deployed on the FM class constructor,

the code in FM is inserted below to illustrate the process.

```
/** FM constructor */
public FeatureModel (int ft) {
    nodes = ft;    --> describes nodes from initial Size Population variable
    FMMatrix = new int[ft][3];    --> Creates FMmatrix storing the node tree

    /** Creates a temp set of nodes which can choose from */

    int perassign = nodes -2;
    int[] tempNodeSet = new int[nodes]; //Fills temp node Set

    for (int i=0; i<perassign; i++)
        tempNodeSet[i] = i + 1;

    /** Fills FMMatrix with n nodes */

    /** Constraint 1 - First feature is mandatory */

    FMMatrix[0][relation] = 1;
    FMMatrix[0][children] = xRandom(nodes-1);

    int totalchildren = FMMatrix[0][children];

    for (int i=1; i<nodes-2; i++) {
        int xset=0, xnode=0;
        boolean invalidNode = true;

        xset = xRandom(perassign);    //Select random position
        xnode = tempNodeSet[xset];    //Select node from feaure set

        perassign--;    //Decrease total of nodes to assign
        tempNodeSet[xset] = tempNodeSet[perassign];

        //Copy last node to xset pos to reduce the set

        FMMatrix[xnode][relation] = xRandom(4);
        if (totalchildren > xnode)
            FMMatrix[xnode][children] = xRandom(((nodes-2)-totalchildren)+1);
        //Assign progenie
        else
            FMMatrix[xnode][children] = xRandom(((nodes-2) - xnode) +1);
    }
}
```

```

totalchildren = totalchildren + FMMatrix[xnode][children];
} // end for statement

/**Constraint 2 - Last feature is mandatory ***/

FMMatrix[nodes-1][relation] = 1;
FMMatrix[nodes-1][children] = 0;

/******* Ended Fill out FMMatrix *****/

if (nodes > 2)
    fillOutCTCMatrix(); //Fill Out CTC Matrix
    setFitness();
} // End FM class constructor
    
```

In MAPDS FeatureModel class, the FMMatrix array stores the feature model tree [1] and the CTCMatrix [1] array stores the feature model constraints for this tree. So that, the chromosome is comprised by these two integer arrays.

=====	4,0,Or,0
=	5,0,Or,0
* Chromosome No. 0	6,0,Mandatory,0
=====	7,0,Alternative,0
=	8,0,Or,0
C T C Matrix	9,0,Mandatory,0
*****	10,0,Or,0
* R,56,21 E,28,26 *	11,0,Optional,0
*****	12,0,Optional,0
Feature Model Matrix	13,0,Optional,0
*****	14,0,Mandatory,0
0,26,Mandatory,-1	15,0,Or,0
1,0,Mandatory,0	16,0,Mandatory,0
2,0,Or,0	17,0,Optional,0
3,0,Optional,0	18,8,Mandatory,0

19,0,Or,18	39,0,Alternative,27
20,0,Optional,18	40,0,Alternative,27
21,0,Or,18	41,0,Mandatory,27
22,0,Alternative,18	42,0,Or,27
23,0,Mandatory,18	43,2,Mandatory,27
24,0,Or,18	44,0,Optional,43
25,0,Or,18	45,0,Alternative,43
26,0,Optional,18	46,0,Mandatory,27
27,18,Or,0	47,0,Mandatory,27
28,0,Alternative,27	48,0,Or,27
29,0,Alternative,27	49,0,Mandatory,0
30,0,Optional,27	50,0,Mandatory,0
31,0,Optional,27	51,0,Alternative,0
32,0,Optional,27	52,0,Optional,0
33,0,Alternative,27	53,0,Or,0
34,0,Mandatory,27	54,0,Optional,0
35,1,Mandatory,27	55,0,Optional,0
36,0,Alternative,35	56,0,Mandatory,-1
37,0,Optional,27	*****
38,0,Or,27	

In this sample, the FM has been converted in a chromosome which is created based on 56 features on table 3 where CTCMatrix [1] contains two constraints: nodes 56 and 21 are mutually required and nodes 28 and 26 are mutually excluded for this particular sample.

The number of children in the node is calculated by extracting a random number between the children sum and the maximum branching. In same way, the relation in the node is assigned using a random number from 0 to three where values are depicted as, 0="Optional", 1="Mandatory", 2="Alternative" and 3="Or". The position in the array represents the feature number [1].

The MUI chromosome

Given the space of solutions to our problem is found into the *mobile user interfaces* set, the individual or chromosome is represented by a piece of XML code that runs as a *mobile user* interface in an android user device. Even though a MUI can be represented in terms of java code, the decision to use XML is due, XML simplifies the coding of *mobile user interfaces* in Android and provides modularity by using descriptors for java code pieces. By bounding XML elements to visual features in screen a feature model for a MUIs family can be depicted. See figure 14 below.

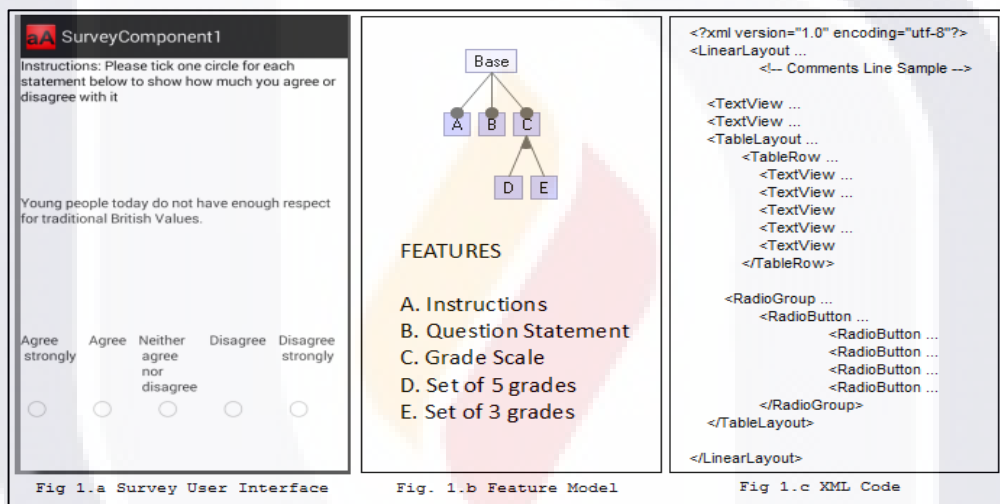


Figure 18. Feature model of SuervyComponent1

The chromosome prototype of ETHOM [1] to depict feature diagrams is used in this research work to represent feature models of MUI, due it is the first approach found in literature to depict feature models in terms of EC and it accomplish with the *automation concept* proposed in this work for a SPL in terms of SEI [5].

The translation of Android MUIs written on XML code to FM chromosomes starts by pointed out every XML label into a feature of the MUI. The chromosome is comprised by an array that depicts the transversal feature tree and a second one that stores the

constraints relations of the feature model, in figure 15 both are deployed for reference, the ETHOM chromosome was introduced on Chapter 2.

The existence or not of a label exposes a feature on the MUI over the android screen these approach helps easily link features with code components. The FM chromosome according to ETHOM [1] can be depicted using two arrays, one for describing the transversal tree and the other one for the CTC constraints.

As illustrated on figure 15, each node in the tree array contains the number of children and type of relation of this node related to its parent. The four types of relations are described using Ma = mandatory, Op=optional, Alt=alternate, Or=or-relationship, the CTC array uses “E” for exclusion and “R” for requires.

According to ETHOM [1] authors, “generic encodings for evolutionary algorithms were ruled out since these were either not adequate to represent tree structures (i.e. binary encoding) or were not able to produce solutions of a fixed size (e.g. tree encoding), a key requirement in our approach... [1]”

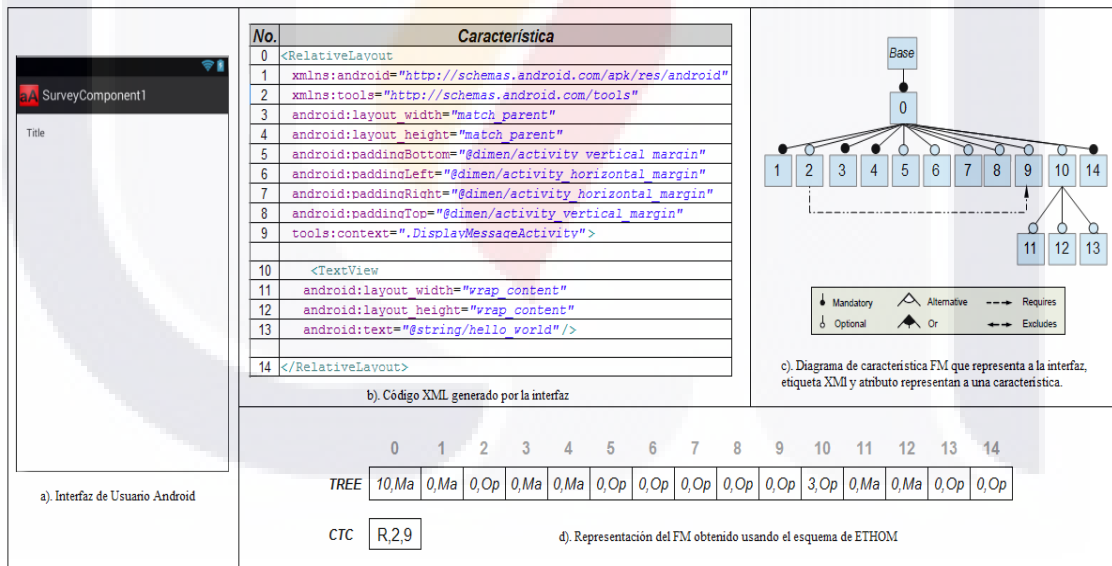


Figure 19. MUI feature model proposed on this research work and its translation to ETHOM [1] chromosome

The fitness function on MAPDS

ETHOM proposed measuring the fitness of a FM chromosome by counting the number of backtracks required for an analysis tool to cross the transversal-tree, the approach was discarded here.

In terms of this research, the goal is to look for the most fitted derived product regardless the FM in question, to do this, it is needed optimizing the quality of the products produced. The FM only needs to be created according to FODA.

In consequence, this work proposes evaluating the fitness function of a FM chromosome by counting the number of XML elements and attributes in the MUIs. So that, the optimization goal is the less number of elements and attributes of the MUI XML code and therefore the less number of code lines required for coding a MUI.

The fitness function proposed to measure will be,

$$f(x, y) = \sum x + \sum y$$

where f is the fitness function, “x” is the number of Android elements depicted by the XML tags [5] listed on table 4 and “y” is the weight of every Android properties of each element that can be listed on [7].

In order to illustrate the idea, the sample code below is showing how the fitness is evaluated by counting relevant elements which are underlined.

By making calculations with the fitness function, the scalar gotten of the MUI sample code is 94 Fitness sample is below.

$$f(x, y) = 16 (elements) + 78 (properties) = 94$$

The sample XML code is below,

SAMPLE CODE

```
<?xml version="1.0" encoding="utf-8"?>    ols"
<u>LinearLayout                        Android:layout_width="match_parent"
xmlns:Android="http://schemas.Android.com Android:layout_height="match_parent"
/apk/res/Android"                        Android:alpha="1.0"
                                           Android:orientation="vertical" >
xmlns:tools="http://schemas.Android.com/to
```

<!-- Comments Line Sample -->

<TextView

Android:id="@+id/textView1"
 Android:layout_width="fill_parent"
 Android:layout_height="0dp"
 Android:layout_weight="1.0"
 Android:gravity="left"
 Android:text="@string/instructions"
 Android:textColor="@color/black" />

<TextView

Android:id="@+id/textView2"
 Android:layout_width="fill_parent"
 Android:layout_height="0dp"
 Android:layout_weight="1.0"

Android:text="@string/quest_description" />

<TableLayout

Android:layout_width="match_parent"
 Android:layout_height="0dp"
 Android:layout_weight="1.0"
 Android:background="#ffffff"
 Android:shrinkColumns="*"
 Android:stretchColumns="*" >

<TableRow

Android:layout_width="fill_parent"

Android:layout_height="wrap_content"
 Android:gravity="center_horizontal"

>

<TextView

Android:id="@+id/textView4"

Android:layout_width="wrap_content"

Android:layout_height="wrap_content"

Android:layout_weight="1"
 Android:text="@string/scale1" />

<TextView

Android:id="@+id/textView5"

Android:layout_width="wrap_content"

Android:layout_height="wrap_content"

Android:layout_weight="1"

Android:text="@string/scale2" />

<TextView

Android:id="@+id/textView6"

Android:layout_width="wrap_content"

Android:layout_height="wrap_content"

Android:layout_weight="1"

Android:text="@string/scale3" />

<TextView

Android:id="@+id/textView7"

Android:layout_width="wrap_content"

Android:layout_height="wrap_content"

Android:layout_weight="1"

Android:text="@string/scale4" />

<TextView

Android:id="@+id/textView8"

Android:layout_width="wrap_content"

Android:layout_height="wrap_content"

Android:layout_weight="1"

Android:text="@string/scale5" />

</TableRow>

<RadioGroup

Android:layout_width="fill_parent"

Android:layout_height="wrap_content"

Android:orientation="horizontal" >

```

<RadioButton
    Android:id="@+id/radioButton1"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center" />
<RadioButton
    Android:id="@+id/radioButton2"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center" />
<RadioButton
    Android:id="@+id/radioButton3"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center" />
    </RadioGroup>
</TableLayout>
</LinearLayout>
    <RadioButton
    Android:id="@+id/radioButton4"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center" />
    <RadioButton
    Android:id="@+id/radioButton5"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center" />
    </RadioGroup>
</TableLayout>
</LinearLayout>
    
```

