UNIVERSIDAD NACIONAL DE COLOMBIA

Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software

Katerine Villamizar Suaza

Universidad Nacional de Colombia

Facultad de Minas – Departamento de Ciencias de la Computación y de la Decisión

Medellín, Colombia

2013

Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software

Katerine Villamizar Suaza

Tesis de investigación presentada como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas

Director:

Carlos Mario Zapata Jaramillo, Ph.D.

Línea de Investigación:

Ingeniería de Software

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Ciencias de la Computación y de la Decisión

Medellín, Colombia

2013

Dedicatoria

A Nicole y Edward por ser parte fundamental de mi vida.

A mis papitos Luis y Nora por darme su apoyo cada día.

A mis hermanitas Luisa y Jessica por darme la alegría en los diferentes momentos que compartimos.

Katika.

Agradecimientos

Este proyecto es el resultado del apoyo de diferentes personas, a ellos muchas gracias.

Primeramente agradezco de una manera infinita a Dios por darme el valor, la fuerza y la constancia que me permitieron llevar a cabo este proyecto a pesar de los obstáculos presentados a lo largo del camino.

Agradezco a mi tutor Carlos Mario Zapata quien como padre me enseña la labor de la investigación, de la enseñanza, de la escritura y de la formación como persona pese a mi personalidad impulsiva.

Finalmente, agradezco a mis amigos y compañeros de estudio que fueron también un apoyo en cada uno de los procesos por los que tuve que pasar para desarrollar esta Tesis.

Resumen

En el desarrollo de software, las metodologías ágiles muestran efectividad en proyectos con restricciones de tiempo y flexibilidad. Esto se debe al valor que se le da a la interacción entre los interesados y los desarrolladores, concibiendo un desarrollo incremental del software con iteraciones muy cortas. La metodología XP (*Extreme* programming), es adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. En ésta metodología, se emplea la técnica de captura de requisitos como historias de usuario, que los interesados suelen escribir. Sin embargo, las metodologías ágiles, aparte de requieren alto conocimiento, gran experiencia de los desarrolladores y por ende un alto costo del mismo, se necesita de una especificación detallada y precisa que no sea ambigua ni que se presente para malas interpretaciones. Adicional a esto, se requiere por agilidad reducir los tiempos de implementación en gran manera y una capacidad de dar soporte a partir de una documentación bien elaborada que no se propicia en las metodologías ágiles. Debido a esto, en esta Tesis se definen equivalencias entre las historias de usuario y el lenguaje controlado UN-LENCEP que actualmente genera código, como una forma de mejorar el desarrollo ágil de software.

Palabras clave: UN-LENCEP, historias de usuario, metodología ágil, tiempo, costo.

Abstract

In the software development lifecycle, agile methodologies seem to be effective in projects with time and flexibility constraints. Such effectiveness is rooted on the value given to the stakeholder-developer interaction. Consequently, an incremental software development with short iterations is suitable. XP (Extreme Programming) is adequate for projects with changing, imprecise requirements and prone to high technical risks. Requirements elicitation in XP is made by using stakeholder-written User Stories. However, agile methods require a high level of knowledge and vast experience of the developers, raising the software development cost. Also, a detailed, precise, unambiguous specification is needed in order to reduce misunderstandings. Finally, agility demands reduction of the implementation time and the usage of a well-written documentation—but documentation is not promoted by agile methods. As a way to deal with the aforementioned problems, in this Thesis I define equivalences among user stories and the UN-LENCEP controlled language, which currently generates source code. I aim the improvement of the agile software development.

Keywords: UN-LENCEP, user stories, agile methodology, time, cost.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de abreviaturas

- AgEnD: *Agile enviroment Development.*
- ASD: *Adaptive Software Development.*
- AUP: *Agile unified process.*
- ATDD: *Acceptance Test Driven Development*.
- DTD: *Document Type Definition*.
- EP: Esquema Preconceptual.
- KAOS*: Knowledge Acquisition in Automated Specification.*
- HTML: *HyperText Markup Language.*
- HU: Historia de usuario.
- FDD: *Feature-driven development.*
- LD: *Lean Development.*
- TDD: *Test Driven Development*
- UNC-PSCoder: Una herramienta CASE basada en Esquemas Preconceptuales. para la generación automática de código.
- UN-HULEN: Traductor de historias de usuario a UN-LENCEP.
- UN-LENCEP: Universidad Nacional de Colombia—Lenguaje Controlado para la Especificación de Esquemas Preconceptuales.
- SXP: SCRUM y XP.
- STDD: *Story Test-Driven Development*.
- UML: *Unified Modeling Language.*
- XML: *eXtensible Markup Language*
- XP: *eXtreme Programming*

Introducción

Las metodologías ágiles constituyen un conjunto integrado de técnicas y métodos que permiten abordar de manera rápida cada una de las actividades del ciclo de vida de un proyecto de desarrollo (Cockburn, 2000). Las metodologías ágiles emplean las historias de usuario para especificar los requisitos del software, describiendo brevemente las características que el sistema debe poseer, ya sean requisitos funcionales como no funcionales (Gutiérrez, Escalona, Mejías y Torres, 2005).

A pesar de la acogida reciente de estas metodologías, presentan problemas de implementación, mantenimiento y costo (Gutiérrez, *et al.*, 2005). Los problemas de implementación se centran en la creación de las historias de usuario, debido a que los interesados las construyen y, en la mayoría de los casos, "no tienen claro lo que quieren" (Cooper y Sawaf, 2005). Adicional a esto, tanto las estimaciones como la implementación de las historias de usuario son subjetivas para los desarrolladores, ya que se hacen sin una metodología de estimación y la implementación se realiza de acuerdo con lo que entiende el desarrollador de la historia de usuario (Citón, 2006). También, se presentan problemas en cuanto al costo, debido a que se requiere una alta cualificación de los desarrolladores que las implementan. Finalmente, no se levanta la cantidad de documentación necesaria que se puede requerir en la fase de mantenimiento, por lo cual esta fase presenta falencias (Yagüe y Garbajosa, 2009).

En un intento por solucionar cada una de las problemáticas mencionadas anteriormente, se desarrollaron nuevas metodologías ágiles, integrando especificaciones de metodologías tradicionales (Meneses y Peñalver 2010). También, se proponen métodos de mejora en la gestión de las historias de usuario (Sánchez y Caños, 2010), la construcción de las mismas (Echeverry y Delgado, 2007) y la generación de código a partir de ellas (Blé y Colaboradores, 2010). Sin embargo, no se encuentra un método integrado que permita especificar de una manera inambigüa las historias de usuario, o

una generación automática de código a partir de las mismas que permita disminuir los costos de estas metodologías.

Así, en esta Tesis se propone una definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP, como una búsqueda para el mejoramiento de las metodologías ágiles, mediante una solución que permita atacar los diferentes problemas planteados. Se emplea UN-LENCEP, que es un lenguaje controlado que permite la comunicación y eliminación de ambigüedades entre los analistas de requisitos y los interesados (Zapata y Garcés, 2008). Dicho lenguaje corresponde de manera inambigüa a un esquema preconceptual único (Zapata, Chaverra y Zapata 2010). Al generar una traducción desde UN-LENCEP a las historias de usuario se logra una estructura definida de estas historias, centrando al interesado en el dominio de su problema y solucionando el problema de ambigüedad y completitud de las historias de usuario. Adicional a esto, los esquemas preconceptuales generan automáticamente diagramas de UML (Zapata, Arango y Gelbukh, 2007) y código fuente (Chaverra, 2011), lo que resuelve los problemas de mantenimiento y costo, respectivamente.

Esta Tesis posee la siguiente estructura: en el Capítulo 1 se presenta un marco teórico de los conceptos necesarios para la especificación de UN-LENCEP y de las historias de usuario en las metodologías ágiles; en el Capítulo 2 se analizan antecedentes de mejoras de historias de usuario creadas para la solución a los diferentes problemas de las metodologías ágiles; en el Capítulo 3 se presenta la definición del problema de investigación; en el Capítulo 4 se presenta la justificación de la tesis; en el Capítulo 5 se presentan los objetivos del trabajo de tesis; en el Capítulo 6 se presenta la solución, equivalencias entre las historias de usuario y especificaciones en UN-LENCEP partiendo de la propuesta de un formato para la construcción de historias de usuario y presentando un caso de estudio completo empleando las equivalencias definidas; en el Capítulo 7 se presentan las conclusiones y recomendaciones que se puede derivar de esta Tesis.

1. Marco teórico

1.1 Conceptos teóricos

En esta sección se describen los conceptos teóricos básicos de esta Tesis: las Metodologías ágiles, las historias de usuario y el lenguaje controlado UN-LENCEP, los cuales conforman la base para el entendimiento y desarrollo del trabajo propuesto.

1.1.1 Metodologías ágiles

En 2001 nace el término “ágil” aplicado al desarrollo de software. Existe un manifiesto ágil que presenta los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Este manifiesto pretende ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas (Letelier y Penádes, 2006).

Las metodologías ágiles constituyen un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo (Amaro y Valverde, 2007, P. 8) y permite resolver conflictos de costos y de tiempo. Entre las metodologías ágiles se encuentran XP y SCRUM. Estas metodologías se apoyan en un proceso iterativo e incremental que permite entregar versiones operativas de los proyectos al cliente más rápido. Ambas metodologías emplean las historias de usuario para la especificación de los requisitos (Letelier, *et al.,* 2006).

1.1.2 Historias de usuario

Se utilizan para especificar los requisitos de las aplicaciones software en las metodologías ágiles (SCRUM, XP, FDD, ASD, AUP, LD, etc.). Las historias de usuario son tarjetas en dónde el interesado describe brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer, sean requisitos funcionales o no funcionales (Amaro, *et al.,* 2007). Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que se pueda implementar en unas tres semanas aproximadamente (para no superar el tamaño de una iteración, que es el tiempo estimado para una entrega de un componente de desarrollo de manera incremental; Jeffries, Anderson y Hendrickson 2001). Las historias de usuario se descomponen en tareas de programación (*task card*) que los programadores implementan.

Beck (2000) presenta un ejemplo de historia de usuario integrado con una tarjeta de trabajo (*customer story and task card*) en los cuales se presentan los elementos de la figura 1-1:

- Historia de usuario: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del interesado, referencia a otra historia de usuario previa, riesgo, estimación técnica, descripción, notas.
- Tarjeta de trabajo: Lista de seguimiento con la fecha, estado cosas por terminar y comentarios.

Figura 1-1 Historia de usuario (Beck, 2000).