The optimization goal of the fitness function

Best solutions are those MUIs that meet the user requirements with the less number of XML elements and attributes, so the optimization goal of this problem will tend to minimize.

$$minf(x, y) = \sum x + \sum y$$

A second version of the sample code above that meets the same user requirements but with a lower fitness will provide same functionality to the end user. Based on that, a sample second version of the five grade scale *mobile user interface* is showing below together with its fitness function to validate this optimization approach, in this new one the fitness is 60.

$$f(x, y) = 50 \text{ (elements)} + 10 \text{ (properties)} = 60$$

Below code is the second version of sample code above of the MUI produced.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
  xmlns:tools="http://schemas.Android.com/tools"
  Android:layout_width="match_parent"
  Android:layout_height="match_parent"
  Android:alpha="1.0"
  Android:orientation="vertical" >

<!-- Comments Line Sample -->

<TextView
  Android:id="@+id/textView1"
  Android:layout_width="fill_parent"
  Android:layout_height="0dp"
  Android:layout_weight="1.0"
  Android:gravity="left"
  Android:text="@string/instructions"
  Android:textColor="@color/black" />

<TextView
  Android:id="@+id/textView2"
  Android:layout_width="fill_parent"
  Android:layout_height="0dp"
  Android:layout_weight="1.0"
  Android:text="@string/quest_description" />

<RadioGroup
  Android:layout_width="299dp"
  Android:layout_height="wrap_content"
  Android:orientation="vertical" >

  <RadioButton
    Android:id="@+id/radioButton1"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:checked="false"
  
```

```

        Android:text="@string/scale1" />

<RadioButton
    Android:id="@+id/radioButton2"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center"
    Android:text="@string/scale2" />

<RadioButton
    Android:id="@+id/radioButton3"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center"
    Android:text="@string/scale3" />

<RadioButton
    Android:id="@+id/radioButton4"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center"
    Android:text="@string/scale4" />

<RadioButton
    Android:id="@+id/radioButton5"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_weight="1"
    Android:gravity="center"
    Android:text="@string/scale5" />

</RadioGroup>
</LinearLayout>

```

Each FM object has a property called fitness which stores the fitness scalar value, this is implemented on FeatureModel class.

As a short overview, the fitness of a FM in terms of this work is defined by the product with the minor number of elements and properties that accomplish all the user

requirements, then the optimization objective will be to have FMs that depict families of MUIs with less XML code.

The selection operators

Two selection techniques are used in this work to choose the best fit individuals for reproduction purposes, both have proved to be efficient in literature [1], Rank-Based and Tournament Selection strategies. MAPDS tool implemented sexual reproduction which selects two parents for reproduction.

The Selection() method on ATM class depicts their implementation, the java code is below.

```
public FeatureModel Selection(int op) {
    switch (op) {
        /***** rank-based roulettewheel *****/
        case 1: {
            int i = 1;
            double sum=0.0,summy=0.0;
            for (int j=0; j<=sizePopulation; j++)
                sum = sum + population[j].getFitness();
            summy = population[i].getFitness() / sum;
            double ro = Math.random();
            while (summy < ro) {
                i++;
                summy = summy + population[i].getFitness() / sum;
            } //end while
            return population[i];
        } //end case 1

        /***** Tournament selection based *****/
        case 2: {
            FeatureModel best, next;
            best = population[(int) Math.random() * sizePopulation];
            for(int i=0;i<=10;i++) {
                next = population[(int) Math.random() * sizePopulation];
            }
        }
    }
}
```

```

        if (best.getFitness() < next.getFitness())
            best = next;
        }
        return best;

    } //end case 2

} // end switch
return null;
} //end Selection method

```

As illustrated before, rank-based selection uses the rank ordering of fitness values to determine the probability of selection. Therefore, the selection process is separated of the actual fitness values, with the advantage that the best individual will not dominate in the selection process [11].

By other hand, tournament selection compares the performance of the selected individuals picked randomly and the best one from this group is returned by the operator. For crossover with two parents, the selection method is executed twice, once for the selection of each parent [15].

Crossover

The `crossOver()` method in class `ATM` implements *sexual* reproduction. Once the parents are selected one point of crossover is chosen, so that, the point of crossover is common for both parents. Code below shows how the arrays are combined using the left part of parent A and the right part of parent B, this approach was introduced by ETHOM [1] for crossing features models chromosomes, the approach was adopted by MAPDS as a starting point. Code below illustrates the implementation on MAPDS.

```

public void crossOver (int op) {
    FeatureModel offSpring;

    /** Select Individual for CrossOver */

    FeatureModel parentA = Selection(1);
    FeatureModel parentB = Selection(1);

```

```

offSpring = parentA;

/***** CroosOver Point *****/
int crossOverPoint = (int) (Math.random() * (features - 1));

/***** CroosOver *****/
for(int i=crossOverPoint;i<features;i++) {
    offSpring.setChildren(i,parentB.getChildren(i));
    offSpring.setRelation(i,parentB.getChildren(i));
}

offSpring.printFeatureModel(60);

} // end crossover function
    
```

Mutation

The main goal of the mutation is to introduce new genetic material into the population [15]. Mutation is implemented on MAPDS using two of the four customized operators introduced by ETHOM [1], only operators 1 and 2 that mutate the FM transversal- tree array were incorporated in MAPDS. Constraints are not mutated. The mutation operators were implemented on Mutation() method in ATM class. Below code shows the implementation.

```

public void Mutation (FeatureModel FM) {
    switch((int) (Math.random() * 3)) {
        // set a new relation
        case 1: {
            int actualRelation=FM.getRelation((int)Math.random()* features),newRelation=0;
            while(actualRelation == newRelation) {
                newRelation = (int) (Math.random() * 3);
            }
            FM.setRelation(actualRelation,(int) Math.random() * features);
        } //end case 1

        // It changes randomly the children number

        case 2: {
            int pos = (int) (Math.random() * features);
    
```

```

FM.setChildren((int) Math.random() * features,(int) Math.random() * features);      } //
end case 2

    // It change CTC type.
        case 3: FM.changeCTCType();          --> Change
        case 4: FM.changeCTCPoint();
    } // end switch

} // end mutation method

/// Cases 3 & 4

public void changeCTCType() {
    if (CTCMatrix[1][0] == 0)
        CTCMatrix[1][0] = 1;
    else
        CTCMatrix[1][0] = 0;
}
public void changeCTCPoint() {
    int newPoint = (int) (Math.random() * nodes);

    while (CTCMatrix[1][1] == newPoint) {
        newPoint = (int) (Math.random() * nodes);
    }

    CTCMatrix[1][1] = newPoint;
}
}

```

Infeasible individuals

The repairing approach proposed by ETHOM [1] was implemented on MAPDS in order to provide support for the correctness of FMs that depict families of MUIs. Then individuals are repaired to turn them on feasible individuals, it means to avoid a chromosome to be semantically redundant.

Briefly Operator 1 replaces relationships (or- and alternative) with a single child by optional relationships. Operator 2 changes randomly the number of children of a node. Operator 3 changes the type of constraint in CTC array. Operator 4 changes randomly the source and destiny nodes on CTC array. Chapter 2 gives a full introduction to these operators.

Survival

Finally, the offspring considers a replacement policy by which individuals can be inserted into the population. The approach is replacing the worst parent with the breeding, if this one has a better fitness.

Configurator Class

The Configurator class is in charge of automatic derivation of products by using an ad-hoc approach. The FM chromosome represents a set of feature combinations (product configurations) based in the existence or not of a component, in consequence, a product can be represented by a binary array and therefore an FM is a family of binary arrays, figure 20 illustrates this concept.

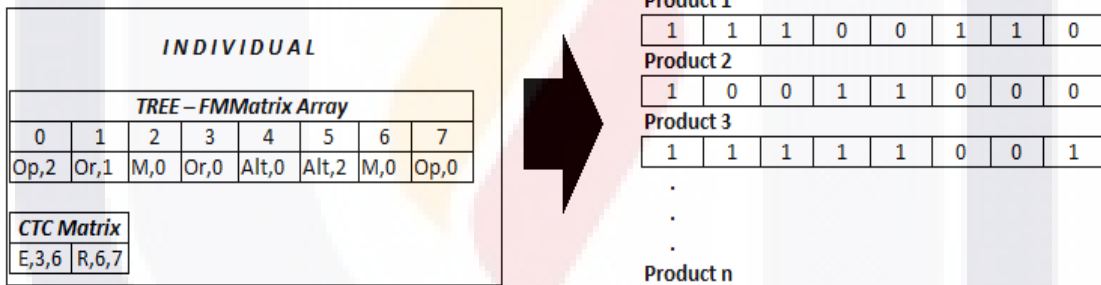


Figure 20. A FM Individual and its derived products

So, a universe of possible products is found on the binary table comprised by the n combination of products. For sample on figure 20, 126 possible products will be gotten from this FM individual. For automatic product derivation, Configurator() class will test each product in the pool of possible products (the binary table) to determine those combinations that belongs to the family in question. This cycle is repeated for each FM produced. Once the real products are found this class checks the resultant combination complies the requirements needed to build the XML code and once this is done it prints the XML file (MUI).

In this chapter, it was introduced a method for automatic product derivation of *mobile user interfaces* on XML language descriptor. A software prototype that implements the methods used for automatic derivation based on evolutionary computation. In the next chapter, results gathered from the tool are showing to demonstrate the validity of the model proposed.



Chapter 5. Results and Evaluation

This chapter presents the methodology used to evaluate the approach proposed for automatic produce derivation in the context of this research and its analysis of results.

5.1 Evaluation

The research methodology is based on a quantitative approach by automatically producing solutions that improve the number of lines in XML programs. The solutions produced by the MAPDS tool are totally usable and were successfully compiled using the `adt-bundle-windows-x86_64-20130522` [31] provided by the Android community in their portal. It was evaluated the optimization objective according to the fitness function proposed.

The efficiency of this research was measure by the quality of the solutions provided in terms of usability. Each solution responded to those requirements described in the study case section in chapter 3. The MAPDS tool and the solutions produced by the experiments were added as a complement of this document on electronic files.

All runs were performed on a computer laptop Toshiba brand with 6GB RAM and Intel Core i3 processor to 1.90 Ghz.

Next paragraphs describe the two rounds of experiments to create these solutions, basically, the experiments try to prove if the solutions improved the fitness, and therefore, they minimized the number of code lines required.

Experiment 1 – Verification of the correctness of the MUI at the initial populations.

The goal of this experiment is validating the generation of *mobile user interfaces* that can be successfully compiled on ADT through MAPDS tool in order to validate the automatic derivation method that was implemented in the `Configurator()` class. Thus, the experiment consists of running the initial population and validating the XML products

produced by compiling the XML codes in ADT.

In this experiment, it is tested the XML files obtained from all of the FMs in the initial population, in order to determine the number of well-formed products that can be produced randomly, and finding the best fitness at this stage of the process for later check into the second experiment if the fitness is improved after using crossover and mutation techniques.

Method -Experiment 1

The method is to run five tests and find the biggest number of products produced and the best fitness. The table 5 shows the results gathered from the run of test which are discussed ahead.

Table 4. Results from experiment 1

Execution#	Parameters		Results				Results
	Initial population	Features	Best Fitness	5-grade	3-grade	Android tags from table 4	
1	4,000	20	0	0	0	57	None MUI was produced
	16,000	30	0	0	0	10	None MUI was produced
2	300	10	0	0	0	5	None MUI was processed
3	300	20	47	9	266	2	275 products (MUIs)
4	300	30	0	0	0	20	Lack of memory
5	300	50	0	0	0	5	Lack of memory

From this experiment 275 MUIS were produced using a randomly approach, the best fitness gotten is 8. It can be seen of the total of the Android tags suggested on table 4 could not be processed with the actual computing resources. So the experiment is effective only using two tags <TextView> and <RadioButton>.

Experiment 2 – Verification of the correctness of the MUI after mutation and crossover.

Products gathered from experiment 1 are the result of random combinations techniques, for this reason in experiment 2, it is validated the optimization approach using mutation and crossover parameters in MAPDS tool. The expected result is to get a lower fitness and a bigger number of products.

Method -Experiment 2

Five executions of the process are tested using the same number of products from experiment 1 and parameters in table below.

Table 5. Results from experiment 2

IT M #	Selection strategy	Mutation probability	Executions Number	Infeasible individuals	Fitness	Products
1	Roulete-wheel	0.005	2000	Repairing	42	312
2	Roulete-wheel	0.0075	5000	Replacing	42	175
3	Tournamnent	0.005	2000	Repairing	42	423
4	Tournamnent	0.0075	5000	Replacing	42	231

From experiment 2, it can be deducted that fitness is not improved due this is the smallest fitness which can be obtained for code in the products of our study case for 3-grade and 5-grade scale, it means 3 text view labels and 3 or 5 radio buttons.

5.2 Analysis of Results

A MUI gathered from experiment 2 is showed below to illustrate the approach. However the number of products increase due the EA parameters which is a good result, this will result in a bigger number of option for the final user.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="84dp"
        android:gravity="left"
        android:text="@string/Text X"
        android:textColor="@color/black" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_marginTop="60dp"
        android:gravity="left"
        android:text="@string/Text X"
        android:textColor="@color/black" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="31dp"
        android:gravity="left"
        android:text="@string/Text X"
        android:textColor="@color/black" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="44dp"
        android:layout_marginTop="156dp"
        android:checked="false"
        android:text="@string/scale X" />

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

android:layout_marginLeft="25dp"
android:layout_marginTop="62dp"
android:checked="false"
android:text="@string/scale X" />

<RadioButton

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="37dp"
android:checked="false"
android:text="@string/scale X" />

</RelativeLayout>

After a second look into the results from experiment 2, it was found that more than one product has a similar fitness despite of being different between them, a sample is shown if figure 21 below. From this result, it can be found the problem is deriving to a multimodal optimization due this is not a unique optimization goal. A sample of that are two MUIs that are different only in the color of text.

FITNESS = 32

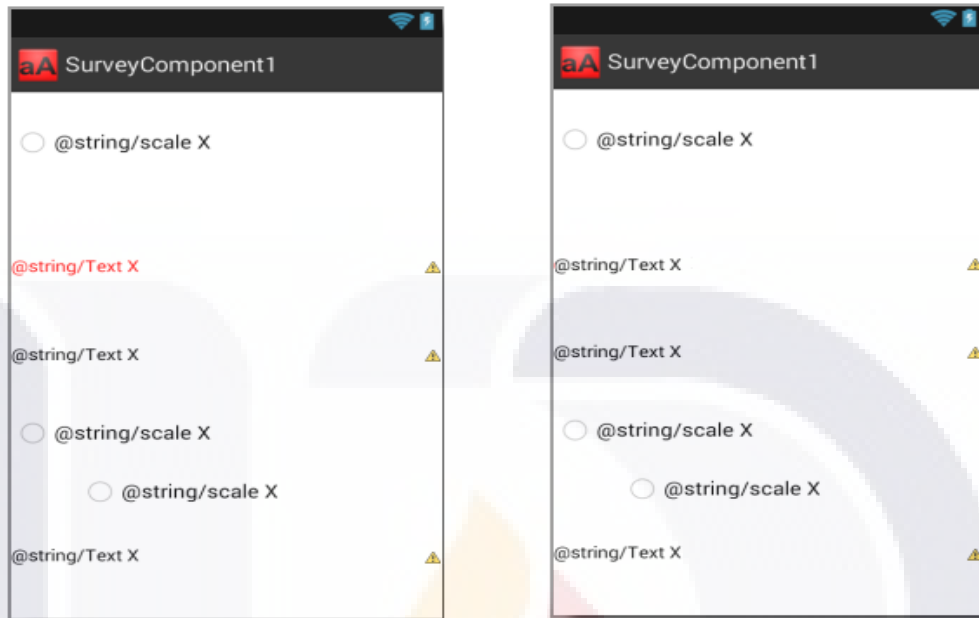


Figure 21. Sample of MUIs with similar fitness and different XML code.

As a result of these experiments, the correctness and accuracy of this model can be proved by measuring the total of products that can be re-used and that were obtained from production of a specific domain field exposed at the study case section. These requirements accomplished those needed to build a MUI for a survey question.

CONCLUSIONS

This dissertation proposed an *automation concept* in terms of AIT for SPL to model the problem it was proposed a study case which is based on the domain of the online survey applications to outline the requirements of a Production Line Software under this context.

Evolutionary computation was used for automatic product derivation purposes using as reference the ETHOM algorithm in the context of automatic generation of feature models. However, fitness function on ETHOM measures the feature model not of the derived products, for this reason a new fitness function was introduced by this work to measure the quality of the products derived. As a result of this, a total of 475 products from scratch were successfully produced by using our prototype, many of them with same fitness which derives the problem to a multimodal optimization field as a future work.

The objective to prove AIT in the context of automation in SPL was successfully reached and the objective to provide with applications to the ETHOM algorithm as well. We claim as future work, additional analysis to build these innovative mobile collection data interfaces through practicing of software production lines. More characteristics of *mobile user interface* can be generated by the MAPDS tool through this approach. The visualization of survey data view is generally adequate to produce MUI re-usable by using XML code. On this basis, and taking into consideration the basic user requirements, we proposed an automatic method to do it.

GLOSARY

Automation concept

Refers to a general approach of automation in software product line that will provide a paradigm.

Automatic derivation of mobile user interfaces

Refers to the process of coding user interfaces for applications running in Android platform with artificial intelligence mechanisms.

Mobile Android Product Derivation Software (MAPDS)

Software tool for automatic derivation of mobile user interfaces that implements evolutionary computation.

Mobile device

Refers to an electronic device like a smartphone or tablet.

Mobile user interface

A software user interface in XML code which is reusable at applications running in Android operating system.

Mobile domain

In terms of this research, mobile domain refers to the landscape of all those requirements to build applications running in Android operating system.

REFERENCES

- [1] S. Segura, J. Parejo, H. M. Robert, D. Benavides y A. Ruiz-Cortés, «ETHOM: An Evolutionary Algorithm for Optimized Feature Models Generation (v. 1.3),» Applied Software Engineering Research Group & Universidad de Sevilla, Sevilla, 2013.
- [2] K. Schmid, E. Rommes y F. J. Van DerLinden, *Software Production Lines in Action*, New York: Springer, 2007.
- [3] H. Ammar, W. Abdelmoez y M. Salah Hamdi, «Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems.,» de *International Conference on Communications and Information Technology (ICCIT-2012)*, Tunisia, 2012.
- [4] D. Parnas, "On the design and development of program families," *IEEE transactions on software engineering*, Vols. SE-2, no. 1, pp. 1-9, 1976.
- [5] Software Community, «Framework for Software Product Line Practice, Version 5.0,» 2014. [En línea]. Available: http://www.sei.cmu.edu/productlines/frame_report/index.html. [Último acceso: 21 May 2014].
- [6] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak y A. S. Peterson, «Feature-Oriented Domain Analysis (FODA) Feasibility Study,» Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1990.
- [7] Android Community, "The android open source project site," Android Community, 2014. [Online]. Available: <http://developer.android.com/guide/topics/ui/index.html>. [Accessed 2014].
- [8] D. Benavides, A. Felfering, J. A. Galindo y F. Reifrank, «Automated Analysis in Feature Modelling and Product Configuration,» Springer-Verlag, Berlin Heidelberg,

2013.

- [9] D. Benavides, S. Segura y A. Ruiz-Cortéz, «Automated analysis of featuremodels 20 years later: A literature review,» *Information Systems*, vol. 1, p. 615–636, 2010.
- [10] S. Russel y P. Norving, *Artificial Intelligence A Modern Approach*, 3rd ed., New Jersey: 2009, 2010.
- [11] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed., Indianapolis, IN 46256: John Wiley & Sons, 2007, pp. 127-142.
- [12] C. Darwin , *On The Origin of Species try Mrans of Natural Selection*, London: John Murray, 1859.
- [13] G. Mendel, «Experiments on Plant Hybridization,» de *Natural History Society of Brünn*, Brünn, Czech Republic, 1866.
- [14] J. B. Lamarck, *Philosophie zoologique*, Paris, Francia: Oxford University, 1809.
- [15] A. E. Eiben y J. E. Smith, *Introduction to Evolutionary Computing*, Germany: Springer-Verlaga Berlin Heidelberg, 2003.
- [16] A. Touring, *Intelligent machinery*, London: Oxford University Press on behalf of the Mind Association, 1959.
- [17] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Michigan: University of Michigan Press, 1975.
- [18] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan y D. B. Shmoys, *The Traveling Salesman*, The university of michigan: John Wiley and Sons, 1985.
- [19] X. Jiafu, «Software Automation: from "Silly" to "Intelligent",» de *International conference on tools with AI*, Arlington, VA, 1992.
- [20] M. Harman, «The Role of Artificial Intelligence in Software Engineering,» de *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, 2012 *First International Workshop on*, Zurich, Switzerland, 2012.
- [21] R. E. Lopez-Herrejon, J. A. Galindo, D. Benavides, S. Segura y A. Egyed, «Reverse engineering feature models with evolutionary algorithms: an exploratory study,» de

4th International Symposium, SSBSE, Riva del Garda, Italy, 2012.

- [22] S. Soltani, M. Asadi, M. Hatala, D. Gasevic y E. Bagheri, «Automated Planning for Feature Model Configuration based on Stakeholders' Business Concerns,» de *ASE 2011*, Lawrence, KS, USA, 2011.
- [23] S. Runyu, G. Jianmei y W. Yinglin, «A Preliminary Experimental Study on Optimal Feature Selection for Product Derivation Using Knapsack Approximation,» IEEE, Shanghai 200240, China, 2010.
- [24] S. Soltani, M. Asadi, M. Hatala, Gasëvic y E. Bagheri, «Toward automated feature model configuration with optimizing non-functional requirements,» Elsevier, Toronto, Canada, 2013.
- [25] W. Zhang, H. Zhao y Z. Jin , «Mining binary constraints in the construction of feature models,» de *20th IEEE International Requirements Engineering Conference (RE)*, Chicago IL, USA, 2012.
- [26] E. Rosa, R. F. De Lucerna, V. C. Cordeiro y J. E. Chaves, «Dynamic and Automated Product Derivation for Consumer Electronics Software Applications,» de *Springer-Verlag*, Berlin Heidelberg, 2013.