Historia de usuario y Tarjeta de Tareas

Fecha:	**Tipo:**	Nueva: __	Corrección: __	Mejora: __
No. Historia	**Prioridad**	Interesado: __	Técnica:__	**Prueba funcional**
Referencia: __	**Riesgo:** __		**Estimación técnica:**__	
Descripción de la tarea:				
Notas:				
Seguimiento de la tarea				
fecha	estado	Cosas por terminar	Comentarios	

A pesar de la estructura anteriormente presentada, actualmente, se usan las historias de usuario de acuerdo con su división fundamental, el nombre de la historia de usuario, un enunciado y los criterios de aceptación (Jeffries, 2001).

- El enunciado se enfoca en definir lo que quiere el usuario.
- Los criterios de aceptación, definen los requisitos del interesado del producto sobre cómo se debe comportar el sistema ante distintos eventos.

Los puntos anteriores se resumen en la propuesta del formato realizada por Cohn (2004).

> "Como (Rol) quiero (funcionalidad) que (beneficio)."

Ejemplo: Como estudiante quiero comprar un pase de estacionamiento que pueda conducir a la escuela.

A la hora de especificar una historia de usuario, Beck (2000) afirma que la única información que se debe recoger es el nombre de la historia y una descripción de la misma. Sin embargo, no es recomendable el uso de las plantillas que propone Beck donde solo se recoge estos dos detalles (Jeffries, 2001).

1.1.3 UN-LENCEP (Universidad Nacional de Colombia—Lenguaje Controlado para la Especificación de Esquemas Preconceptuales).

Es un Lenguaje Controlado diseñado para que el Interesado pueda expresar las ideas del dominio del problema, con el fin de realizar su traducción automática hacia los denominados Esquemas Preconceptuales (EP) (Zapata, *et al.*, 2007).

En la Figura 1-2 se presentan las equivalencias de sus especificaciones básicas en un subconjunto del lenguaje natural. En dónde,

- A, B, C y D son conceptos.
- <ES> y <TIENE> son relaciones estructurales, que tienen sus equivalencias mostradas en la Figura 1-2.
- <R1> y <R2> son relaciones dinámicas, como por ejemplo: registra, revisa, valida, chequea, califica, aprueba, presenta, llama, despacha, asigna, regresa, paga, realiza, elabora, prepara, entrega, recibe, solicita, etc. Estas relaciones son, por lo general, diferentes para los dominios, pero se suelen asociar con verbos de actividad.
- {COND} es una condición, que incluye un conjunto de conceptos relacionados mediante operadores de comparación (igual, mayor o igual que, menor o igual) y operadores algebraicos (suma, resta, multiplicación, división).
- <CUANDO>, <SI>, <ENTONCES>, <DADO QUE>, <LUEGO DE QUE> son palabras reservadas para el uso de condicionales e implicaciones.

Figura 1-2 Equivalencias de la especificación básica de UN–Lencep en un subconjunto del lenguaje natural (Zapata, *et al*., 2007).

Especificación básica de UN–Lencep	Equivalencia en un subconjunto del lenguaje natural
A <ES> B	A <ES UN TIPO DE> B A <ES UNA CLASE DE> B A <ES UNA ESPECIE DE> B B <SE DIVIDE EN> A
A <TIENE> B	A <INCLUYE> B A <CONTIENE> B A <POSEE> B A <CONSTA DE> B A <ESTA CONFORMADO POR> B A <ESTA FORMADO POR> B A <SE COMPONE DE> B A <ESTA COMPUESTO POR> B B <PERTENECE A> A B <ES UNA PARTE DE> A B <ESTA INCLUIDO EN> A B <ESTA CONTENIDO EN> A B <ES UN ELEMENTO DE> A B <ES UN SUBCONJUNTO DE> A
<CUANDO> A <R1> B, C <R2> D	<SI> A <R1> B <ENTONCES> C <R2> D <DADO QUE> A <R1> B, C <R2> D <LUEGO DE QUE> A <R1> B, C <R2> D

Un ejemplo de un discurso de un interesado en lenguaje UN-LENCEP se presenta a continuación:

Estudiante **es** persona

Profesor **es** persona

Profesor **tiene** curso

Curso **tiene** estudiante

Cuando estudiante **presenta** examen, profesor **califica** examen

Si nota>3 **entonces** estudiante **aprueba** curso

Examen **tiene** nota.

2. Antecedentes

A continuación, se presentan diferentes casos de solución encontrados en la revisión de la literatura sobre el problema planteado.

2.1 Desarrollo Dirigido por pruebas TDD (*Test Driven Development*)

La técnica sobre el diseño ágil con TDD, establece las pruebas como una herramienta del diseño del código escribiéndolas antes de comenzar la codificación. Los requisitos son pruebas de aceptación de las historias de usuario lo suficientemente claros para que los interesados los aprueben. Las pruebas de aceptación se validan con máquinas de estados (Blé *et al.*, 2010).

También, se plantea la alternativa de pasar directamente las historias de usuario a casos de uso mediante las características de las mismas. La correspondencia entre historias de usuario y casos de uso se puede ver en que el título de la historia de usuario se corresponde con el del caso de uso tradicional. Sin embargo, no se toma la historia de usuario para definir los requisitos y traducir formalmente las historias de usuario en casos de uso, ya que se puede caer en imprecisiones y en la malinterpretación del requisito a definir, mientras que contarlo con ejemplos ilustrativos, transmite la idea sin complicaciones.

El desarrollo dirigido por pruebas de aceptación (ATDD) o técnica de historias de desarrollo basada en pruebas (STDD) es lo mismo que TDD pero a un nivel diferente. En este caso, las pruebas de aceptación son los requisitos del negocio. "*En ATDD cada*

historia de usuario contiene una lista de ejemplos que cuentan lo que el interesado quiere, con "total claridad y ninguna ambigüedad" (Blé *et al.*, 2010).

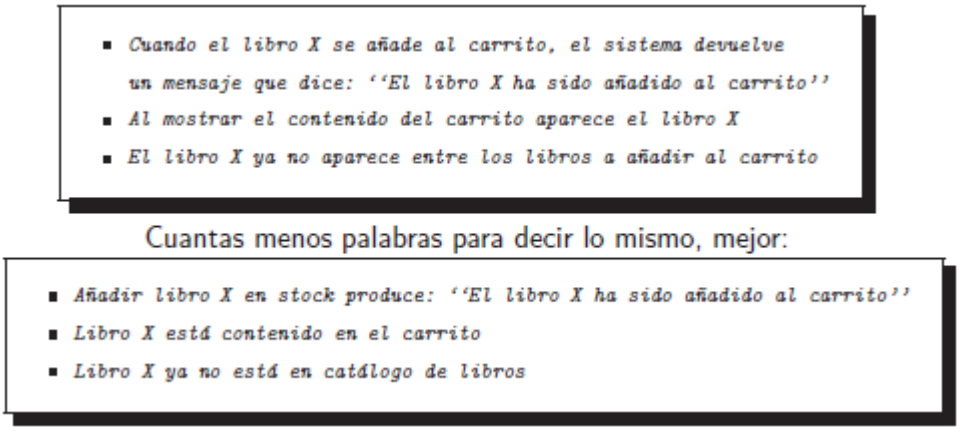
Estas historias de usuario tienen la característica de ser breves, concretas y su esfuerzo se puede estimar parcialmente. Son el resultado de escuchar al interesado y ayudarle a resumir el requisito en una sola frase. Estas historias de usuario tienen las siguientes características:

- Se escriben con el vocabulario de negocio del interesado.
- Se refinan a medida que pasan las iteraciones para su mejor estimación.

A partir de dichas historias de usuario se generan unas pruebas de aceptación (preguntas restringidas para validar la información con el interesado), que son afirmaciones en lenguaje natural. Luego, el equipo de desarrollo hace el esfuerzo de conectar esas frases con los puntos de entrada y salida del código. Básicamente, lo que proponen es escribir las frases con un formato determinado, como por ejemplo HTML, usando etiquetas de una manera específica para delimitar qué partes de la frase son variables de entrada para el código y cuáles son datos para validación del resultado de la ejecución (Blé *et al.*, 2010).

En la Figura 2-1, se presenta un ejemplo de historia de usuario.

Figura 2-1 Historias de usuario (Blé *et al.*, 2010)



Concordion, una herramienta para la especificación de pruebas de aceptación automáticas en java, produce un HTML modificado que marca en rojo las afirmaciones que no se cumplieron, además de mostrar las estadísticas generales sobre cuántas pruebas pasaron y cuantas no. En la Figura 2-2 se presentan un ejemplo de historia de usuario como prueba de aceptación y en la Figura 2-3 su correspondiente traducción a HTML en Concordion para la primera prueba de desarrollo.

Figura 2-2 Prueba de aceptación

```
# Aceptación - "2 + 2", devuelve 4
  • Sumar 2 al número 2, devuelve 4
  • Restar 3 al número 5, devuelve 2
  • La cadena "2 + 2"tiene dos números y un operador que son '2', '2' y '+'
# Aceptación - "5 + 4 * 2 / 2", devuelve 9
# Aceptación - "3 / 2", produce ERROR
# Aceptación - "* * 4 - 2": produce ERROR
# Aceptación -"* 4 5 - 2": produce ERROR
# Aceptación -"* 4 5 - 2 : produce ERROR
# Aceptación -"*45-2-": produce ERROR
```

Figura 2-3 Primera prueba de desarrollo

```
using System;
using System.Collections.Generic;
using System.Text;
using NUnit.Framework;
using SuperCalculator;

namespace UnitTests
{
    [TestFixture]
    public class CalculatorTests
    {
        [Test]
        public void Add()
        {
            Calculator calculator = new Calculator();
            int result = calculator.Add(2, 2);
            Assert.AreEqual(4, result);
        }
    }
}
```

De esta manera, se le pide al interesado definir las frases para su traducción, basándose en qué quiere el interesado y no en cómo lo quiere. Una ventaja de usar las historias de usuario y sus ejemplos para dirigir el desarrollo, es que se comprueba rápidamente si el programa está cumpliendo o no con los objetivos (Blé *et al.*, 2010). El hecho de pasar las historias de usuario a casos de uso sería una solución a la falta de documentación de las metodologías ágiles para el posterior mantenimiento de las mismas. Sin embargo, en esta técnica de desarrollo, no se define ninguna traducción directa para generar dicha documentación.

Además, se tiene la premisa de que el equipo de desarrollo tiene pocas probabilidades éxito ya que, si los analistas no trabajan estrechamente con los desarrolladores y *testers (analistas de pruebas)*, no se podría originar un flujo de comunicación adecuado como para que las preguntas y respuestas aporten valor al negocio.

2.2 Caso práctico de la metodología XP al desarrollo

Echeverry y Delgado (2007) plantean que los interesados no pueden escribir las historias de usuario. Ello se debe a que, en primer lugar, los interesados no saben qué es una historia de usuario y, en segundo lugar, a los interesados no le es fácil exteriorizar sus necesidades y redactarlas. Por tales razones, se plantea una estrategia para la captura de los requisitos, que consiste en realizar un diálogo entre los interesados y los desarrolladores, canalizando en un documento las ideas comprensibles para ambas partes (Echeverry, *et al.,* 2007). En la Figura 2-4 se presenta un ejemplo del formato de una historia de usuario, empleado para la captura de los requisitos.