- [27] E. Cirilo, U. Kulesza, M. Torres y C. Lucena, «Experience with Automatic Product Derivation of Mobile Applications Using Model-Driven Techniques,» de *The Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications*, Canada, IRMA, 2012, p. 11.
- [28] H. Gomaa y M. E. Shin, «Automated Software Product Line Engineering and Product Derivation,» de *40th Hawaii International Conference on System Sciences*, Hawaii USA, 2007.
- [29] D. Mellado, J. Rodriguez, E. Fernández-Medina y M. Piattini, «Automated Support for Security Requirements Engineering in Software,» de *Availability, Reliability and Security, 2009. ARES '09.*, Fukuoka Japan, 2009.
- [30] S. Miranda , H. Mariano, U. Kulesza y T. Batista, «Automating Software Product Line

- Development: A Repository-Based,» de *Software Engineering and Advanced Applications (SEAA)*, 2010 36th EUROMICRO Conference on, Lille, France, IEEE.
- [31] Android Community, «ADT Plugin,» Android Community, 2014. [En línea]. Available: <http://developer.android.com/tools/sdk/eclipse-adt.html>. [Último acceso: 2014].
- [32] R. Jain, J. Bose y T. Arif, «Contextual Adaptive User Interface For Android Devices,» de *IEEE India Conference (INDICON)*, India, 2013.
- [33] World Bank, Information and Communications for Development, Washington D.C: The World Bank, 2012.
- [34] J. Bethlehem y S. Biffignandi , Handbook of Web Surveys, New York, USA: John Wiley & Sons, Inc., 2011.
- [35] SurveyAnyplace, «SurveyAnyplace,» SurveyAnyplace, [En línea]. Available: <https://surveyanyplace.com/why-mobile-surveys/>. [Último acceso: March 2015].
- [36] C. Burger, R. Valentin, J. Grafeneder, B. Woisetschläger, D. Vidovic y A. Hergovich, «Reaching the Mobile Respondent : Determinants of High-Level Mobile Phone Use Among a High-Coverage Group,» *Social Science Computer Review*, n° 2009, 2010.
- [37] R. Johns, LIKERT ITEMS AND SCALES, University of Strathclyde, Glasgow, United Kingdom: SURVEY QUESTION BANK, 2010.
- [38] T. Thüm, D. Batory y C. Kästner, «Reasoning about edits to feature models,» de *International Conference on Software Engineering*, Vancouver Ca, 2009.
- [39] C. Prins, «A simple and effective evolutionary algorithm for the vehicle routing problem,» *COMPUTERS AND OPERATIONS RESEARCH*, vol. 21, p. 2004, 1986.
- [40] C. Burger, V. Riemer, B. Grafeneder, D. Woisetschläger, D. Vidovic y A. Hergovich, «Reaching the Mobile Respondent: Determinants of High-Level Mobile Phone Use Among a High-Coverage Group,» 2010.
- [41] J. Fawcett, L. Quin y D. Ayers, Beginning XML, Indianapolis, IN 46256: John Wiley & Sons, 2012, pp. 3-23.

ANNEXS

ANEX A: Article Desarrollo de software basado en el paradigma de líneas de producción: caso de estudio software de fidelidad para PyMEs.

ANEX B: Article Service Oriented Architecture Proposal for a Mobile Survey Platform.

ANEX C: Article Arquitectura basada en patrones de diseño para la producción textual colaborativa

ANNEX A

Article

Desarrollo de Software basado en el paradigma de Líneas de Producción: Caso de Estudio Software de Fidelidad para PyMEs

CONISOFT 2013


CONISOFT 2013 (author)
Help Log out

My Submissions
CONISOFT 2013
EasyChair

CONISOFT 2013 Submission 21

[Update authors](#)
[Withdraw](#)

If you want to **change any information** about your paper or withdraw it, use links in the upper right corner.
 For all questions related to processing your submission you should contact the conference organizers. [Click here to see information about this conference.](#)
 All **reviews sent to you** can be found at the bottom of this page.

Paper 21

Title:	Desarrollo de Software basado en el paradigma de Líneas de Producción: Caso de Estudio Software de Fidelidad para PyMEs
Submission	
Author keywords:	Líneas de producción de software ingeniería de requerimientos PyME
Abstract:	Este trabajo presenta el resultado de la re-utilización obtenida mediante el proceso de análisis de requerimientos de una Línea de Producción de Software (LPS) para aplicaciones de marketing de fidelización basado en tarjetas. A través del uso del paradigma de la ingeniería de líneas de producción de software podemos promover productos de bajo costo y buena calidad dirigidos a pequeñas y medianas empresas (PyMEs) que representan el sector empresarial más grande del país, dado que este paradigma propone una alternativa a las costosas implementaciones ad-hoc de sistemas informáticos. Numerosos casos de estudio en la literatura documentan los beneficios del desarrollo del ciclo de vida de la ingeniería de líneas de producción de software, sin embargo no encontramos aquellos enfocados al contexto de las PyMEs en el mercado mexicano. Por lo que este trabajo aporta también un caso de estudio enfocado al análisis de la ingeniería de requerimientos de las líneas de producción de software para las PyMEs mexicanas.
Time:	Apr 28, 17:46 GMT

Authors						
first name	last name	email	country	organization	Web site	corresponding?
Catalina	Calderon	ccalderon62@hotmail.com	Mexico	Universidad Autónoma de Aguascalientes	http://www.uaa.mx	✓
Francisco	Alvarez	fjalvar@correo.uaa.mx	Mexico	Universidad Autónoma de Aguascalientes	http://www.uaa.mx	

Reviews

Desarrollo de Software basado en el paradigma de Líneas de Producción: Caso de Estudio Software de Fidelidad para PyMEs

Catalina Calderón García, Francisco J. Alvarez
Unidad de Ciencias Básicas, Universidad Autónoma de Aguascalientes
ccalderon62@hotmail.com, fjalvar@correo.uaa.mx

Resumén

Este trabajo presenta el resultado de la re-utilización obtenida mediante el proceso de análisis de requerimientos de una Línea de Producción de Software (LPS) para aplicaciones de marketing de fidelización basado en tarjetas. A través del uso del paradigma de la ingeniería de líneas de producción de software podemos promover productos de bajo costo y buena calidad dirigidos a pequeñas y medianas empresas (PyMEs) que representan el sector empresarial más grande del país, dado que este paradigma propone una alternativa a las costosas implementaciones ad-hoc de sistemas informáticos.

Numerosos casos de estudio en la literatura documentan los beneficios del desarrollo del ciclo de vida de la ingeniería de líneas de producción de software, sin embargo no encontramos aquellos enfocados al contexto de las PyMEs en el mercado mexicano. Por lo que este trabajo aporta también un caso de estudio enfocado al análisis de la ingeniería de requerimientos de las líneas de producción de software para las PyMEs mexicanas.

Abstract

This paper presents the results of the re-use obtained through the requirements analysis process of Software Production Lines (LPS) for loyalty marketing applications. By the usage of software production lines engineering, inexpensive and good quality products can be promoted for small and medium enterprises (PyMEs), which represent the largest business sector in the country, due this paradigm proposes an alternative to costly ad-hoc implementations.

Even, numerous cases of study in the literature document benefits of the development life cycle engineering software product lines, we do not find those focused on the context of PyMEs in the Mexican market. So this paper provides a case study also

focused to software requirements engineering of software production lines for Mexican PyMEs.

1. Introduction

En México más del 90% de las empresas son Pequeñas y Medianas Empresas conocidas como PyMEs¹ las cuales generan más del 70% de los empleos del país. Estas enfrentan a grandes retos para su supervivencia entre ellos la dotación de infraestructura informática adecuada. Dada su naturaleza las PyMEs no pueden lidiar con costosas y largas implementaciones, que suponen los desarrollos adhoc, dado que estas empresas cuentan con poco personal y presupuestos limitados.

El paradigma de Líneas de producción de Software (LPS) puede proporcionar una respuesta a la demanda de software en éste sector, ya que a través de mecanismos de re-utilización se puede producir software a bajo costo y basado en componentes (software modular). Así una empresa no tendría que instalar una solución completa sino sólo aquello que necesita y a medida que crece incrementar su base de software instalada. Otro beneficio importante es que la PyME se ve beneficiada del dominio de conocimiento que la LPS aborda, ya que el software producido por la LPS además de presentar un espectro de soluciones para el dominio incrementa su conocimiento a través de las lecciones aprendidas de otros proyectos, en este aspecto los sistemas adhoc son limitados y aún más en el área de las PyMEs que no pueden pagar por grandes grupos de especialistas que puedan analizar sus procesos y dar recomendaciones.

Así iniciamos nuestro estudio haciendo un poco de historia y presentando el contexto de éste interesante paradigma.

El término de líneas de producción de software fue introducido completamente en la década de 1990 como resultado de la exploración en el concepto de *centrarse*

¹ FUENTE: INEGI <http://www.inegi.org.mx/>

en un dominio específico como una base para desarrollar activos reutilizables. Una de las primeras contribuciones fue la descripción del método de Análisis Orientado a Características de Dominio conocido por sus siglas en siglas como FODA[1]. La diferencia fundamental entre el desarrollo tradicional de un sistema único ad-hoc y LPS es un cambio fundamental de enfoque que implica sobre todo un cambio en la estrategia de desarrollo: desde la reunión de negocios estratégica del próximo contrato ad-hoc a una visión estratégica de desarrollo de software enfocado a un campo de negocio. La ingeniería de líneas de producción de software se basa en una distinción fundamental entre el desarrollo para la reutilización y la idea del desarrollo con reutilización de los sistemas tradicionales. La reutilización abarca todos los activos que son relevantes al completo desarrollo del ciclo de vida del software. La clave del paradigma de LPS es el proporcionar herramientas para el manejo y disminución de la variabilidad de la reutilización cuando nuevos requerimientos del cliente son generados.[2]

2. Definición del problema y contribución

Los beneficios de la ingeniería de líneas de producción de software son muchos y bien documentados por importantes casos de estudio en todo el mundo y todo tipo de organizaciones, de los cuales para efectos de nuestro trabajo es importante destacar: (1) un abaratamiento de los costos de fabricación de los productos de software dado el enfoque de desarrollo para reutilización a partir del desarrollo de tres aplicaciones[2], (2) mejora de la calidad del producto final debido a procesos de testing más exhaustivos y (3) una mayor calidad de los requerimientos ya que estos no se obtienen de una reunión con un cliente específico sino de un campo de dominio de productos lo que hace que se vean beneficiados de lecciones aprendidas de otros proyectos similares. Sin embargo a pesar de los beneficios, los retos que implica la creación de líneas de producción de software también son muchos un enfoque mal aplicado del paradigma puede conducir a generar demasiados activos reutilizables que no sean aprovechados, o que se le de una complejidad gratuita a un producto dado el amplio campo de requerimientos que pudiera tener un dominio, sumando otros factores, como el extenso marco metodológico y la compleja estructura per sé de los productos de software hacen que no abunden las líneas de producción de software.

Sin embargo, la creencia que dada la baja complejidad funcional de la familia de productos de

software enfocados a PyMEs a desarrollar pudiera facilitar la implementación de la línea de producción, dió motivación a este trabajo donde nuestra contribución consiste en este primer análisis de los requerimientos de una línea de producción de software para el desarrollo de software de fidelidad basado en tarjetas con el objetivo de poner en el futuro al alcance de las pequeñas y medianas empresas, software a bajo costo.

3. Análisis del requerimiento de la línea de producción de software

La ingeniería de líneas de producción de software comprende dos ciclos de vida: la ingeniería de dominio y la ingeniería de aplicación. La primera es el ciclo de vida del que resultan los activos comunes que juntos forman la plataforma de la línea de producción. Esta también es responsable del alcance de la línea de producto y asegura que la plataforma tiene la variabilidad que es necesaria para soportar el alcance de productos necesario dentro de la estrategia de mercado planeada para la administración del producto final, para lograrlo esta se vale de la ingeniería de requerimiento de dominio. [2]

Por otro lado, el ciclo de vida de análisis de requerimientos de aplicación, especifica un producto particular, donde los requerimientos vienen de un grupo de diferentes beneficiarios de este producto. En este punto el análisis de variabilidad y comunalidad es una herramienta útil para elicitar este tipo de requerimientos[2] y poder determinar las partes que puede obtenerse por reuso y aquellas que deban desarrollarse por otros medios.

De tal manera que el análisis de requerimientos de la línea de producción de software lo determina los requerimientos del dominio y los requerimientos del producto particular a producir, ya que los segundos retroalimentan a los primeros para ampliar el espectro de problemas que aborda la línea de producción de software.

3.1 Análisis de requerimientos de la ingeniería de dominio.

La ingeniería de requerimientos del dominio, representa el proceso de crear y administrar los requerimientos para la arquitectura de referencia y su implementación a través del seguimiento de cinco fases básicas: *elicitación, documentación, negociación, validación y verificación y administración.* [2]

El corazón de nuestro trabajo lo constituye la aplicación de las fases 1 y 4 para el desarrollo del modelo de dominio y el diagrama de característica de la línea de producción propuesta. Las fases 2,3 y 5 se llevaron a cabo de manera cíclica pero son obviadas en éste artículo para concentrarnos en la descripción del modelo.

FASE 1 – Elicitación de requerimientos. La elicitación de los requerimientos dentro de la ingeniería de dominio consiste en el proceso de recabar las necesidades de los usuarios y otros involucrados en el proceso de análisis y diseño para desarrollar una plataforma que contruye un amplio número de productos de una misma familia.

El descubrimiento y la explotación sistemática de elementos comunes en todos los sistemas relacionados a la familia es un requisito técnico fundamental para lograr con éxito la reutilización de software. *La metodología de Análisis de Dominio Orientada a la Característica* y conocida por sus siglas en inglés como FODA[2], es una técnica que se puede aplicar para cumplir con este requisito. Mediante el examen de una clase de sistemas de software relacionados y la teoría subyacente común de esos sistemas, el *análisis de dominio* puede proporcionar un modelo de referencia para la descripción de la clase. Puede proporcionar también una base para la comprensión y la comunicación sobre el espacio del problema abordado por el software en el dominio y en última instancia puede proponer un conjunto de enfoques arquitectónicos para la implementación de nuevos sistemas. El método para ejecutar un análisis de dominio consiste de tres etapas: *análisis del contexto*, que establece los límites del dominio, *modelado del dominio*, que describe el espacio de problemas que son abordados por el dominio y *modelado de la arquitectura*, que crea la arquitectura de software que implementa una solución para cada problema en el dominio. [1]