Figura 2-4Plantilla historia de usuario (Echeverry, *et al.,* 2007).

Historia de Usuario	
Número:	**Nombre Historia de Usuario:**
Modificación (o extensión) de Historia de Usuario (Nro. y Nombre):	
Usuario:	**Iteración Asignada:**
Prioridad en Negocio: (Alta / Media / Baja)	**Puntos Estimados:**
Riesgo en Desarrollo: (Alto / Medio / Bajo)	**Puntos Reales:**
Descripción:	
Observaciones:	

Sin embargo, a pesar de aplicar esta estrategia, se encuentra, como ya se mencionó anteriormente, que el interesado no sabe lo que quiere y que en este caso práctico no se emplea ninguna especificación del lenguaje que ambas partes (interesados y analistas) pueden usar, lo que hará que genera subjetividad en el desarrollo de la aplicación de software.

2.3 Mejorando la gestión de Historias de usuario en *eXtreme Programming*

Debido a la necesidad de definir y representar la información de un sistema informático, Sánchez y Caños (2010) proponen una plantilla esencial representada en XML y basada en una plantilla de DTD (*Document Type Definition*). Esta plantilla en XML se genera directamente desde una historia de usuario y se representa en forma de escenarios para facilitar el control de versiones. De esta manera, se puede disponer de un formato que puedan procesar las herramientas informáticas. En la Figura 2-5 se presenta un ejemplo de un código XML generado a partir de una historia de usuario.

Figura 2-5 DTD del XML generado a partir de la Historia de Usuario (Sánchez, *et al.,* 2010).

```
<?xml version="1.0"?>
<!ELEMENT user_story (story_id,name,creation_date,customer,
                      priority,dependence*,estimation, risk,
                      release,iteration,upgrade*,base?,
      description)>
<!ELEMENT story_id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT creation_date (#PCDATA)>
<!ELEMENT customer (#PCDATA)>
<!ELEMENT priority (#PCDATA)>
<!ELEMENT dependence (#PCDATA)>
<!ELEMENT estimation (#PCDATA)>
<!ELEMENT risk (#PCDATA)>
<!ELEMENT release (#PCDATA)>
<!ELEMENT iteration (#PCDATA)>
<!ELEMENT upgrade (#PCDATA)>
<!ELEMENT base (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

Adicionalmente, en este proceso se dan algunas pautas para abordar las alternativas de evolución que puede sufrir la historia de usuario y se presenta el prototipo de una herramienta para gestionar historias de usuario que apoya dichas pautas (Sánchez, *et al.*, 2010). Sin embargo, esta herramienta se orienta más hacia el versionamiento, debido a que las historias de usuario son “volatiles” y, por lo tanto, la gestión de las mismas se puede complicar. Si estas especificaciones de historias de usuario en archivos XML se orientaran a una codificación automática inicial, se podría intentar solucionar la problemática presentada en cuanto a las metodologías ágiles. Fuera de eso, este proceso no presenta una estructura restringida y limitada de las historias de usuario para traducirlas directamente a código, sino que se copian directamente de la estructura general de las historias de usuario en lenguaje natural.

2.4 SXP (SCRUM y XP), Metodología ágil para el desarrollo de software

SXP es un “híbrido” cubano de metodologías ágiles que tiene como base las metodologías SCRUM y XP. SXP permite actualizar los procesos de desarrollo de software para el mejoramiento de su producción y su particularidad es tener, como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Esta metodología plantea la utilización de plantillas en las historias de usuario y, así, crear una estructura definida para ellas (Meneses, *et al.,* 2010). Sin embargo, la definición de las plantillas no es concreta, lo que no ayuda para disminuir la ambigüedad de las mismas.

2.5 Diseño de una metodología ágil de Software

Se construye una metodología AgEnD (Desarrollo Mejorado Por Agilidad) que reúne las mejores prácticas del universo ágil uniéndolas con un *framework* que ayuda a las organizaciones a implantar este tipo de metodologías. Se realiza una configuración de un proceso ágil de desarrollo basado en las prácticas ágiles y en factores relacionados con las personas, por ejemplo su productividad para generar mejores estimaciones (experiencia en aplicaciones, experiencias en lenguajes de programación y herramientas, entre otros). Se realiza, luego, un análisis de las prácticas y patrones que contribuyen a la implementación y posterior adaptación del proceso a la realidad de cada organización (Schenone, 2004).

Al contar con las personas como base de dicha metodología, se puede llegar a la subjetividad, debido a ciertas consideraciones en relación con la naturaleza de las personas, que no se pueden medir, como por ejemplo, sus dependencias emocionales. Las personas son organismos vivientes impredecibles (Cockburn, 2000), lo que no ayuda en cuanto a las estimaciones y a la subjetividad de la programación de los desarrolladores y el equipo de desarrollo.

De acuerdo a los antecedentes anteriormente mencionados, se presenta en la Tabla 2-1 su contribución en la problemática que se plantea en la sección 3.

Tabla 2-1Antecedentes – Problemática (elaboración propia).

Antecedente/ Problemática	Implementación	Mantenimiento	Costo
Desarrollo Dirigido por pruebas TDD (Blé *et al.*, 2010).	x		x
Caso práctico de la metodología XP al desarrollo (Echeverry, *et al.*, 2007).	x		
Mejorando la gestión de Historias de usuario en eXtreme Programming (Sánchez, *et al.*, 2010).	x		
SXP, Metodología ágil para el desarrollo de software (Meneses, 2010).	x		
Diseño de una metodología ágil de Software (Shenone, 2004).	x		

3. Definición del problema de investigación

3.1 Problemática

Existen diferentes metodologías en el proceso de desarrollo de software, entre las que se encuentran las metodologías ágiles, que constituyen uno de los temas de interés en la industria, debido a la rapidez que proporcionan a la construcción de un producto de software (Cockburn, 2000).

A pesar de la acogida de las metodologías ágiles y sus ventajas (que se revisan en la sección 4.1 de esta propuesta) para el desarrollo de software, éstas presentan problemas de implementación, mantenimiento y costo.

3.1.1 Problemas de implementación

El proceso de construcción de software, en una metodología ágil, inicia con el levantamiento de las historias de usuario (tarjetas que presentan un fragmento de la funcionalidad del sistema esperado); (Gutiérrez, *et al.*, 2005), empleadas para la captura de requisitos. Los interesados construyen las historias de usuario sin una estructura definida (Canós, Letelier y Penadés 2008) y, en la mayoría de los casos, "no tienen claro lo que quieren" (Isaacson, 2011). Por tal razón, las historias de usuario presentan descripciones ambiguas, incompletas e informales (Gutiérrez, *et al.*, 2005).

Luego de este proceso, los programadores realizan la estimación del esfuerzo requerido en la implementación de las historias de usuario. La estimación se inicia en la primera reunión de planeación y continúa a medida que se va construyendo el prototipo (Citón, 2006). Estas estimaciones son subjetivas, debido a que los programadores las hacen de acuerdo con lo que cada uno entendió de las historias de usuario y según su experiencia en desarrollo, sin emplear una metodología apropiada para tal caso (Cooper *et al.*, 2005).

Este proceso tiene gran incertidumbre, lo que puede ocasionar retrasos a la hora de la implementación del producto de software.

Al igual que en la estimación del esfuerzo, el proceso de desarrollo depende de lo que entienden los programadores sobre las historias de usuario (Citón, 2006). Así, la construcción del software se supedita a la subjetividad o habilidad de cada programador para entender las historias de usuario (Leffingwell, 2012), lo que puede traer retrasos y quizá insatisfacción del interesado en relación con el producto de software esperado.

3.1.2 Problemas de costo

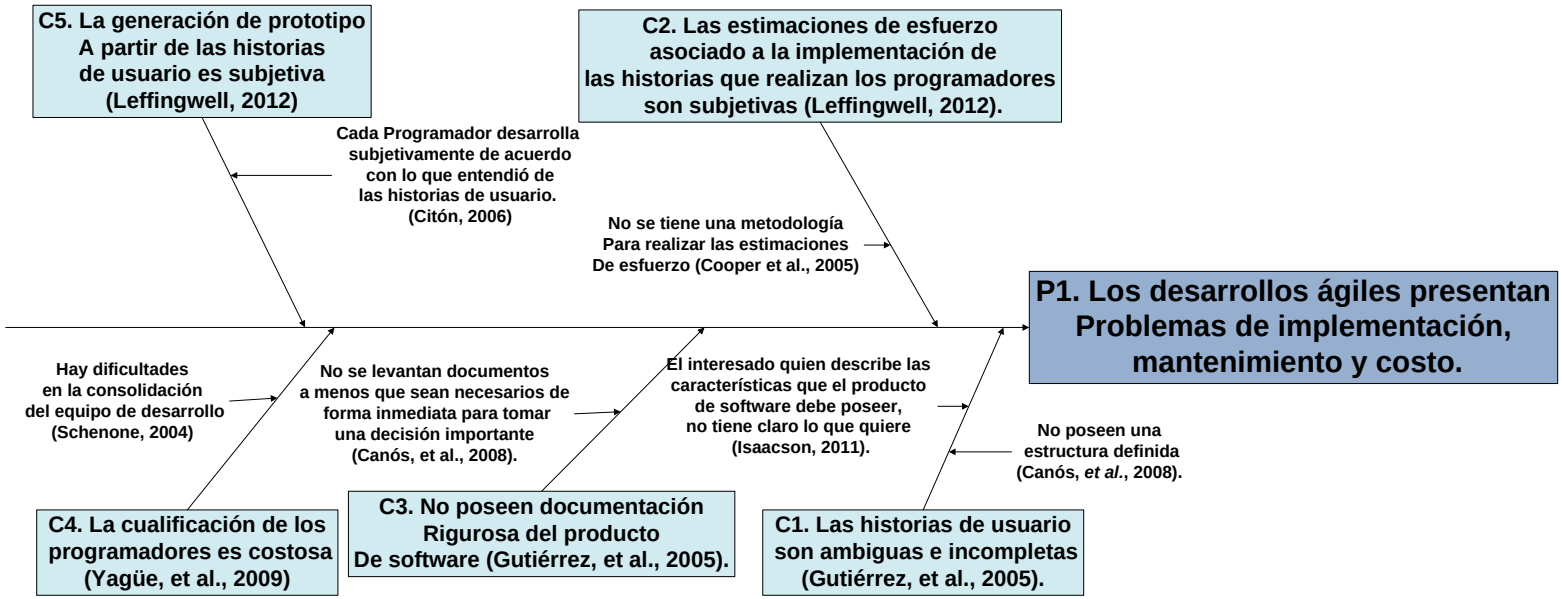
En las metodologías ágiles se requiere una alta calificación de los programadores (Yagüe, *et al.*, 2009). En este sentido, se generan deficiencias en la comunicación por el incremento en la complejidad de los productos de software, lo que ocasiona dificultades de consolidación del equipo de desarrollo y, consecuentemente, unos altos costos de la calificación requerida (Schenone, 2004).

3.1.3 Problemas de mantenimiento

El ciclo de vida del software termina con la implantación y el mantenimiento del producto de software. Se considera que el mantenimiento es uno de los mayores costos del desarrollo de software y se basa en la documentación generada del mismo (Sommerville, 2005). Las Metodologías ágiles no poseen documentación rigurosa del producto de software (Gutiérrez, *et al.*, 2005), pues se tiene la premisa de “no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante” (Canós, *et al.*, 2008). Por tal razón, el mantenimiento podría traer retrasos en la finalización de un proyecto de software e incrementar los costos del mismo.

En la Figura 3-1 se presenta un resumen de los problemas mediante un diagrama causa efecto.

Figura 3-1 Causa efecto "Problemas de las metodologías ágiles".



4. Justificación

En esta Tesis se busca el mejoramiento de las metodologías ágiles, mediante una solución que permita atacar los diferentes problemas planteados. La Universidad Nacional de Colombia, sede Medellín, emplea el lenguaje controlado UN-LENCEP (Zapata, *et al.*, 2008), el cual permite definir el dominio de cualquier problema. Este lenguaje es de fácil comprensión para un interesado en el desarrollo de un producto de software, dada su cercanía con el lenguaje natural.

Partiendo del lenguaje controlado UN-LENCEP se plantea la siguiente pregunta: ¿Es posible generar, a partir del Lenguaje Controlado UN-LENCEP, un acercamiento a las historias de usuario (base de las metodologías ágiles) que genere, directamente, un prototipo perteneciente a cualquier dominio, permitiendo que se resuelvan las inconformidades de implementación, mantenimiento y costo de las metodologías ágiles presentadas en la sección anterior?

La motivación de esta propuesta se fundamenta en la búsqueda de soluciones a los problemas presentados, para lo cual se emplearán los siguientes recursos:

- UN-LENCEP es un lenguaje controlado que permite la comunicación entre los analistas de requisitos y los interesados, que elimina ambigüedades de las definiciones que el interesado da al analista de requisitos sobre su sistema (Zapata, *et al.,* 2008). Dicho lenguaje corresponde de manera inambigua a un esquema preconceptual único (Zapata, *et al.*, 2010). Al generar una traducción desde UN-LENCEP a las historias de usuario se logra una estructura definida de éstas, centrando al interesado en el dominio de su problema. De esta manera, se ataca el problema de ambigüedad y completitud de las historias de usuario.

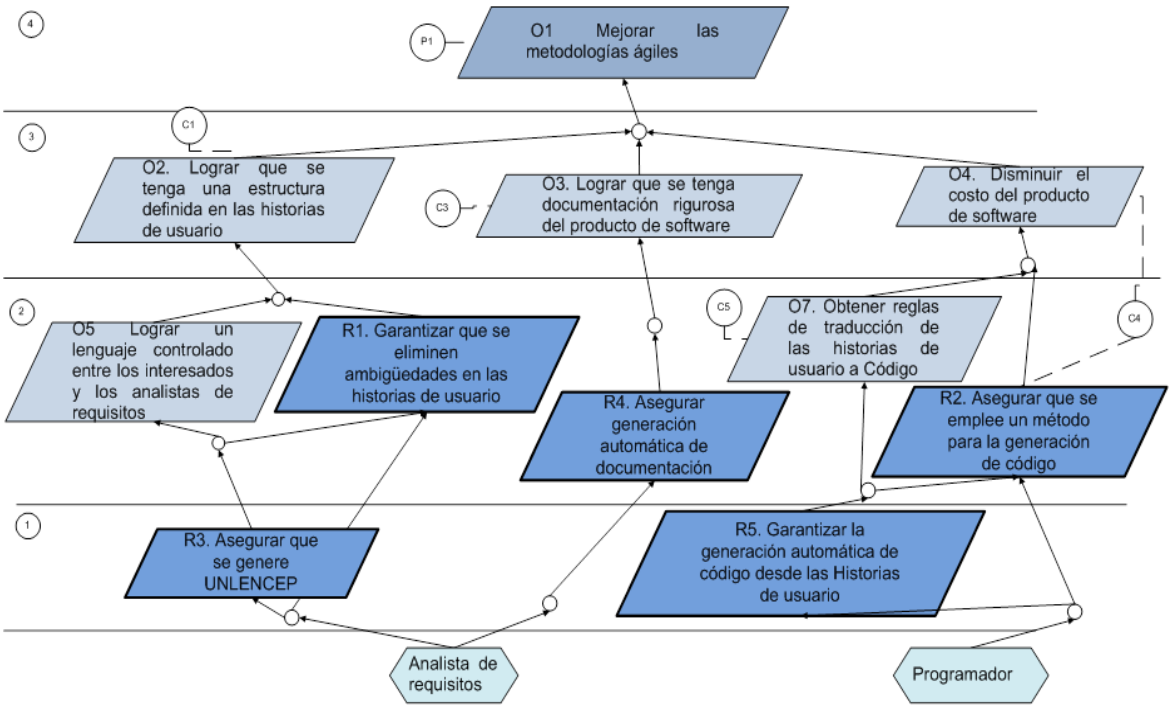
- Actualmente, UN-LENCEP genera automáticamente documentación, pues es posible realizar la transformación automática del discurso en cuatro de los diagramas de UML: Clases, Comunicación, Máquina de Estados y Secuencias (Zapata, *et al.,* 2008), los cuales pertenecen respectivamente a los diagramas estructurales, de Interacción y comportamentales (Zapata, *et al.*, 2007). Generando dichos diagramas, se lograría obtener, automáticamente, una documentación completa del producto de software, sin las pérdidas de tiempo que esto ocasiona en metodologías más rigurosas. Dicha documentación ayudaría a disminuir el problema de mantenimiento en las metodologías ágiles debido a la falta de documentación.

- Además, UN-LENCEP genera automáticamente código (Zapata, *et al.*, 2010) y un prototipo funcional a partir del dominio del problema (Chaverra, 2011). Creando reglas de transformación desde las historias de usuario a UN-LENCEP, se genera automáticamente código, lo que reduce la subjetividad en cuanto a la generación de código del prototipo que desarrollan los programadores. Así, el prototipo tiene, explícitamente, lo que el usuario plasmó en las historias de usuario junto con el analista. Esto ayuda, también, a que no se incrementen los costos que implica corregir errores de subjetividad en cuanto al entendimiento del problema. Consecuentemente, se disminuye la experiencia que requieren los programadores en las metodologías ágiles (Yagüe, *et al.*, 2009).