Para iniciar este estudio el alcance del contexto esta delimitado por el marketing de fidelización basado en tarjetas, que se define dentro de la literatura[3] como un enfoque de marketing de lealtad que hace uso de una tarjeta para todas las transacciones necesarias en el programa de lealtad. El marketing de fidelización se define como parte de la gestión estratégica, en la que una empresa se centra en el crecimiento y retención de los clientes existentes a través de incentivos o recompensas.[3]

El diseño del dominio objetivo de nuestro trabajo, se basó en seis tipos de escenarios de programas de fidelización basados en tarjetas que fueron obtenidos

de la literatura, [4] los cuales se detallan a continuación.

a.- Simple sistemas de puntos acumulados. Este es el método más común de programa de fidelización. Los clientes frecuentes ganan puntos por compra, que se traducen en algún tipo de recompensa. A estos puntos se tiene acceso mediante la tarjeta del cliente. Si se trata de un descuento, un regalo de promoción o un trato especial al cliente, los clientes trabajan para una cierta cantidad de puntos por la cual canjear su premio.

b.- Sistema de niveles para recompensar la lealtad inicial y alentar a más compras. Encontrar un equilibrio entre los beneficios alcanzables y deseables es un reto para la mayoría de empresas que implementan programas de fidelización. Una manera de combatir esto es poner en práctica un sistema escalonado. Ofrecer pequeñas recompensas como una oferta base para formar parte del programa y animar a los clientes, aumentando el valor de los beneficios moviéndose hacia arriba en la escala de fidelidad también ayuda a obtener recompensas en cortos períodos de tiempo.

c.- Cargar una cantidad por adelantado para obtener beneficios VIP. Este refiere a aquellos programas que dan beneficios adicionales a los que clientes que pagan una cuota al inicio del programa o algún tipo de servicio prepago.

d.- Sistemas de doble-precio. Estos programas proporcionan precios especiales a aquellas personas que adquieren una tarjeta o la obtienen a través de una recompensa.

e.- Asociarse con otra empresa para prestar servicios todo incluido. Son programas que animan a alianzas estratégicas para incrementar la base de clientes y su lealtad, ya que una empresa puede ofrecer beneficios más allá de su propia capacidad poniendolo por delante de competidores en el mismo sector.

De la síntesis de estos escenarios determinamos los requerimientos de software específicos del dominio el resultado se presenta en la Figura 1 y a continuación mostramos el contexto de los conceptos presentados en el diagrama.

Tarjeta. La tarjeta funciona como el medio de identificación del cliente, contiene un código que identifica al clientes a través del proceso. Se dividen en tarjetas

VIP, aquellas que mediante una cuota inicial anual proporcionar servicios o bienes preferenciales al cliente.

Precargo, aquellas que compras servicios a precios preferenciales por adelantado.

Standard, identifica al cliente sólo como parte de un programa de fidelización.

Motor de análisis de datos. Representa un modelo de análisis de datos estadísticas que ayuda a definir las tendencias que dirijan los nuevos programas de recompensas.

Compra. Indica el proceso de compra.

Puntaje. El total de puntos acumulados por cliente.

Recompensa. Indica una recompensa que se obtiene cuando el cliente cumple ciertas reglas asociadas a un programa.

Catálogo. Identifica el tipo de la recompensa.

Monetarias, que regresan un valor a través de descuentos, precios preferenciales, etc.

No monetarias, éstas son aquellas asociadas a un programa benefico o que dan servicios al cliente como estatus, etc.

ServiciosVIP, éstas se obtienen a través de alianzas estratégicas con otras empresas.

Programa. Determina el tipo de programa de fidelidad basado en los escenarios anteriormente descritos.

Reglas. Indican las reglas de operación asociadas al programa.

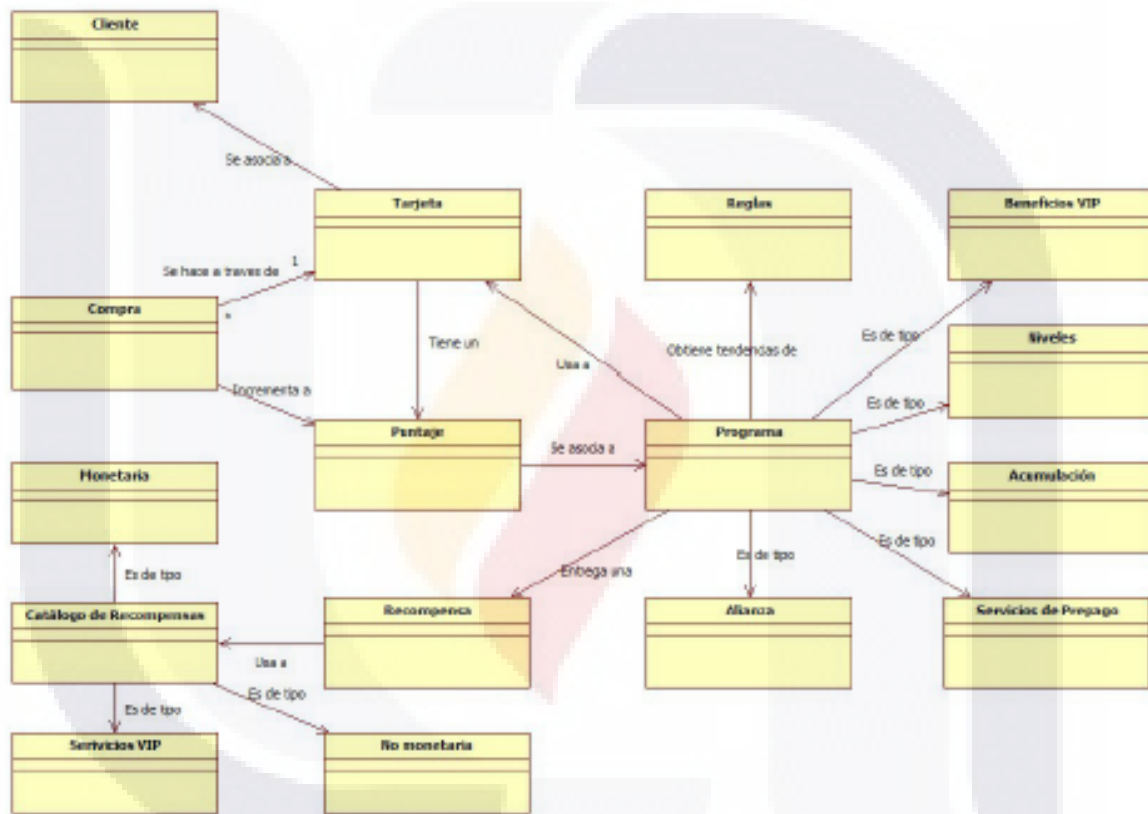


Figura 1. Diagrama de dominio -Marketing de Fidelización

FASE 4 – Validación de los requerimientos del dominio. La validación del requerimiento se realizo a partir del análisis en la Figura de donde obtenemos el diagrama de característica conocido en la metodología FODA por sus siglas en inglés *"feature model"* de la Figura 2 el cual tiene como objetivo la comprensión del espacio de requerimientos de la línea de producción de software de fidelización basado en tarjetas.

Hasta aquí hemos definido un conjunto de características que darán forma a los diferentes productos para la familia de software de nuestra línea de producción a la que se le dio el nombre *"SPL Fidelización"*

En las siguiente secciones, mostramos el requerimiento de la aplicación específica de nuestro caso de estudio, y hacemos el análisis de variabilidad y

comunalidad para determinar el grado de reutilización del la aplicación solicitada como resultado del estudio.

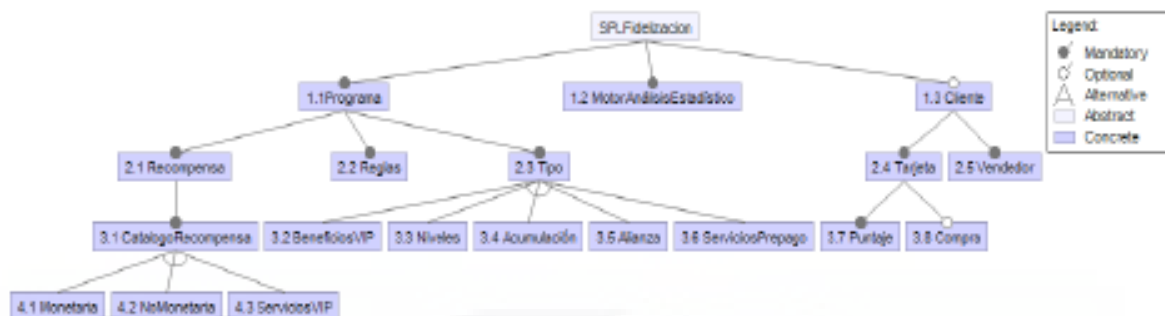


Figura 2. Diagrama de características de la familia de software de fidelidad orientado a tarjetas

3.2 Análisis de requerimientos de aplicación

La segunda parte del análisis de requerimientos de la línea de producción corresponde al análisis de los requerimientos de la aplicación o producto de software a desarrollar, para la obtención de requerimientos con el usuario final/cliente.

Nuestro caso de estudio lo conforma la PyME Cartune Fuel Injection y su necesidad de un sistema de cómputo para el control de su reciente programa de fidelización de clientes basado en tarjetas.

Cartune Fuel Injection se dedicada a la reparación y mantenimiento de vehículos automotores con una base aproximada de 300 clientes, como parte de la ampliación de su oferta de servicios pondrá al alcance de sus clientes una tarjeta de protección de viaje, con la cual ellos podrán adquirir servicios prepagados de protección en el camino a precios preferenciales y a su vez tener acceso a una línea de promociones para clientes distinguidos conforme a cada compra. El estado de cuenta de dicha tarjeta será consultado a través de la página WEB que ahora se encuentra en línea en www.car-tune.com.mx. Car-tune cuenta con

una plantilla no mayor a 10 trabajadores en un solo establecimiento en la ciudad de Aguascalientes México.

Una de las etapas más difíciles durante este proceso de análisis de requerimientos fue la conceptualización del producto de software requerido por la PyME, ya que la base de requerimientos proporcionada por la PyME es muy escueta, lo cual es de esperar, la PyME no cuenta con recursos para poder inventir en un estudio de marketing. Otro problema es que no contamos con una línea de producción previa, por lo que no tenemos activos para reutilizar. Es así que el reto es crear una familia de productos que deberá tener una visión de crecimiento futuro a través de la experiencia de la historia de producción, pensando en una infraestructura que podrá ser moldeada conforme a la experiencia de la adaptación de nuevos productos dentro del dominio. En la siguiente sección se muestra el análisis de requerimientos y como se abordó la problemática anteriormente expuesta.

, este caso de estudio se presentó en la sección 3. Se aplicaron una serie de entrevistas que dieron como resultado la funcionalidad esperada del sistema de acuerdo a las expectativas de la PyME en la Tabla 1.

Tabla 1. Requerimientos funcionales y de sistema de la aplicación de software del caso de estudio

#	Requerimiento Funcional	#	Requerimiento del software
1	Activar y desactivar tarjetas preimpresas con un código de barras para identificación del cliente y sus movimientos en el sistema.	1.1	Crear una función para dar de alta la tarjeta en el sistema a través de leer el código de barras en ella.
		1.2	Crear una función para dar de baja la tarjeta y evitar que se siga usando. El historial de compras del cliente se mantiene como estadística en la base de datos del sistema. Desactivar la tarjeta en el sistema a través de leer el código de barras en ella.
2	La compra de la tarjeta deberá contener un o	2.1	Crear un módulo para activar servicios prepagados a la

	varios servicios y/o artículos de acuerdo al monto de prepago. Ejemplo: Servicio de Grúa ó 10 litros de Gasolina.	2.2	Deberá crearse en el sistema una interfaz para la creación y actualización de los servicios, artículos y promociones que participan en los diferentes programas.
		2.3	Si se hace una compra el sistema abona un puntaje a la cuenta del cliente de acuerdo al tipo de servicio.
3	Agregar nuevas promociones en el software según la conveniencia.	3.1	Crear un módulo para ingresar nuevas promociones
4	Los clientes deberán poder ver el estado de sus tarjetas a través del sitio de la compañía.	4.1	Agregar una sección de consulta en la página web del sitio para solicitar un reporte con el historial del cliente, usando el código de barras de la tarjeta.
5	Enviar una felicitación de cumpleaños al cliente que será recibida en su celular.	5.1	El sistema deberá enviar un mensaje vía celular al cliente el día de su cumpleaños con o sin promoción.

De la misma forma se produjo un conjunto de restricciones del software que se presenta en la tabla 3.2.

Tabla 2 Restricciones

#	Requerimiento del producto de software
R1	El sistema no debe imprimir las tarjetas.
R2	No deberá haber tarjetas sin servicio prepago asociado.
R3	Los servicios prepago no asignan puntos a la tarjeta.
R4	Las promociones deberán tener un tiempo de caducidad.

En la siguiente sección, las Tablas 3.1 y 3.2 son reinterpretadas en términos del espectro de requerimientos de la línea producción que mostramos en la Figura 2, a fin de determinar la funcionalidad que que puedes ser absorbida por el modelo de la línea de producción de software y determinar el porcentaje de reutilización.

6. Resultados

Como resultado de este trabajo obtuvimos un diagrama de característica -Figura 2- usando la metodología FODA. El cual produce una familia de software de fidelidad orientado a tarjetas. La validez y exactitud de este modelo la comprobamos midiendo el grado de re-utilización que se obtiene a partir de la producción de una aplicación específica, este requerimientos se explicó en la sección 5. De tal forma que un análisis de comunalidad y variabilidad entre el dominio y la aplicación, en la Tabla 1 podemos ver .

Los primeros tres requerimientos están cubiertos por las características 1,2,3,4,4 y 4.1, de tal forma que un

submodelo de característica para la aplicación del caso de estudio se presenta en la Figura 3. De los 8 requerimientos del caso de estudio 6 son cubiertos por el espacio de requerimientos de la línea de producción de software, dos son características muy ad-hoc al PyME, obtuvimos un porcentaje de re-utilización del 75% de los requerimientos del caso de estudio.

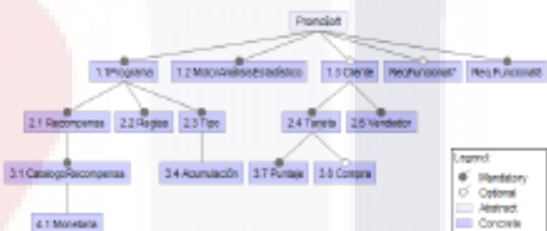


Figura 3. Diagrama de características de PromoSoft

7. Conclusiones y trabajo futuro

Este primer ensayo propone la conceptualización de los requerimientos de una Línea de Producción de Software destinada a la fabricación de productos de software de fidelidad, nuestro caso de estudio esta basado en una PyME a fin de demostrar que dicho paradigma enriquece las aplicaciones orientadas a éste sector como fue nuestra posición inicial. Las PyMEs se ven beneficiadas del conocimiento proporcionado por el dominio subyacente en el campo de requerimientos, por otro lado las empresas que fabrican software para este sector pueden verse beneficiadas por la reducción de los costos de desarrollo que comprueba la literatura. [2]

Hasta aquí pudimos demostrar que existe un porcentaje de funcionalidad cubierta por la línea en el puro análisis del requerimiento de la sección de resultados pero habrá que hacer una segunda

demonstración a partir del diseño de las arquitectura asociadas por lo que como trabajo futuro deberemos diseñar la infraestructura de la plataforma para hacer la implementación de producción de componentes. En un tercer estudio, será interesante ver la codificación sobre algún tipo de plataforma abierta y orientada a componente, para mantener siempre el enfoque de costos bajos que las PyMEs requieren, como podría ser Javabeans™.

Algunos conceptos que quedan también abiertos a la experimentación son el de la calidad, ya que al ser esta una línea de producción tal vez enfoques que se utilizan en otras líneas de producción de manufactura pueden ser retomados como el de Total Quality Management (TQM), el cual pueden ser aplicado a través de la relación que puede darse entre re-utilización y el ciclo de vida del control de la calidad.

10. Referencias

[1] Kyo C, Sholom G, James A, William E, Spencer A, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Technical Report CMU/SEI-90-TR-21 ESD-90-TR-222. November 1990.

[2] F.J. Linden, K. Schmid, E. Rommes, *Software Production Lines in Action*, Springer, New York, 2007.


[3] E. Stuart. (2007) "No Such Thing as Loyalty", The financial times LTD , London, 2007.

[4] B. Wolf, *Loyalty Marketing -The second art-*, Teal Books, Greenville,2001.

ANNEX B

Article

Service Oriented Architecture Proposal for a Mobile Survey Platform CONISOFT 2013


CONISOFT 2013 (author)
[Help](#) [Log out](#)


My Submissions
CONISOFT 2013
EasyChair

CONISOFT 2013 Submission 22

[Update authors](#)
[Withdraw](#)

If you want to **change any information** about your paper or withdraw it, use links in the upper right corner.
 For all questions related to processing your submission you should contact the conference organizers. [Click here to see information about this conference.](#)
 All **reviews sent to you** can be found at the bottom of this page.

Paper 22

Title:	Service Oriented Architecture Proposal for a Mobile Survey Platform
Submission	
Author	Mobile Data Collection
keywords:	Service Oriented Architecture Movil Devices Architecture
EasyChair keyphrases:	mobile survey platform (221), mobile user interface (190), service oriented architecture (142), mobile device (140), web service (120), functional requirement (115), user interface (110), non functional requirement (110), survey form (100), mobile survey (100), survey configurator service (95), soa mobile survey platform (60), data collection (56), mobile client (50), mobile phone (50), mobile data collection (47), soap based web service (40)
Abstract:	Mobile Data Collection or Mobile Surveys is a growing trend of data collection where a survey form, application or collection tool is on a mobile device such as a Smartphone or a Tablet. By using an online platform data gathered can be retrieved and processed. This data collection trend is growing due low costs and short times needed for applying a survey compared to traditional methods, which are being replaced by web surveys rapidly. So that, the first online commercial surveys platforms that provides mobile capabilities and a total customer business integration through SOA are appearing into market in order to deliver on-line survey services. In this context, our contribution is to provide a Service Oriented Architecture (SOA) for an On-line Survey Data Collection Platform through mobile computing.
Time:	Apr 28, 17:53 GMT

Authors