En la Figura 4-1 se presenta un diagrama de KAOS (*Knowledge Acquisition in Automated Specification*) como resumen de la manera como se atacarán los problemas presentados en la sección anterior.

El diagrama de KAOS (Lamsweerde, Dardenne y Fickas, 1993) representa jerárquicamente objetivos, de manera que los de alto nivel se presentan en la raíz del árbol y los de bajo nivel (más operativos y relacionados con los requisitos y expectativas del software) en la parte inferior. Además, se agregan las responsabilidades a los diferentes actores.

Figura 4-1 Diagrama de KAOS



5. Objetivos

5.1 Objetivo general

Mejorar las metodologías ágiles por medio del planteamiento de un conjunto de reglas que permitan la traducción automática de las historias de usuario al lenguaje controlado UN-LENCEP.

5.2 Objetivos específicos

1. Analizar las historias de usuario para determinar sus principales características y obtener una estructura definida de las mismas.

2. Definir los elementos que se pueden mapear desde las historias de usuario a UN-LENCEP, comparando las características identificadas de las historias de usuario con la estructura de este lenguaje.

3. Diseñar reglas que permitan la traducción de las historias de usuario a UN-LENCEP.

4. Realizar un conjunto de casos de estudio representando la traducción de las historias de usuario a UN-LENCEP.

5. Validar las reglas de traducción por medio de la generación automática de documentación y de un prototipo funcional.

6. Propuesta de solución

De acuerdo con los principios ágiles de software, las historias de usuario deben tener la cantidad mínima de información que sea útil para el proyecto de desarrollo (Wake, 2001). Sin embargo, para muchos proyectos esto no es suficiente, lo que implica adicionar información a las mismas. Para mantener la simplicidad de las metodologías ágiles, cualquier otra información asociada con la historia de usuario se debe justificar respecto del beneficio que aporte (Sánchez, *et al.*, 2003).

En la Tabla 6-1 se presenta un ejemplo de una historia de usuario con los componentes básicos. Un nombre, una descripción y unos criterios de aceptación.

Tabla 6-1 Ejemplo de historia de usuario con estructura básica

Historia de Usuario		
Nombre	Inicio de sesión	
Descripción	Como usuario, quiero iniciar sesión en la página web, para poder utilizar la funcionalidad específica del usuario registrado.	
Criterios de Aceptación		
	1. Camino exitoso: Inicio de sesión satisfactorio	
	Dado	usuario se ingresa en la página de "Inicio de sesión"
	Cuando	usuario ingresa nombre de usuario y contraseña en la página de "Inicio de sesión" y da clic en el botón "Enter".
	Entonces	Se presenta mensaje "Usuario logueado satisfactoriamente"
	2. Nombre de usuario o contraseña faltante	
	Dado	usuario se ingresa en la página de "Inicio de sesión"
	Cuando	usuario NO ingresa nombre de usuario y/o contraseña en la página de "Inicio de sesión" y da clic en el botón "Enter".
	Entonces	Se presenta mensaje de error "Falta usuario o contraseña"
	3. Nombre de usuario o contraseña incorrectos	
	Dado	usuario se ingresa en la página de "Inicio de sesión"
	Cuando	usuario ingresa nombre de usuario y/o contraseña erróneos en la página de "Inicio de sesión" y da clic en el botón "Enter".
	Entonces	Se presenta mensaje de error "Usuario o contraseña

6.1 Estructura específica de las historias de usuario

Recabando datos durante algún tiempo, en una empresa colombiana de software se encuentra que, a pesar de que existe una estructura específica para la escritura de las historias de usuario, se presentan diversas falencias. La estructura para la escritura de las historias de usuario es la siguiente:

- Descripción: Como (Rol) requiero (funcionalidad) para (beneficio).
- Criterios de aceptación: Verificar (Resultado), cuando (Condición).

En la Tabla 6-2 y Tabla 6-3, se elaboran dos ejemplos de historias de usuarios con varias de las falencias encontradas.

1. Historia de usuario: Administración de emails

Tabla 6-2 Historia de usuario "Administración de emails"

Historia de usuario	
Nombre	Administración de emails
Descripción	Como **administrador** requiero poder consultar emails y poder borrarlos, activarlos y desactivarlos.
Criterios de aceptación	
1. Verificar que si el usuario no ingresa ningún criterio de búsqueda, muestre un mensaje solicitando algún criterio, y no ejecute la consulta. 2. Verificar cuando se ejecuta cualquiera de las acciones para un email, se haya realizado una consulta previa. 3. Verificar que la consulta arroje los emails según los criterios de búsqueda. 4. Verificar que los emails con diferentes estados aparezcan en la consulta con su estado correspondiente. 5. Verificar que se visualicen los emails, cuando se realiza alguna consulta.	

Falencias:

- Se presentan diferentes actores en la descripción y los criterios de aceptación: “Administrador” y “usuario” respectivamente, lo que hace que no se tenga claro cuál es el rol que va a usar la funcionalidad de la historia de usuario.
- Uso confusos de verbos: “Requiero poder” en la descripción. Se tiene por definición que las historias de usuario deben ser lo “suficientemente” comprensibles y delimitadas para implementarlas en poco tiempo (Jeffries *et al.*, 2001). En este caso, no se presenta una escritura comprensible.
- No hay trazabilidad en los verbos usados. “Arrojar”, “visualizar”, “presenten”: se usan diferentes verbos para referirse a lo mismo. Esto puede presentar confusión a la hora de que un desarrollador, o incluso el interesado, lo interprete.
- Se plantea un formato y, sin embargo, no se sigue con el mismo: “Verificar cuando” en el criterio número 2.
- Sólo se describe la acción *consultar*; sin embargo, hacen falta las acciones borrar, activar y desactivar. Para dar claridad a la funcionalidad que se presenta en la descripción, los criterios de aceptación deberían especificar cada una de dichas acciones.

2. Historia de usuario.

Tabla 6-3 Historia de usuario "Registrar y editar datos básicos".

Historia de usuario	
Nombre	Registrar y editar datos básicos
Descripción	Como Coordinador requiero registrar y editar los datos básicos de mis usuarios.
Criterios de aceptación	
1. Permitir consultar los datos de mis usuarios, cuando se ingresa a la aplicación y se ejecute la acción de Buscar. Nota: Mostrar la consulta ordenada ascendentemente por fecha de creación del usuario en el sistema. 2. Verificar que se muestren los valores por defecto, cuando un nuevo usuario se va a crear. 3. Verificar que se encuentre habilitado el botón para buscar el rol asociado al usuario, cuando se da clic en la lupa del campo Rol usuario.	

Falencias:

- El nombre de la historia de usuario se refiere a dos acciones que se realizarán en el sistema; sin embargo, las historias de usuario deben ser atómicas y es una de las características que las identifica. En este caso se están presentando dos funcionalidades en una sola.
- A pesar de que existe una estructura definida para los criterios de aceptación, se incluyen notas a veces con expresiones muy técnicas, lo que hace tediosa la comprensión de las historias de usuario para los interesados.

Realizando un estudio de las diferentes falencias encontradas y de los antecedentes presentados, se propone una estructura para las historias de usuario en la Tabla 6-4, construida a partir de una ontología, con base en un método de implementación de desarrollo de ontologías (Noy y McGuinness, 2001).

Tabla 6-4 Propuesta formato historia de usuario.

Historia de Usuario				
Código				
Nombre				
Actor				
Descripción	"Como (Actor) quiero (funcionalidad) que (beneficio)." (Cohn, 2004)			
HU Relacionada(s):	**Código:**		**Nombre:**	
Módulo				
Criterios de aceptación	**Condición**		**Resultado**	
	Cuando se Acción + Concepto		Se debe cumplir que + ESPECIFICACIÓN	

6.1.1 Código

Es un valor único que identifica la historia de usuario de otras historias dentro de un proyecto. Se puede presentar en cualquier formato.

Ejemplos: 01, 001, HU001.

6.1.2 Nombre

Es el título de la historia de usuario como resumen de la funcionalidad que se describirá en ella. Generalmente se presenta en infinitivo como la acción que se desarrollará.

Ejemplos: Iniciar sesión, Ingresar nómina, Registrar datos.

6.1.3 Actor

Es el rol que tendrá la persona que realiza la funcionalidad de la historia de usuario en el sistema a construir.

Ejemplos: Administrador, Coordinador, Auxiliar.

6.1.4 Descripción

Representa en una frase lo que quiere el usuario. El formato de la descripción es el siguiente (Cohn, 2004):

"Como (Actor) quiero (funcionalidad) que (beneficio)."

En dónde,

- El actor es el rol descrito anteriormente.
- La funcionalidad es la necesidad del interesado. La funcionalidad tendrá la siguiente estructura:

  Verbo + Concepto.
- El beneficio es lo que se obtiene al ejecutar dicha funcionalidad. La cláusula "que (beneficio)" es opcional.

Ejemplo: Como estudiante quiero comprar un tiquete de bus que sirva para ir a la escuela.

En dónde,

- Actor: Estudiante
- Funcionalidad:
  - Verbo: Comprar
  - Concepto: tiquete de bus.
- Beneficio: sirva para ir a la escuela.

6.1.5 HU relacionada(s)

Es (son) la(s) historia(s) de usuario que pertenece(n) al mismo módulo dentro de una aplicación de software. En este campo, se presenta el código y el nombre de la historia de usuario relacionada. Este elemento se agrega con el fin de generar completitud. Así, entre diferentes historias de usuario, se puede implementar un módulo completo.

6.1.6 Módulo

Es el nombre que lleva un conjunto de historias de usuario cuya funcionalidad es similar o que tiene un fin específico.

Ejemplo:

Módulo: Liquidaciones

Historias de usuario relacionadas: Administrar liquidaciones, Parametrizar liquidaciones, Consultar liquidaciones.

6.1.7 Criterios de aceptación

Detallan cómo se debe comportar el sistema para ejecutar la descripción de la historia de usuario.

Se compone de condición más resultado, en donde la condición representa lo que se debe cumplir para obtener dicho resultado. Para estas dos partes se propone una estructura definida.