first name	last name	email	country	organization	Web site	corresponding?
Catalina	Calderon	ccalderon62@hotmail.com	Mexico	Universidad Autónoma de Aguascalientes	http://www.uaa.mx	✓
Francisco	Alvarez	fjalvar@correo.uaa.mx	Mexico	Universidad Autónoma de Aguascalientes	http://www.uaa.mx	✓

Service Oriented Architecture Proposal for a Mobile Survey Platform

¹Catalina Calderón, ²Francisco Álvarez, ³Angel Muñoz
Depto. Ciencias Básicas, Universidad Autónoma de Aguascalientes
Aguascalientes, México
ccalderon62@hotmail.com
jmauaa@gmail.com
aemz@correo.uaa.mx

Abstract

Mobile Data Collection or Mobile Surveys is a growing trend of data collection where a survey form, application or collection tool is on a mobile device such as a Smartphone or a Tablet. By using an online platform data gathered can be retrieved and processed. This data collection trend is growing due low costs and short times needed for applying a survey compared to traditional methods, which are being replaced by web surveys rapidly. So that, the first online commercial surveys platforms that provides mobile capabilities and a total customer business integration through SOA are appearing into market in order to deliver on-line survey services. In this context, our contribution is to provide a Service Oriented Architecture (SOA) for an On-line Survey Data Collection Platform through mobile computing.

1. Introduction

According to the World Bank in 2012, there are about six billion mobile subscriptions in use worldwide, about three quarters of the world's people now have access to a mobile phone. In some developing countries, more people have access to a mobile phone than a bank account, electricity or even clean water. Mobile communications now provide great opportunities to advance human development, from basic access to information and education health to make cash payments to encourage participation of citizens in democratic processes.[2] Along with the technological mobile base growth, the increasing of functionality of Smartphone devices in recent years have given rise to an explosion in developing applications for this sector. One sample of that are Mobile Survey Platforms which take advantages of the

on line data treatment too. Online (Internet) surveys are becoming an essential research tool for a variety of research fields, including marketing, social and official statistics research. According to ESOMAR¹, online survey research accounted for 20% of global data-collection expenditure in 2006.[3]

The methods involved in survey data collection are any of a number of ways in which data can be collected for a statistical survey. These are methods that are used to collect information from a sample of individuals in a systematic way. First there was the change from traditional paper-and-pencil interviewing (PAPI) to computer-assisted interviewing (CAI). Now, face-to-face surveys (CAPI), telephone surveys (CATI), and mail surveys (CASI, CSAQ) are increasingly replaced by web surveys.[1] The mobile devices offer innovative ways to gather data regardless of time and location of the respondent. Apart from the high mobile phone penetration, further advantages are quicker response times and the possibility to reach previously hard-to-reach target groups.[4] An advantage of using a mobile device for data collection is to gather information more discreetly if needed. For example, an observation survey could be performed on a smart phone or tablet instead of a clipboard or stack of survey papers.

2. Problem definition and contribution

The novel online apps scenarios led by cloud computing and web services are pushing organizations to claim for a total business services and technology integration, where surveys provider platforms can be integrated into business workflows in order to enhance survey operation and quality information. Samples of

¹ The European Society for Opinion and Market Research (ESOMAR) is the world association for market, social and opinion researchers

that are commercial platforms like “CheckBox”, “Question Pro” and “MExplorer” which offer a survey business portfolio that includes mobile and web services under a Software Oriented Architecture (SOA).

Since SOA-Mobile survey platforms have appeared in recent years, none architecture model has been proposed by which designers and architects can take advantage in order to start new endeavors or a model that can be improved by lessons learned. So that, this work proposes a Service Oriented Arch for a Mobile Survey which provides extended facilities in comparison to those proposed by commercial vendors, addressing the problem of a rich mobile user interface replacing the regular use of a web browser user interface in the mobile device by a XML based-on client which consumes native functions, what will be provide an enhanced user experience. By isolated the mobile user interface facilities, it can be abstracted on a component that can be produced under software production lines approach.

In next section, we introduce requirements gathered from a basic on-line survey scenario,[2] which we have added our mobile functionality model over the mobile client side, in order to provide the mobile data collection introduced previously. Requirements have been divided into functional & non-functional so that we can describe them clearly on the arch design proposed on section 4.

3. Functional & non-functional requirements

In mobile survey context, two basic roles are taking place (1) a surveyed, who is the person or group of people that solved a survey form in the mobile phone, (2) surveyor, who creates and transmits a survey form, specific functional requirements are written on table 1.



Fig. 1. Mobile survey platform functional requirements

Table 1. Functional requirements overview

#	Description
FR 1	Surveyed must run a client application over a mobile device.
FR 2	The mobile client application must provide the look and feel of a native user interface.
FR 3	The mobile user interface should request to record surveyed identification data on a central repository.
FR 4	The survey form on the mobile client must be customizable via a web service and it should be stored on a central repository.
FR 5	Surveyor should to be provided of a survey formatting tool accordingly.
FR 6	Any formatting tools should be accessed via WEB browser.
FR 7	A mechanism for authentication against the database should be provided.

A subsystem model on Figure 2 was developed based on functional requirements on Table 1, in order to recognize the functional units, as software components, which will help to identify views and services that our arch should deliver.

From functional requirements four subsystems have been defined, each one will depict on a component by itself that will be included on our SOA mobile survey platform.

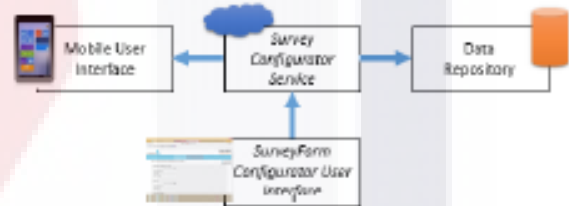


Fig. 2. Mobile survey platform functional subsystems

The objective and foundation of every one component is shown below and its relations and interfaces are shown in detail on section 4.

1. *Mobile User Interface (MUI)*. It represents a mobile user interface view component that interacts with the surveyed to display the survey form to be solved. Functional requirements FR1, FR2 and FR3.

2. *Survey Configurator Service (SCS)*. Represents a web service in SOA terms that provides the functionality requested to create a new survey format. It interacts with the *SurveyForm Configurator User Interface*. Functional requirements FR4, FR5.

3. *SurveyForm Configurator User Interface (SFCUI)*. It represents an HTML view that is presented to the surveyor to interact with the *Configurator Service*. Functional requirements FR6 and FR7.

4. *Data Repository*. It contains the survey data. Functional requirements FR3 and FR4.

Following the requirements analysis process to build a software architecture, non-functional requirements are introduced on Table 2. They meet those needs that according to Sommerville[5] may affect the overall architecture of a system rather than the individual components, inside this context, Sommerville have classified non-functional requirements in three main lines: "product requirements", "organizational requirements" and "external requirements", for a practical effect over our SOA arch we have decided working "product requirements" only, and left organizational and external non-functional requirements for future implementations of our arch model on real life.

Table 2. Non- functional requirements overview

Requirement	Description
Performance	The response of the mobile user interface should not exceed 15 seconds in each transaction.
Design constraints	The solution must be scalable under the scale-up strategy (Add more resources to the server) initially and then scale-out (add more servers) as processing needs. The solution should have low coupling and the ability to easily edit the parameters that are considered dynamic and require frequent changes.
Availability	In the event of component failure, there should be no loss of information. The solution should include transactional consistency requirements. Given the failure of the application, there must be mechanisms that include disruption of transactions for these completed

4. Service oriented architecture for a mobile survey platform

Service Oriented Architecture (SOA) is an architectural paradigm emergent that allows to organizations share resources as services across different business processes and its technological domains, under a low coupling approach. In this context applications are modularized, and every single piece operates as resource by exposing an interface of capabilities that can be consumed for internal or external requestors. This is possible due the adoption of standard protocols, XML based, like Simple Object

	correctly. Having an authentication and authorization mechanism against the single database of identities (standard LDAP v3), on which has a directory service in which all users will be consolidated.
Security[6]	Having a mechanism registration / de-registration of web services, limiting the search of services to those registered by authorized entities. Having an authentication mechanism between the service provider and the APPLICANT, services limited to using only known sources Having a legitimizing mechanism requests by authorized entities Having a mechanism alerts in case of failures. Include in the program code mechanisms to trap errors in transactions and send them to a central log of the functionality.
Maintenance	Within the "release document" integrate preventive maintenance activities. The solution should provide a mechanism to reuse the different components
Portability	The solution must provide a mechanism for re-use. Must provide a platform for growth of the installed base in the same operation scheme. The solution must provide a mechanism for growth of the business areas.
Escalability	Growth should include requirements for internal and external users.

Non-functional requirements also helped us to address another SOA technology definitions and mobile device limitations which are exposed in next section.

Access Protocol called SOAP, which addresses petitions through message which pass between different hardware and software resources. Some advantages of SOA includes code re-use and infrastructure, third party components consumption, software and hardware as a service and security increasing thanks to a layered architecture.

A mobile data collection platform SOA is remarkable in order to pursuing a common framework that resumes the diversity of technologies involved in Mobile Platforms and address another limitations in mobile devices, as non-intense computational tasks, limited memories, storages and screen sizes which

could take advantages of the distributed computational resources over the network.

The SOA for a mobile survey platform introduced on this work was created on a SOAP-based web services approach and since data needs to be gathered from different sources, intermediate layers are involved to create a component based framework according to SOA principles. Figure 3 shows the interaction between layers and depicts our SOA for a mobile survey platform.

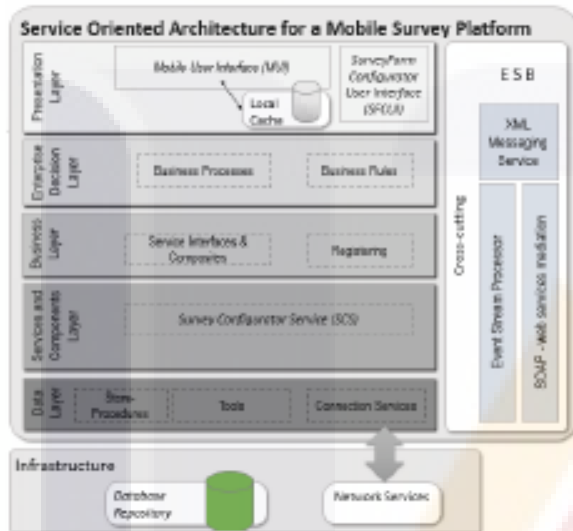


Fig. 3. Service oriented architecture for a mobile survey platform

Five layers along the architecture with an Enterprise Service Bus (ESB) have been defined in order to meet with SOA design principles and specific functional and non-functional requirements of a mobile survey platform on section 3, as well. The goal and foundations of every layer is explained in detail below.

4.1 Presentation layer

The presentation layer keeps the user interface components at the top of the platform. According to our component approach on section 3, two different types of user interfaces are requested on our mobile survey platform, a *Mobile User Interface (MUI)* and a desktop configuration view.

The component of *Mobile User Interface (MUI)* encapsulates whole functionality requested to show a survey form over a mobile device, once the survey form is completed, data are sent back to feed the central database. This MUI comprises two subcomponents an XML code view and a java service called Local Cache that resides on the mobile device.

Both components interact independently so that they can be re-used in different XML platforms, Figure 4 depicts components interplay.

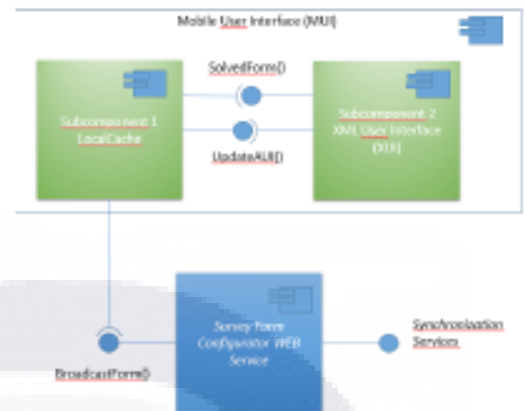


Fig. 4. MUI component model

The method `UpdateAUI()` updates the view XML code into the mobile device. So that, a new survey form created through of the *Survey Configurator Service (SCS)* is sent to mobile device by putting the view XML code as part of the SOAP message, and once it is received by the mobile local cache it extracts the view code from the message and run `UpdateAUI()` to refresh.

The `SolvedForm()` method works to extract the XML code produced once the survey form is solved by the surveyed. The *Mobile Local Cache* component is in charged to send via a SOAP message the XML code to feed the data layer.

The mobile client *LocalCache* module ensures that when more data than requested arrive, they are not rejected. It acts as a filtering module, though without discarding what was not requested. This data is stored locally and can be used for example for off-line usage or simply to avoid a new communication session with the application server.[7] It is free to use every platform, which supports the deserialization of soap messages, namely the mapping from XML structures to the supported object model.

4.2 Enterprise decision layer

Enterprise decision layer is part of SOA core characteristics. This layer contains rules and processes business at the top level that make up the enterprise applications, for this work this layer is depicted over the model but none workflow or rule has been created given the generic nature of the model. Deployment is reserved for real life scenarios.

4.3 Business layer

The business layer contains more coarse-grained services that consist of two or more individual components placed in composite files.[8] For our purpose, a composite file is created to expose the *Survey Configurator Service (SCS)* in order it can provide with the functionality needed to create new survey forms and sending them to the mobile devices. The contents of this file are shown below.

```
<composite
  xmlns...

  <component name="SurveyConfigurator">
    <implementation...
    <service name="SurveyConfigurator">
      <binding ws uri=
        "http://localhost:8085/
        SurveyConfigurator "/>
    </service>
  </component>
</composite>
```

Another usage of the business layer is to comprise services interfaces/contracts. A contract is the complete specification of a service between a service provider and a specific consumer which is a core SOA characteristic [8] given the open service oriented motivation of SOA.

4.4 Services and components layer

Under a SOA model, Mobile User Interface (MUI) and Survey Configurator Service (SCS) do not communicate each other because of the loose coupling, then an ESB (Enterprise Service Bus) interacts between them, thanks to ESB, components can be re-used, figure 7 shows component interaction for our SOA mobile survey platform.



Fig. 6. Component Communications using ESB

4.5 Data layer

General speaking, a mobile application solution does not used to interfere with data layer. All available data sources normally consists of a data warehouse with the corresponding Database servers. Nevertheless, it is possible to define some operations in server, in order to achieve results, relevant with business analysis. The corresponding computations can be performed either by an external source or directly using database query code, or by the creation and publication of new services in the application server. Generally speaking, this layer should remain as transparent as possible to the rest of the architecture.[7]

Achieving SOA requires more than SOAP-based web services. The characteristics of SOA transcend a particular technology. SOA is an amalgamation of technologies, patterns, and practices, the most important of which were addressed by our proposed, so as, the mobile survey platform requirements as well. In the next section, this is put on practice through a prototype based on mobile devices Android and open source software.

5 Prototype

The correctness and validity of the SOA proposed was checked by creating a prototype on table 3. This prototype is made on open source software following the guideline presented on the study conducted on [8].

Given our mobile survey platform, let's recap the process of starting a Service Component Architecture (SCA) domain/server, receiving an inbound web service request for our exposed service. Table 3 illustrates the steps requested to messaging between Survey Configurator Service (SCS) and Mobile User Interface (MUI).

Table 3. Description of the steps requested to messaging

Step	Description
1	The LaunchSurveyServer class starts the SCA domain/container by using the "embedded" server. It instantiates the server by specifying the composite XML file used, which in this case is called mobilesurvey.composite.
2	The SCADomain class, which is part of the Apache Tuscany implementation, is used to start the embedded Jetty server. In turn, this is used to host the web service that's being exposed by the assembly's service element.
3	A web services client initiates a Configurator request against the hosted web service using the dynamically generated WSDL

	<p>created by the web service binding associated with the service element defined in the composite. The web service SOAP request might resemble the following:</p> <pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/s oap/envelope/" xmlns:chap="http://host/mobilesurvey"> <soapenv:Header/> <soapenv:Body> <chap:createTicket> <arg0> MUI code... </arg0> </chap:createTicket> </soapenv:Body> </soapenv:Envelope></pre>
4	The inbound request is received by the SCA embedded server, which then delegates the processing of the request to the component implementing the service, ConfiguratorImpl.
6	The request is processed and the results are spread to the mobile clients via ESB.

SOAP over HTTP was chosen as messaging protocol over our SOA arch due its characteristics of extensibility (WS-security and routing are extensions applied in development), neutrality (SOAP can be used over any transport protocol such as HTTP, SMTP, TCP or JMS) and independence of programming language. In this way SOAP allows to lead our service to be integrated with other business and mobile architecture which was one of our main goals in order to justify a SOA platform for Survey Mobile Platform.

7. Conclusions and future work

In this work, we have introduced a Service Oriented Architecture for a mobile survey platform that emphasizes on the decomposition of the mobile user interface which can be re-used. The visualization of survey data view is generally adequate to produce MUI re-usable by using XML code. On this basis, and taking into consideration the basic user requirements, we proposed a service oriented architecture, based on the SOAP-Web Services technology.

We claim as future work, additional analysis to build these innovative mobile collection data interfaces through practicing of software production lines. More characteristics of mobile user interface can be generated by the Survey Configurator Service (SCS)

through this approach. A security analysis over the communications model should be performed too as future work.

This SOA platform alleviates the problems stemming from the currently highly heterogeneous domain of mobile survey platforms, but which at the same time also takes into consideration the constantly evolving characteristics of mobile devices in this field.

8. Referencias

- [1] J. Bethlehem, S. Biffignandi, *Handbook of Web Surveys*, John Wiley & Sons, New Jersey, 2012, ISBN 978-1-118-12172-6.
- [2] World Bank, 2012 "Information and Communications for Development", The World Bank, Washington D.C, 2012, pp 3-4.
- [3] V. Vehovar, K. Lozar Manfreda, "Overview: Online Surveys". *The SAGE Handbook of Online Research Methods*. London: 2008, pp. 177-194. ISBN 978-1-4129-2293-7.
- [4] C. Burger, V. Riemer, J. Grafeneder, B. Woisetschlager, D. Vidovic, A. Hergovich, "Reaching the Mobile Respondent: Determinants of High-Level Mobile Phone Use Among a High-Coverage Group". *Social Science Computer Review*. SAGE, Viena, 2010.
- [5] I. Sommerville, "Software Engineering Ninth Edition", Addison-Wesley, International Edition, 2011, ISBN-13: 978-0-13-703515-1, ISBN-10: 0-13-703515-2
- [6] D. Cotroneo, A. Graziano, S. Russo, "Security Requirements in Service Oriented Architectures for Ubiquitous Computing", *2nd Workshop on Middleware for Pervasive and AdHoc Computing*, Toronto, Canada, 2004.
- [7] I. Michalarias, V. Koppen, "Towards a Service Oriented Architecture for Mobile Reporting", *In Proceedings of International Conference for Intelligent Systems 2005*, Kuala Lumpur, Malaysia, 2005.
- [8] J. Davis, "Open Source SOA", Manning Publications Co, Greenwich, CT 06830, 2009, ISBN 978-1-933988-54-2

ANNEX C

Article

Arquitectura basada en patrones de diseño para la producción textual colaborativa
CCITA Cancún 2014

UT Cancún

ccita Conferencia Conjunta Iberoamericana sobre Tecnologías y Aprendizaje

UCLM Universidad de Castilla-La Mancha

La Universidad Tecnológica de Cancún
Organismo Descentralizado del Gobierno del Estado de Quintana Roo

Otorga la presente

CONSTANCIA

a

Jaime Muñoz Arteaga, Catalina Calderón, Francisco Alvarez Rodríguez y Ángel Muñoz Zavala

por haber participado como autor (es) con el trabajo titulado

"Arquitectura Basada en Patrones de Diseño para la Producción Textual Colaborativa"

Durante la V Conferencia Conjunta Iberoamericana sobre Tecnologías y Aprendizaje
CciTA 2013.
Cancún, Quintana Roo, México a 6 de septiembre de 2013

[Signature]
Dr. Manuel E. Prieto Méndez
Coordinador del Comité Internacional CciTA 2013
Universidad Castilla- La Mancha

[Signature]
M. en D. Leslie Angelina Hendricks Rubio
Rectora
Universidad Tecnológica de Cancún

Logos: Raambal, Universidad del Caribe, UNIVERSIDAD POLITÉCNICA DE QUINTANA ROO, De la Salle, Universidad La Salle Cancún, UTM, Riviera Maya, UNID, turnitin, spdece, CAACECA

Arquitectura Basada en Patrones de Diseño para la Producción Textual Colaborativa

Catalina Calderón¹, Jaime Muñoz², Francisco Álvarez³
*Depto. Ciencias Básicas, Universidad Autónoma de Aguascalientes
Aguascalientes, México*

[¹ccalderon62@hotmail.com](mailto:ccalderon62@hotmail.com)

[²jmauaa@gmail.com](mailto:jmauaa@gmail.com)

[³fjalvar@correo.uaa.mx](mailto:fjalvar@correo.uaa.mx)

Resumen. La etapa de implementación de los Sistemas Colaborativos se apoya de diversas herramientas y marcos de desarrollo en general orientados a objetos, pero no existe hasta ahora una manera unificada para especificar y diseñar aplicaciones sobre los mismos. Esto hace que el diseño de tales sistemas o plataformas sea impreciso debido a la falta de un modelo arquitectónico que permita representar los elementos principales de las plataformas de colaboración. El paradigma de patrones de diseño por su parte ofrece una metodología probada para mejorar el diseño de un sistema. De esta forma, este trabajo propone una arquitectura basada en patrones de diseño para la plataforma de producción textual colaborativa.

KeyWords: producción textual colaborativa, groupware, arquitectura colaborativa.

1 Introducción

La producción textual colaborativa es una modalidad de trabajo que se basa en la actitud altruista de los colaboradores con el fin de contribuir a incrementar y mejorar el conocimiento. El concepto puede ser aplicado a cualquier proceso de creación textual, ya sea literatura o proyectos grupales, discusión en foros, etc.

La etapa de implementación de las plataformas colaborativas regularmente se apoya de diversas herramientas y marcos de desarrollo en general orientados a objetos, pero no existe hasta ahora una manera unificada para especificar y diseñar aplicaciones sobre las mismas. Esto hace que el diseño de tales sistemas o plataformas sea impreciso.[2][3] Por otro lado, un patrón de diseño es producido para resolver un problema o encontrar una respuesta en alguno ya existente y probado reduciendo el costoso camino de programar desde cero [4], y por lo tanto puede dar una respuesta contundente a los problemas de diseño y especificación de los sistemas orientados a la producción textual colaborativa.

En este sentido la contribución de este trabajo consiste en una arquitectura basada en patrones de diseño que ofrece facilidades extendidas en comparación a las ya existentes ya que además de buscar una mejora en el diseño de la plataforma se toma como base de requerimiento funcional la metodología propuesta por LATIN Project dando como resultado la inclusión de procesos y métodos no existentes en las plataformas actuales, y que son el resultado de las mejores prácticas en producción colaborativa hasta este momento. En las siguientes secciones se muestra el panorama

de procesos de la metodología de LATIN Project y como estos se transforman en un conjunto de características “atributos” que entretujan los requerimientos funcionales que dan paso a la arquitectura textual colaborativa propuesta bajo este enfoque.

2 Análisis de los requerimientos funcionales

De acuerdo con la propuesta de marco de trabajo del LATIN Project una iniciativa de la Unión Europea para generar producción colaborativa de textos en Latinoamérica, el proceso de producción debería incorporar cuatro diferentes dimensiones metodológicas, [1] estas son: *funciones, procesos, control de documentos y estrategia de escritura* con el fin de producir un libro. La generación de contenidos a su vez se descompone en dos sub-dimensiones: una que conduce a la creación de un libro completo (Book-Driven Process), y otra que conduce a la creación de piezas atómicas o pequeñas de contenido (Content-Driven Process) que podrán o no formar parte de un libro.

El enfoque “Book-Driven process” se centra en tres procesos base: la formación del grupo (lo que incluye la asignación de tareas y roles), la certificación de la obra y la publicación, considerando previamente la generación de piezas individuales de contenidos como una tarea cíclica incluida dentro del enfoque orientado a la creación de libros. Tomando como base la dimensión de procesos, las funciones provistas por la plataforma para soportar a la tarea (proceso) y a su actor se analizan en la Tabla 1.

Tabla 1. Actividades y roles propuestos para la plataforma

“Rol”	“Actividades”	“Atributo en Plataforma”	“Permisos”
<i>Define quien es el usuario</i>	<i>Define las responsabilidades del rol y por lo tanto las actividades que un usuario desempeña</i>	<i>Instancia que referencia de manera abstracta las actividades</i>	<i>Define el nivel de acceso a contenidos</i>
Productor de Contenidos	Crea contenido sobre un tema específico	AgregarContenido()	Contenido Propio
Generador de la idea	Dirige el proyecto y coordina la idea general del contenido	VerContenido()	Contenido propio y de todo el grupo
Revisor	Revisor de Contenido: Propone modificaciones al contenido Revisor Técnico: Revisa aspectos técnicos de la obra Revisor de idioma: revisa el libro para asegurarse de su sintaxis, ortografía y gramática	AgregaRevisión() EliminaRevisión() ModificaRevisión() VerContenido()	Contenido propio
Organizador	Organizador de Contenido: Organiza los contenidos Organizador de grupo: Coordina las actividades del grupo en conjunto con el generador de la idea	AgregarContenido() EditarContenido() EliminaContenido() VerContenido()	Contenido propio y de otros autores

Diseñador de plantillas	Interfaz: desarrolla y mantiene el look&feel de la plantilla del libro Pedagógica: Organiza la tabla de contenido de manera pedagógica	EditaFormato() ModificaFormato() EliminaFormato() InsertaTablaContenido() EditaTablaContenido() EliminaTablaContenido()	Contenido Propio
Acreditador	Valida el libro en base a un estándar	VerContenido()	Ve todos los contenidos
Traductor	Traduce contenido de autores cuya lengua nativa es diferente a la de los contenidos	TraduceContenido() VerContenido()	Ve todos los contenidos

La columna “atributo en plataforma” de la tabla refiere a un subsistema que encapsula funciones específicas que dan soporte a un proceso de la metodología “Book Driven Process”, de tal forma que ocultando la representación interna en el concepto “atributo en plataforma”, se nos facilita el diseño del esquema base de la arquitectura de alto nivel. para poder plasmar los patrones en ella y dejar a futuras versiones de este trabajo el crecimiento de la arquitectura, ya sea bajo el mismo uso de patrones de diseño o el re-uso de componentes como servicios.

3 Modelado de la arquitectura de producción textual colaborativa

Dada la naturaleza en línea de las plataformas colaborativas hemos decidido proponer una arquitectura distribuida, diseñada por capas para poder brindar modularidad y reusabilidad a la partes que la componen, como se propone en la literatura [5].

De esta forma la arquitectura se divide en cuatro capas principales que se explican a continuación.

Capa de aplicación. Presenta las vistas de acceso a la aplicación como por ejemplo los editores para revisión o modificación del objeto de contenido creado. **Capa de interfaz.** Utiliza el patrón “Facade” para proporcionar las interfaces de los diferentes servicios a las vistas de la capa de aplicación.

Capa de servicios. Se compone de tres subcapas que realizan el trabajo medular de la plataforma. *Subcapa Control de versiones.* El patrón “Composite” nos ayuda a mantener un agregado de objetos de las diferentes versiones del libro y al mismo tiempo marcar la versión elegida para publicación. *Subcapa Congelar Contenido.* El patrón “memento” congela el objeto libro en tiempo y espacio cuando es llamado para obtener una versión del libro. *Subcapa de creación de contenidos.*

El patrón “Factory Method” cuya implementación se presenta en la Tabla 2, funciona como una fábrica de atributos donde se recibe una llamada del patrón “proxy” de la capa de autenticación con el rol y el número de sesión donde se deberá enviar el objeto de creación de contenido adecuado. La fábrica regresa las instancias de objetos que corresponde a la columna “atributo en plataforma” de la Tabla 1.

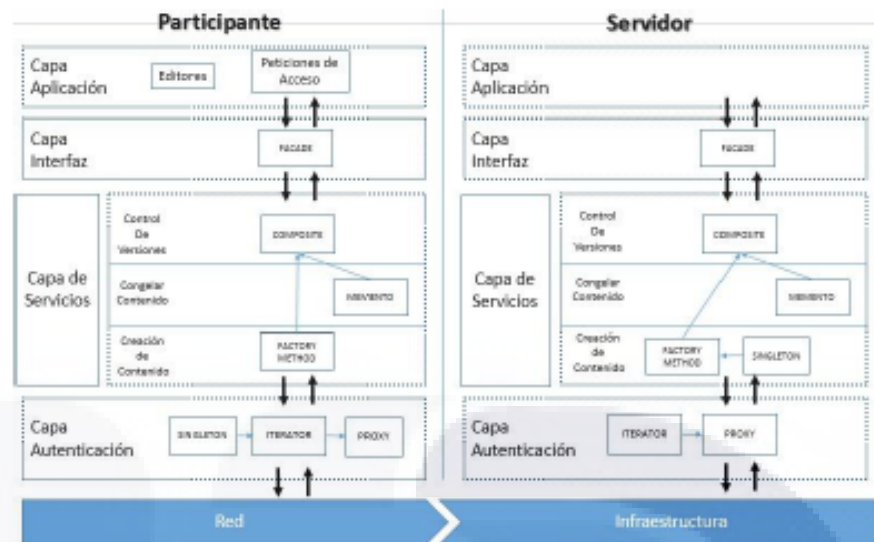


Figura 1. Arquitectura propuesta basada en patrones Gamma[4] para la plataforma de producción textual colaborativa basada en “Book-Driven Process”

Tabla 2. Patrón “factory method”

Nombre	Factory Method
Propósito	Define una interfaz para crear los objetos de servicio (atributos de la plataforma) a solicitud de la instancia “IProxy”.
Motivación	Este patrón proporciona una interfaz para la creación de los objetos de servicio.
Uso	Se usó Factory Method para construir los objetos de servicios.
Estructura	
Participantes	La interfaz “IFabricaServicios”. Define la interfaz de objetos al método de generador “Execute()”. Servicio. Es la entidad plantilla para los productos de servicios: “Contenido”, “Revisión”, “Formato” y “TablaContenido”, “ConcreteContenido”, “ConcreteRevisión”, “ConcreteFormato” y “ConcreteTablaContenido”. Implementa la interfaz del producto.
Colaboraciones	Servicio confía en sus subclases para definir el método de fábrica para que devuelva una instancia del Producto Concreto apropiado.

Consecuencias	Mediante el uso de la fábrica de servicios se pueden instanciar clases no concretas las cuales son dejadas a los atributos de la plataforma, de esta manera se puede agregar indistintamente procesos a la plataforma colaborativa
---------------	--

Capa de autenticación. Realiza el proceso de autenticación de los usuarios en la plataforma de la siguiente forma, el patrón "iterator" (iterador) maneja la lista de usuarios en el sistema en la Tabla 2 se muestra la aplicación del patrón de tal manera que cuando se ingresa "iterator" ésta recibe una llamada al método autentica () que confirma el rol y conecta con la fábrica de servicios(Factory method) .

Tabla 3. Patrón "iterator"

Nombre	Iterator
Propósito	Se usa el patrón "iterator" para recorrer la lista de usuarios de la plataforma colaborativa durante el proceso de autenticación.
Motivación	Cada objeto en la lista de usuarios contiene los elementos de autenticación "login", "password" y "Rol". Cuando un usuario solicita acceso a la plataforma, la interfaz de usuario hace una llamada a la interfaz del iterador "VistaLogin" para confirmar el acceso y permisos dependiendo del rol del usuario.
Uso	En este caso se usó como un mecanismo de autenticación que recorre la lista de usuarios cuando se solicita ingreso a la plataforma colaborativa.
Estructura	
Participantes	VistaLogin. Define la interfaz a la lista de usuarios de la plataforma colaborativa. Iterador. Implementa la interfaz de iteración. Realiza un seguimiento de la posición actual en el recorrido de la lista de objetos de usuarios de la plataforma colaborativa. Grupo de usuarios. Define una interfaz para crear un objeto iterador. Usuario. Implementa la interfaz de creación de iterador para devolver una instancia del iterador concreto adecuado.
Colaboraciones	Iterador pierde de vista el objeto actual en el agregado y puede calcular el usuario siguiente en el recorrido.
Consecuencias	El recorrido en la lista de objetos de usuarios se implementó hacia adelante iniciando por el nodo de cabecera, no se utiliza retroceso.

Dado que los patrones "iterator" y "factory method" se encuentra en capas separadas de la arquitectura es necesario un patrón de conexión de esta manera el patrón "proxy" se usa para generar un representante de la fábrica de servicios basada en "factory method" de la capa de servicios, en la Tabla 4 vemos las características de implementación de "proxy".

Tabla 4. Patrón "proxy"

Nombre	Proxy
Propósito	Proporciona presencia de la interfaz de Factory metodo en la capa de creación de contenidos.

Motivación	La interfaz de fábrica IFabricaServicios se encuentra en otra capa para poder tener acceso de manera remota la interfaz IProxy hace interacción.
Uso	La interfaz IProxy ejecuta la función SolicitaServicio() que solicita la ejecución de un objeto de servicio de la fábrica.
Estructura	<pre> classDiagram class IFabricaServicios { +Ejecuta() } class IProxy { } IFabricaServicios < -- IProxy </pre>
Participantes	IProxy (Proxy Imagen). Mantiene una referencia que le permite acceder a la representación del sujeto real IFabricaServicios. Sujeto (Gráfico). Define la interfaz común para IFabricaServicios e IProxy. IFabricaServicios. Define el objeto real que representa el proxy.
Colaboraciones	Proxy reenvía las solicitudes al "IFabricaServicios", para que este retorne el servicio que corresponde al rol usando el método "retornaServicio()".
Consecuencias	La interfaz "IProxy" llama a la interfaz de fabricación del servicio para creación de contenido que corresponde al rol que retorna el patrón "iterator" de la lista de usuarios de la plataforma colaborativa.

Finalmente para evitar que haya varias instancias de la lista ligada de usuario que recorre el patrón "iterator" que puedan generar problemas de autenticación, usamos al patrón "singleton" alrededor del objeto agregado de ListaUsuarios la implementación concreta se se muestra en la siguiente tabla.

Tabla 5. Patrón "singleton"

Nombre	Singleton
Propósito	Garantiza que la instancia del agregado de la lista de usuario de la plataforma colaborativa sea única.
Motivación	Asegurar una instancia única para evitar duplicación en la información de usuarios.
Uso	Se modifica la clase ListaUsuario para ser única.
Estructura	<pre> classDiagram class ListaUsuarios { +primero: Usuario +ultimo: Usuario +instancia +obtenerInstancia() -ListaUsuarios() } </pre>
Participantes	Clase Lista de usuarios es un agregado de varias instancias de la clase Usuario, la cual contiene el rol que indica cuales servicios se pueden solicitar a la fábrica a través del proxy.
Colaboraciones	Los clientes acceden a la instancia únicamente a través de la propiedad UInstancia.
Consecuencias	Controla que solo exista una sola instancia de la lista de usuarios de la plataforma colaborativa

4 Resultados

Para mostrar la validez y exactitud de la arquitectura se realizó un modelado entidad-relación de la arquitectura para confirmar la correcta alineación entre los patrones, en

la Figura 2 presentamos una parte de este modelo que muestra los primeros cuatro patrones que proponemos en las dos primeras capas “iterator”, “factory method”, “proxy” y “singleton”, y la validación de los principios de diseño orientados a objetos cuyo resultado se muestra a continuación.

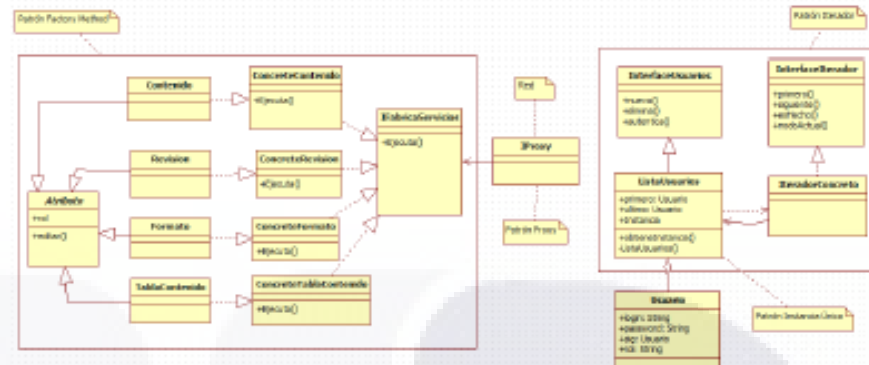


Figura 2. Modelo de datos de la capa de autenticación (patrones “iterator”, “singleton” y “proxy”) y la subcapa de creación de contenidos (patrón “factory method”)

Confirmación del principio “OpenClose”. El uso del patrón “Factory method” para definir una fábrica de atributos permite ingresar nuevos servicios como por ejemplo traducción sin modificar otros componentes de la arquitectura.

Confirmación del principio “Dependency Inversion”. se aplicó el principio de inversión de dependencia al separar la arquitectura en capas y modelar en base a patrones ya que ellos nos proporcionan interfaces para no tener dependencias directas de clases mayores a menores.

Confirmación de los principios “Interface Segregation”, “Liskov” y “Single Responsibility”. el ligado puro entre los patrones de diseño por sí mismo nos previene de generar interfaces que no se usan, dar dos razones de cambio a un módulo y hacer abuso de la herencia o el polimorfismo (principio de Liskov) ya que los patrones tienen métodos de interacción bien definidos. En nuestro diseño nos apegamos a los patrones tal cual y sólo se generaron dos nuevas entidades VistaLogin y Usuario, la primera que representa a la vista WEB del usuarios y la segunda los datos abstractos de usuario las cuales funciona como entidades de apoyo solamente. Como puede verse el modelo cumple con los principios de diseño orientado a objetos lo que establece una consistencia del mismo.

5 Conclusiones y Trabajo Futuro

En este caso se presentó una arquitectura para el desarrollo de una plataforma textual colaborativa basada en patrones de diseño GoF, la cual se diseñó tomando los requerimientos de la metodología de producción colaborativa llamado “Book-Driven Process” propuesto por LATIn project y que responde a la necesidad de marcos de producción textual colaborativa estándares.

El uso de patrones de diseño en la arquitectura obedece al hecho de proporcionar un modelo mejorado en comparación con aquellos elaborados sin el uso de patrones,

lo cual se confirmó con el análisis de la aplicación de los principios de diseño orientados a objetos. Como pudo verse, se estableció un primer diseño que propone un sistema distribuido con cuatro capas basándonos en la experiencia de un modelo propuesto anteriormente en la literatura. Para la confirmación de nuestra arquitectura se presentó el modelado de los primeros cuatro patrones de las primeras dos capas de la arquitectura colaborativa.

Como trabajo futuro proponemos la exploración de patrones que vayan enfocados al diseño de arquitecturas distribuidas, un ejemplo pueden ser los patrones J2EE, que aprovechan la distribución de los recursos en red, dado que los patrones GoF están dirigidos a aplicaciones para una sola máquina o “standalone”.

6 Referencias

1. LATIn Project: Analysis of Existing Technological Platforms for the Collaborative Production of Open Textbooks. www.latinproject.org, 2012.
2. Unigarro, G., M.: Educación Virtual: Encuentro Formativo en el Ciberespacio. Editorial UNAB. Bucaramanga, Colombia, 2001.
3. Gómez, L.A.: Aprendizaje en Ambientes Virtuales y Colaborativos, Los Computadores en la Nueva Visión Educativa. Escuela Colombiana de ingeniería, Colombia, 2000.
4. Gamma, E., Helm, R., Johnson, R., Vissides, J.: Design Patterns Elements of Reusable Object-Oriented Software. Ed. Addison-Wesley, 2000.
5. Licea, G., Favela, J.: Reusability in GroupWare Development through a Pattern System. Red de Revistas Científicas de América Latina y el Caribe, España y Portugal, Sistema de Información Científica Redalyc, 2001.