- Condición: Se compone de la palabras reservadas “Cuando se” más una acción que en primera medida será la acción de la descripción de la historia de usuario más un concepto. Así,

  “Cuando se” + Acción + Concepto.

- Resultado: Se compone de las palabras reservadas “Se debe cumplir que” más la especificación de la descripción de la historia de usuario como resultado de la condición.

  “Se debe cumplir que” + Especificación.

  Ejemplo: Cuando se busca los criterios de búsqueda, se debe cumplir que si los campos no tienen valor, se presente un mensaje.

Se define, entonces, que los criterios de aceptación serán la especificación de la o las relaciones dinámicas de la descripción de la historia de usuario.

6.1.8 Aspectos a tener en cuenta

Para la construcción de las historias de usuario el analista de requisitos debe tener en cuenta las siguientes condiciones:

- UN-LENCEP no permite los plurales, por tal razón, se debe escribir la historia de usuario sin los mismos.

  Ejemplo: La historia de usuario para el interesado dice la siguiente frase: “si campo nombre de usuario y/o campo contraseña no tienen valor”, se debe presentar de la siguiente manera: “Si campo nombre de usuario no tiene valor y campo contraseña no tiene valor…”

En la Tabla 6-5 se presenta un ejemplo de una historia de usuario con el formato planteado.

Tabla 6-5 Ejemplo historia de usuario con estructura propuesta.

Historia de Usuario				
Código	1			
Nombre	Iniciar sesión			
Actor	Usuario			
Descripción	"Como usuario quiero iniciar sesión de la página Web que sirva para utilizar la funcionalidad específica del sistema para el usuario registrado."			
HU Relacionada(s):	**Código:**		**Nombre:**	
Módulo				
Criterios de aceptación	**Condición**	**Resultado**		
	Cuando se inicia sesión de la página Web	Se debe cumplir que si usuario ingresa nombre de usuario y usuario ingresa contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".		
	Cuando se inicia sesión de la página Web	Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje "Falta usuario o contraseña".		
	Cuando se inicia sesión de la página Web	Se debe cumplir que Idusuario diferente a usuario Y Idcontraseña diferente a contraseña, se presenta mensaje "Usuario o contraseña incorrectos".		

6.2 Elementos a Mapear / Equivalencias

Al contar con una estructura específica de las historias de usuario, se determina que los elementos a mapear con el lenguaje controlado UN-LENCEP serán la descripción y los criterios de aceptación, debido a que son la parte fundamental de la historia de usuario. Además, porque en estos dos elementos de la historia de usuario se presentan partes que se mapean directamente con el lenguaje UN-LENCEP.

Tabla 6-6 Historia de usuario con elementos a traducir.

<table>
<tr><th colspan="5">Historia de Usuario</th><th>UN-LENCEP</th></tr>
<tr><td>Código</td><td colspan="4"></td><td rowspan="3"></td></tr>
<tr><td>Nombre</td><td colspan="4"></td></tr>
<tr><td>Actor</td><td colspan="4"></td></tr>
<tr><td rowspan="2">Descripción</td><td colspan="4" rowspan="2">“Como (Actor) quiero (funcionalidad) que (beneficio)." (Cohn, 2004)</td><td>Relación dinámica</td></tr>
<tr><td>Actor + Funcionalidad (Verbo + concepto)</td></tr>
<tr><td>HU Relacionada(s):</td><td>Código:</td><td></td><td>Nombre:</td><td></td><td rowspan="2"></td></tr>
<tr><td>Módulo</td><td colspan="4"></td></tr>
<tr><td rowspan="3">Criterios de aceptación</td><td colspan="2">Condición</td><td colspan="2">Resultado</td><td>Condicional</td></tr>
<tr><td colspan="2" rowspan="2">Cuando se + Acción + Concepto</td><td colspan="2" rowspan="2">Se debe cumplir que + ESPECIFICACIÓN (Restricción) o (Condicional + SE + Acción + Concepto)</td><td><SI> {COND} <ENTONCES> A <R1> B, <SINO> C <R2> D</td></tr>
<tr><td>ESPECIFICACIÓN: (Restricción) o (Condición) <R1> C</td></tr>
</table>

6.2.1 Descripción

La funcionalidad de la descripción contiene uno o más verbos más un concepto, que representa en el lenguaje controlado UN-LENCEP una relación dinámica construida de la siguiente manera:
Actor + Funcionalidad = Actor + Verbo + Concepto = Relación dinámica en el lenguaje UN-LENCEP.
Ejemplo: Auxiliar ingresa nombre. Así, Auxiliar es el actor, ingresa es el verbo y nombre es el concepto.

6.2.2 Criterios de aceptación

La condición más el resultado representa un condicional en el lenguaje UN-LENCEP.
El condicional en UN-LENCEP se determina con <SI> {COND} <ENTONCES> A <R1> B, <SINO> C <R2> D, En dónde {COND} es una condición expresada en términos de conceptos. <R1> y <R2> son verbos dinámicos. <SINO> es opcional, por ejemplo: si M es mayor que 100 entonces A registra B
Así, el {COND} representa la condición en la historia de usuario y lo que se escribe posterior al <ENTONCES> sería el resultado.

6.3 Reglas que permiten la traducción de las historias de usuario a UN-LENCEP.

A pesar de que existen elementos que se pueden mapear directamente desde las historias de usuario al lenguaje controlado UN-LENCEP, hay que especificar detalladamente los mismos, debido a que los criterios de aceptación generalmente se escriben sin pensar en una estructura específica lo que puede incurrir en ambigüedades y malas interpretaciones a la hora de referirse a ellos.

Antes de proponer las reglas que traduzcan las historias de usuario a UN-LENCEP, se debe tener en cuenta que existen elementos del esquema preconceptual que aún no se contemplan en UN-LENCEP y que son útiles a la hora de traducir un lenguaje natural a

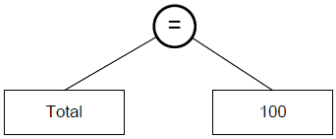
uno controlado. Por esta razón, se propone, a continuación, el uso de los mismos en UN-LENCEP.

6.3.1 Especificaciones.

Son descripciones del comportamiento de las relaciones dinámicas (Chaverra, 2011). Generalmente se componen de restricciones (Sección 6.3.1.1) y condicionales (Sección 6.3.1.2), para las cuales, se presentan las siguientes especificaciones:

- Asignaciones: Le dan valor a un concepto, mediante operaciones matemáticas u operaciones entre diferentes conceptos. Un ejemplo de una asignación a un concepto se presenta en la Figura 6-1, Total “igual a” 100.

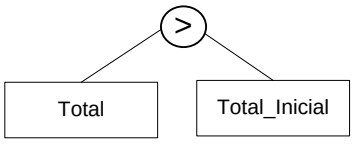
Figura 6-1 Ejemplo de asignación en EP.



De esta manera, los operadores “igual a”.

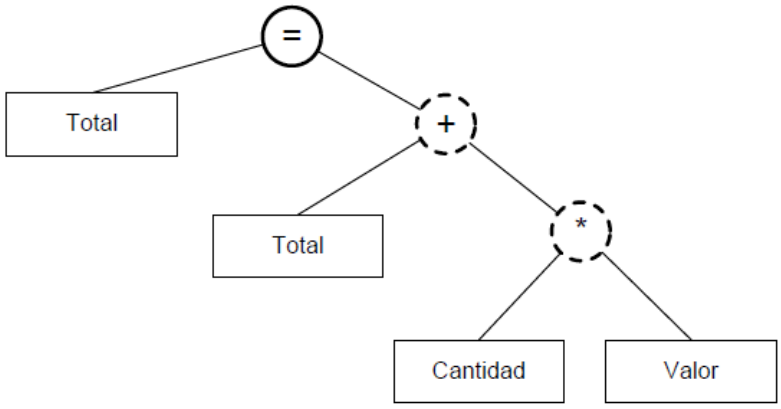
- Comparaciones: Se propone las comparaciones entre conceptos o entre conceptos y valores, ya que no se tienen contempladas en los esquemas preconceptuales. Para esto se tienen las palabras reservadas “menor o igual que”, “mayor o igual que”, “mayor que”, “menor que”, “diferente a”, entre otros, serían palabras empleadas en el lenguaje controlado UN-LENCEP para representar comparaciones. Un ejemplo de comparación se presenta en la Figura 6-2, Total > Total_Inicial.

Figura 6-2 Ejemplo comparaciones en EP.



- Operaciones matemáticas: Se realizan entre conceptos. Por ejemplo, Total “igual a” total “más” cantidad “multiplicado por” valor.

Figura 6-3 Ejemplo operaciones matemáticas en EP.



Así, los operadores matemáticos serían palabras empleadas en el lenguaje UN-LENCEP, “más”, “menos”, “dividido”, “multiplicado por”, “dividido por”, entre otros.

- Operaciones atómicas: representan procedimientos básicos entre conceptos definidos con anterioridad. Para la especificación de una relación dinámica, sólo se deben tomar en consideración los siguientes verbos.

  - Listar: se usa para obtener el listado completo de un concepto específico.

    Actor + “Lista” + Concepto.
    Ejemplo: Profesor lista examen.
    Así, las palabras visualizar, presentar (incluye mensajes), mostrar, listar y enumerar; representan la palabra reservada “lista” en el esquema preconceptual.

  - Insertar: se usa para guardar la información que pertenece a un concepto.

    Actor + “Inserta” + Concepto.
    Ejemplo: Alumno tiene nombre y apellido. Profesor inserta alumno. Significa que, Profesor inserta nombre y apellido del alumno.

Así, las palabras guardar, insertar, ingresar, archivar, almacenar, meter, registrar, digitar, escribir, apuntar.

- Seleccionar: se usa para obtener el listado de un atributo de un concepto.

  Actor + “Selecciona” + Concepto.

  Ejemplo: Examen tiene porcentaje. Profesor selecciona porcentaje de examen.

  Para este caso, se usa la palabra reservada “Selecciona”. Para cualquiera de las siguientes palabras: Selecciona, elige, escoge, chequea, aparta, prefiere, opta.

- Editar: se usa para actualizar la información de un concepto.

  Actor + “Edita” + Concepto.

  Para esto debe existir el nombre de la variable de dicho concepto y el valor de dicha variable, de la siguiente manera.

  Ejemplo: Alumno tiene nombre y apellido.

  Nombre igual a Idnombre y Apellido igual a Idapellido.

  Idnombre es igual a Carlos.

  Profesor edita alumno, Idnombre igual a José y Idapellido igual a Sánchez.

  Así, las palabras: edita, ajusta, modifica, cambia, adecúa, arregla, reforma, transforma, altera y rectifica; en UN-LENCEP representarán la palabra reservada “editar” en el EP.

- Eliminar: se usa para obtener borrar el registro de un concepto.

Actor + “Elimina” + Concepto.

Ejemplo: IdAlumno igual a Alumno. Profesor elimina Alumno.

Para este caso, se usa la palabra reservada “Eliminar”. Para cualquiera de las siguientes palabras: Borrar, eliminar, suprimir, anular, quitar, deshacer.

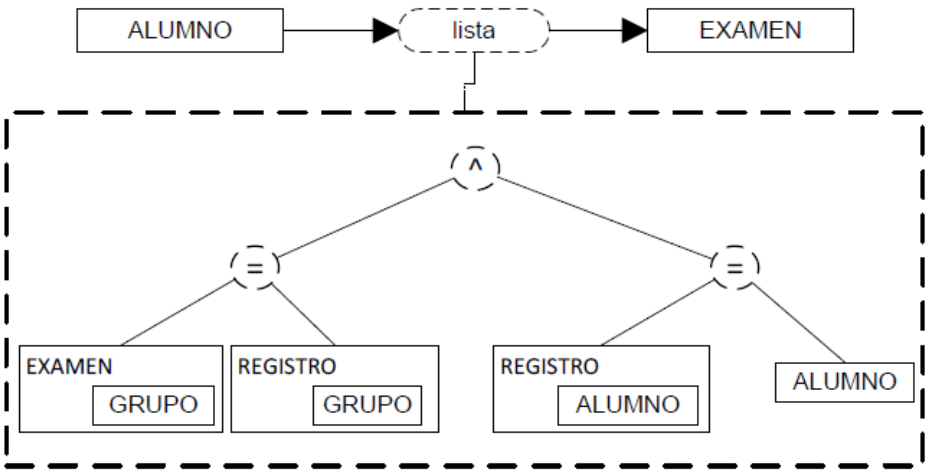
Nota: Adicional a los elementos mencionados, se pueden emplear todos los elementos definidos inicialmente del lenguaje controlado UN-LENCEP. La única restricción será para las relaciones dinámicas, pues se emplearán únicamente las operaciones atómicas.

Para el lenguaje UN-LENCEP se propone la expresión “SIGNIFICA QUE” para representar dichas especificaciones. Así,

Actor + Verbo + Concepto, “Significa que” + ESPECIFICACIÓN.

A partir del esquema preconceptual de la Figura 6-4 se presenta un ejemplo de la cláusula en lenguaje UN-LENCEP.

Figura 6-4 Operación atómica "Lista" con restricciones (Chaverra, 2011).



UN-LENCEP: Alumno lista examen, SIGNIFICA QUE examen del grupo es igual a registro del grupo y registro del alumno es igual al alumno.

Como parte de las especificaciones se deben tener presente además los siguientes elementos del esquema preconceptual.

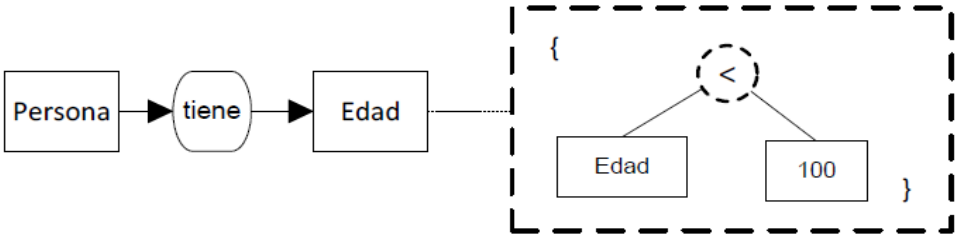
6.3.1.1Restricciones

Las restricciones son limitaciones que se realizan a algún elemento de UN-LENCEP y se pueden asociar con notas, conceptos o especificaciones (Chaverra, 2011). Una definición de las notas es la siguiente:

- Notas: permiten especificar los posibles valores que puede tener un concepto. Para UN-LENCEP, se propone las palabras reservadas “Puede tener los valores:” Así:
  Concepto + “Puede tener los valores:” + Valores
  Ejemplo:
  Estado puede tener los valores: Bueno, malo.

  Para las restricciones, se define las palabras reservadas “Se debe cumplir que”
  Ejemplo:

Figura 6-5 Ejemplo restricciones en EP.



Persona tiene edad, se debe cumplir que edad menor que 100.

6.3.1.2Implicaciones y condicionales

Las implicaciones son expresiones que implican causalidad (Zapata, *et al.*, 2007) y pueden tener alguna de las siguientes formas.

- <SI>A <R1>B <ENTONCES>C<R2>D
- <DADO QUE> A <R1> B, C<R2> D
- <LUEGO DE QUE > A <R1> B, C<R2>D

Sin embargo las implicaciones pueden iniciar en condicionales como se refleja en la primera sentencia. La cláusula <SI> indica que se debe cumplir una condición para que

luego suceda una acción. A pesar de que esto ya se contempla en UN-LENCEP, se pueden presentar varios casos adicionales como los siguientes.

- El condicional no es una relación dinámica <SI> A <R1> B, sino alguno de los siguientes elementos:

    - Asignaciones. Para este caso, se propone la forma <SI> A <AS> B, entonces… En dónde la denotación <AS> significa asignación y puede ser cualquiera de las palabras reservadas mencionadas en la sección 6.3.1 para las asignaciones. Ejemplo: Si número de estudiantes igual a 100, entonces profesor matricula estudiantes.
    - Comparaciones. Para este caso, se propone la forma <SI> A <CO> B, entonces… En dónde la denotación <CO> significa comparación y puede ser cualquiera de las palabras reservadas mencionadas en la sección 6.3.1 para las asignaciones. Ejemplo: Si número de estudiantes mayor a 50, entonces profesor matricula estudiantes.
    - Operaciones matemáticas entre conceptos, para este caso se propone la forma <SI> A <OP> B, entonces… En dónde la denotación <OP> significa operación. Ejemplo: Si total igual a total más cantidad multiplicado por valor, entonces contador agrega total. (Nota: las operaciones matemáticas entre conceptos implican una asignación inicial ejemplo Si total igual a…)

- Se debe tener presente que, en el resultado de la implicación, también se pueden dar los casos anteriores. De esta manera se proponen las siguientes expresiones:

    - <SI>A <R1, AS, CO, OP> B <ENTONCES>C<R2, AS, CO, OP>D
    - <DADO QUE> A <R1> B, C<R2, AS, CO, OP> D
    - <LUEGO DE QUE > A <R1> B, C<R2, AS, CO, OP>D

En dónde las expresiones <R1, AS, CO, OP>, <R2, AS, CO, OP> indican que se pueden presentar cualquiera de las relaciones.

- R1 y R2 indican relación dinámica.
- AS indica asignación.
- CO indica comparación.
- OP indica operación.

- Se requiere más de una expresión para que se dé un resultado. Para esto, se adicionan los operadores “Y”, “O” a los condicionales. Así:
  - <SI>A <R1, AS, CO, OP> B Y E <R3, AS, CO, OP> F <ENTONCES>C<R2, AS, CO, OP>D

    Ejemplo: Si profesor asigna nombre de estudiante Y profesor asigna nota de examen, entonces profesor califica examen

  - <SI>A <R1, AS, CO, OP> B O E <R3, AS, CO, OP> F <ENTONCES>C<R2, AS, CO, OP>D

    Ejemplo: Si fecha igual a “30/06” O fecha igual a “31/12”, entonces contador paga prima.

6.3.1.3Tiene único

Es una relación estructural que denota un identificador único para un concepto (Chaverra, 2011). Así, en UN-LENCEP, se adiciona las palabras reservadas “tiene único”, con la forma A “tiene único”, “tiene única” B. Ejemplo: Profesor tiene única identificación.

6.3.1.4 Artículos

El esquema preconceptual permite de una manera gráfica ver las relaciones entre diferentes conceptos, en dónde, dichos conceptos no tienen un artículo definido. Por esta razón, se excluirán del lenguaje UN-LENCEP los artículos, tanto determinados como indeterminados. El, la, lo, un, una, los, las, unos, unas.

6.3.2 Reglas de traducción

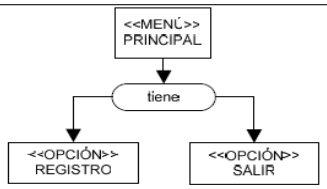
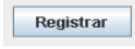
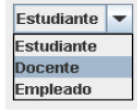
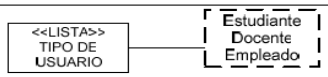
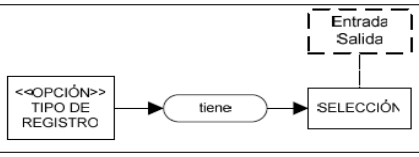
En la historia de usuario, generalmente, se hace referencia al sistema como un actor que realiza acciones, para lo cual, se debe tener presente que se hablan en términos de interfaces gráficas de usuario. En los esquemas preconceptuales ya se tiene la traducción directa a interfaces gráficas de usuario, teniendo en cuenta algunas palabras reservadas del sistema: variable, ventana, dato, usuario, tabla. Así, se emplearán dichas palabras en el lenguaje UN-LENCEP, como parte del sistema, como se presenta en la Tabla 6-7.

Tabla 6-7 Descripción de los elementos de la especificación (Hidalgo, 2009).

Elementos de la especificación		
Elemento	***Descripción***	***Ejemplo***
Variable	Concepto en el que se almacenan otros conceptos	<<VARIABLE>> CÓDIGO DE BARRAS
Ventana	Interfaz desplegada	<<VENTANA>> PRINCIPAL
Dato	Bases de datos, datos contenido en bases de datos	<<DATO>> TÍTULO
Usuario	Persona que interactúa con el sistema.	<<USUARIO>> ASISTENTE

Se debe tener presente que las interfaces de usuario “ventanas” tienen diferentes elementos, para lo cual se tienen también diferentes palabras reservadas: menú, campo, botón, lista y botón radio. Al usarlas, representarán una parte de la ventana como interfaz de usuario, como se presenta en la Tabla 6-8.

Tabla 6-8 Elementos de la ventana (Hidalgo, 2009).

Elemento	*Interfaz*	*Esquemas preconceptual*
Menú		
Campo		
Botón		
Lista		
Botón radio		

Teniendo en cuenta lo anterior y los elementos a mapear de las historias de usuario se definirán las reglas de traducción desde las historias de usuario al lenguaje controlado UN-LENCEP.

El formato del código de la regla será el siguiente: Regla (Tipo de Regla) (Consecutivo de regla) (Resultado). Leyéndose de derecha a izquierda. En dónde,

- Tipo de regla: G: Generales, U: UN-LENCEP (O: Operaciones), D: Descripción, C: Criterio de aceptación (C: Condicional, R: Resultado)
- Consecutivo: Consecutivo de la regla iniciando en 1.
- Resultado:
    - N: representa que se construye una nueva frase para UN-LENCEP.
    - R: significa que se reemplaza en el texto de la HU el resultado.

Ejemplo: CC1R significa regla número 1 de reemplazo para el condicional del criterio de aceptación.

6.3.2.1Reglas de UN-LENCEP

De acuerdo con los aspectos a tener en cuenta y a las restricciones que tiene UN-LENCEP, se proponen las siguientes reglas para las historias de usuario.

Regla UO1R: Eliminar artículos determinados e indeterminados. La, El, lo, los, las, Una, uno, unas, unos.

Excepción: Los artículos de las frases que estén entre comillas no se deben eliminar, debido a que pueden ser nombres, estados, mensajes, etc.

Regla UO2R: Reemplazar palabras reservadas de las operaciones atómicas a sus respectivas operaciones atómicas. Ejemplo: Guarda por inserta.

6.3.2.2Reglas generales

Regla G1N: La forma A “de” B, donde A y B son conceptos, implica una relación estructural en UN-LENCEP, B “tiene” A, como una nueva frase. Cuando se usa esta regla, se debe eliminar el concepto B de la historia de usuario y la palabra “de” para continuar con la traducción de la historia de usuario sin estas palabras.

Excepción: No aplica para las frases que estén entre comillas debido a que pueden ser nombres, estados, mensajes, etc.

Ejemplo

Tabla 6-9 Ejemplo Regla G1N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Descripción	“Como Coordinador quiero guardar información de cliente”.	Cliente tiene información.

Así, A = Información, B= Cliente.

Nota: Se debe tener en cuenta que son excepciones de esta regla los nombres compuestos. Para diferenciarlos irán entre comillas (“”). Ejemplo: “Asociación de tenistas”. Sin embargo, ésta regla se deja a discreción del analista de software.

Regla G2R: Cuando se presente un valor entre comillas (“”) de la forma A “B”, representa una asignación del valor de un concepto, lo cual se transforma a A igual a “B” en la frase encontrada.

Ejemplo

Tabla 6-10 Ejemplo Regla G2R

Historia de usuario		**UN-LENCEP**
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje "Falta usuario o contraseña".	Página Web tiene sesión. Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje igual a “Falta usuario o contraseña”.

Regla G2N: Como consecuencia de la regla anterior, se agrega una nueva frase con la traducción anterior. Un valor entre comillas (“”) de la forma A “B”, representa una asignación del valor de un concepto, lo cual se traduce a A igual a “B”.

Regla G2N.1: Adicional a lo anterior, se realiza la transformación a los posibles valores, cuando se encuentran varias frases que corresponden a un mismo concepto. Así, A igual a “B”, A igual a “C”, A igual a “D” se traduce a A puede tener los valores: “B”, “C,”, “D”.

Ejemplo

Tabla 6-11 Ejemplo Regla G2N.1

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje "Falta usuario o contraseña". Cuando se inicia sesión de la página Web, si usuario ingresa nombre de usuario y usuario ingresa contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".	Página Web tiene sesión. Mensaje puede tener los valores: “Falta usuario o contraseña”, "Usuario inició sesión satisfactoriamente". Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje igual a “Falta usuario o contraseña”.

Regla G3R: Al encontrar las palabras reservadas “no tiene valor”, “es vacío” o “sin valor”, de la forma A “palabra reservada”, esto representa que un concepto se encuentra vacío, es decir un *null* en esquemas preconceptuales y se traduce en UN-LENCEP a “A igual a vacío”, que se reemplaza en la frase encontrada, siendo vacío el valor del concepto A.

Ejemplo:

Tabla 6-12 Ejemplo Regla G3R

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que SI nombre de usuario no tiene valor O contraseña no tiene valor, se presenta mensaje "Falta usuario o contraseña".	Página Web tiene sesión. Coordinador inicia sesión. Significa que, SI nombre de usuario igual a vacío O contraseña igual a vacío, se presenta mensaje igual a “Falta usuario o contraseña”.

Regla G4N: Cuando se tiene la expresión “tiene único” o “tiene única” en UN-LENCEP, de la forma A “expresión” B, se hace referencia a que existe un dato único en base de datos y se traduce a A tiene único dato B. En esquemas preconceptuales se entiende como la creación de la tabla A con el dato B único (Chaverra, 2011).

Ejemplo

Tabla 6-13 Ejemplo Regla G4N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que usuario tiene única identificación	Página Web tiene sesión. Usuario tiene nombre. Coordinador inicia sesión. … usuario tiene único dato identificación…

Regla G5N: Cuando se usa un concepto con prefijo “ID” (Id + A) para representar un valor de un concepto. Se tienen dos reglas:

Regla G5N.1 Inicialmente se hace referencia a una variable, lo cual se traduce a “variable IdA”.

Regla G5N.2 Se deduce una asignación, de una variable a un concepto. Generando una nueva frase. Así, Variable IdA igual a campo A.

Ejemplo:

Tabla 6-14 Ejemplo Regla G5N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Buscar usuario	
Criterios de aceptación	Cuando se busca usuario, Se debe cumplir que Idusuario igual a Usuario.	Variable Idusuario igual a usuario.

Regla G5R: Adicional a lo anterior, también se debe realizar un reemplazo en la frase encontrada para la asignación “igual a” o comparación “diferente a”.

Ejemplo:

Tabla 6-15 Ejemplo Regla G5R

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión, se debe cumplir que SI Idusuario diferente a usuario Y Idcontraseña diferente a contraseña, se presenta mensaje "Usuario o contraseña incorrectos".	Cuando se inicia sesión, se debe cumplir que SI variable Idusuario diferente a campo usuario Y variable Idcontraseña diferente a campo contraseña, se presenta mensaje "Usuario o contraseña incorrectos".

6.3.2.3Reglas de la descripción

En la sección 6.1.4 se determinó que la descripción se define así: “Como (Actor) quiero (funcionalidad) que (beneficio)." En dónde, la funcionalidad será Acción + concepto.

Regla D1N: La funcionalidad (<R1>B) que se encuentra en la descripción de la historia de usuario, generalmente hace referencia a la acción principal que se realiza sobre la ventana, lo que se traduce en “Ventana” + nombre HU + “tiene botón” + R1, como una nueva frase. En dónde R1 es la relación dinámica en UN-LENCEP.

Ejemplo

Tabla 6-16 Ejemplo Regla D1N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Descripción	“Como usuario quiero iniciar sesión de la página Web que sirva para utilizar la funcionalidad del sistema para del usuario registrado."	Ventana Iniciar sesión tiene botón iniciar sesión.

Regla D2N: La funcionalidad de la descripción, se traduce a la forma A <R1> B en UN-LENCEP, como una nueva frase, acompañada de las palabras reservadas “Significa que:”, de dónde parte la especificación de todos los criterios de aceptación.

En dónde,

- A es el actor, que se toma directamente del elemento “Actor” de la historia de usuario.
- <R1> es la acción, que se extrae de la “funcionalidad” de la descripción de la historia de usuario.
- B es el concepto, también extraído del “Concepto” de la “funcionalidad” de la descripción de la historia de usuario.

Ejemplo

Tabla 6-17 Ejemplo Regla D2N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Descripción	"Como (Coordinador) quiero (guardar información de clientes)".	Coordinador guarda información.

Así, A= actor= Coordinador, <R1> = Acción=Guardar; B= Concepto= Información.

6.3.2.4Reglas de los criterios de aceptación

En la sección 6.1.7 se determinó que los criterios de aceptación se definen así: Condición + Resultado = "Cuando se" + Acción + Concepto, "Se debe cumplir que" + Especificación.

6.3.2.4.1 Condición

Regla CC1R: Las palabras reservadas "Cuando se" en la condición de los criterios de aceptación se reemplaza con el "Actor" de la historia de usuario. Así, se traduce la condición a una relación dinámica de la forma A <R1> B en UN-LENCEP.

En dónde,

- A es el actor, que se traduce de la palabra "Cuando se" tomado del elemento "Actor" de la historia de usuario.
- <R1> es la acción.
- B es el concepto.

Ejemplo

Tabla 6-18 Ejemplo Regla CC1R

Historia de usuario		UN-LENCEP
Actor	Coordinador	Página Web tiene sesión. Coordinador inicia sesión de la página Web.
Criterios de aceptación	**Condición**	
	Cuando se inicia sesión de la página Web	

Así, "Cuando se" se cambia con "Coordinador", que es el actor de la historia de usuario.

6.3.2.4.2 Resultado

Las especificaciones del resultado de los criterios de aceptación, como se mencionó en la sección 6.1.7 son la especificación de la descripción de la historia de usuario. Para dichas especificaciones, generalmente, se emplean restricciones, que incluyen condicionales e implicaciones.

Regla CR1N: Cuando se usan las operaciones atómicas insertar o editar sobre conceptos o se usan asignaciones o comparaciones a conceptos (Excepción: No aplica para la asignación “igual a vacío”.), se refiere a que hay un campo en pantalla que tendrá algún valor y se traduce como “campo” + B, dónde B es el concepto. Además, se agrega la frase “ventana” + nombre HU tiene campo B, en UN-LENCEP.

Ejemplo

Tabla 6-19 Ejemplo Regla CR1N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si usuario ingresa nombre de usuario y usuario ingresa contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".	Página Web tiene sesión. Usuario tiene nombre. Ventana Iniciar sesión tiene campo nombre. Ventana iniciar sesión tiene campo contraseña.

Regla CR2N: Cuando se usan la operación atómica seleccionar se refiere a que puede existir en la pantalla un menú, una lista, un botón radio, un *checkbox* o un botón. Así, para no restringir a que sea una opción, se deja a criterio del analista para que agregue la palabra reservada antes del concepto al que se le aplica la operación atómica. Además, se agrega la frase “ventana” + nombre HU tiene “Palabra elegida” B, en UN-LENCEP.

Ejemplo 1:

Tabla 6-20 Ejemplo 1Regla CR2N

Historia de usuario		UN-LENCEP
Actor	Administrador	
Nombre	Crear usuario	
Criterios de aceptación	Cuando se crea usuario, Se debe cumplir que administrador seleccione menú principal.	Ventana crear usuario tiene menú principal.

Así, “menú” es la palabra reservada que elige el analista.

Ejemplo 2:

Tabla 6-21 Ejemplo 2 Regla CR2N

Historia de usuario		UN-LENCEP
Actor	Administrador	
Nombre	Crear usuario	
Criterios de aceptación	Cuando se crea usuario, Se debe cumplir que administrador seleccione lista país.	Ventana crear usuario tiene lista país.

Así, “lista” es la palabra reservada que elige el analista.

Ejemplo 3:

Tabla 6-22 Ejemplo 3 Regla CR2N

Historia de usuario		UN-LENCEP
Actor	Administrador	
Nombre	Crear usuario	
Criterios de aceptación	Cuando se crea usuario, Se debe cumplir que administrador seleccione botón radio género.	Ventana crear usuario tiene botón radio género.

Así, “botón radio” es la palabra reservada que elige el analista.

Regla CR3R: Las palabras reservadas “Se debe cumplir que” se refieren a la especificación de la relación dinámica que se encuentra en la condición del criterio de aceptación, la cual se traduce a las palabras reservadas “Significa que” en la historia de usuario.

Ejemplo

Tabla 6-23 Ejemplo Regla CR3R

Historia de usuario		**UN-LENCEP**
Actor	Coordinador	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si usuario ingresa nombre de usuario y contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".	Coordinador inicia sesión, significa que, si usuario…

Regla CR4R: Luego de las palabras reservadas “Se debe cumplir que” si la frase a continuación inicia con un “SI” significa que se presenta un condicional en UN-LENCEP, lo que se traduce a alguna de las formas:

- <SI>A <R1, AS, CO, OP> B <ENTONCES>C<R2, AS, CO, OP>D
- <DADO QUE> A <R1> B, C<R2, AS, CO, OP> D
- <LUEGO DE QUE > A <R1> B, C<R2, AS, CO, OP>D
- <SI>A <R1, AS, CO, OP> B Y E <R3, AS, CO, OP> F <ENTONCES>C<R2, AS, OP>D
- <SI>A <R1, AS, CO, OP> B O E <R3, AS, CO, OP> F <ENTONCES>C<R2, AS, CO, OP>D

Nótese que, para este caso, en las historias de usuario se presenta la condición de la forma mencionada y el resultado <ENTONCES> como la palabra reservada SE.

Regla CR4R: El pronombre “SE” en la especificación de resultado del criterio de aceptación, se refiere al resultado de la condición que el sistema realiza alguna acción. Esta palabra reservada se traduce a “Entonces” + “Ventana” + Nombre de la historia de usuario.

Ejemplo Condicional

Tabla 6-24 Ejemplo Regla CR4R - Condicional.

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que SI usuario ingresa nombre de usuario y contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".	Página Web tiene sesión. Usuario tiene nombre. Coordinador inicia sesión. Significa que, SI Usuario ingresa nombre de usuario y Usuario ingresa contraseña …

Ejemplo Resultado

Tabla 6-25 Ejemplo Regla CR4R - Resultado.

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si usuario ingresa nombre de usuario y si usuario ingresa contraseña, SE presenta mensaje "Usuario inició sesión satisfactoriamente".	Página Web tiene sesión. Usuario tiene nombre. Coordinador inicia sesión. … ventana iniciar sesión presenta mensaje…

Ejemplo unificado

Tabla 6-26 Ejemplo Regla CR4R - Completo.

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si usuario ingresa nombre de usuario y si usuario ingresa contraseña, SE presenta mensaje "Usuario inició sesión satisfactoriamente".	Página Web tiene sesión. Usuario tiene nombre. Coordinador inicia sesión. Significa que, SI Usuario ingresa nombre de usuario y Usuario ingresa contraseña Entonces ventana iniciar sesión presenta mensaje “Usuario inició sesión satisfactoriamente".

Regla CR4N: Partiendo de la regla anterior, se debe realizar una unificación de los criterios de aceptación. Cuando se presenta la especificación de una misma frase se deben unir presentando las diferentes especificaciones para una sola relación dinámica. Así, A <R1> B, significa que: Especificaciones (<Condicionales> o <Restricciones>)

Ejemplo:

Tabla 6-27 Ejemplo Regla CR4N

Historia de usuario		UN-LENCEP
Actor	Coordinador	
Nombre	Iniciar sesión	
Criterios de aceptación	Cuando se inicia sesión de la página Web, Se debe cumplir que si usuario ingresa nombre de usuario y si usuario ingresa contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente". Cuando se inicia sesión de la página Web, Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se lista mensaje "Falta usuario o contraseña".	Usuario inicia sesión, significa que: Si usuario ingresa nombre y usuario ingresa contraseña, entonces ventana "iniciar sesión" lista mensaje igual a "Usuario inició sesión satisfactoriamente". Si campo nombre igual a vacío o campo contraseña igual a vacío, ventana "iniciar sesión" lista igual a mensaje "Falta usuario o contraseña".

En la Tabla 6-28, Tabla 6-29, Tabla 6-30 y Tabla 6-31 se presenta un resumen de las reglas.

Tabla 6-28 Resumen de reglas de traducción (UN-LENCEP, Generales y Descripción).

Tipo	Código	Descripción	►	Resultado
UN-LENCEP	UO1R	Eliminar artículos determinados e indeterminados.		
	UO2R	Reemplazar palabras reservadas a operaciones atómicas.		
Generales	G1N	A "de" B	►	B tiene A
	G2R G2N	A "B"	►	A igual a B A puede tener los valores: “B”
	G3R	A "no tiene valor" A "es vacío" A "sin valor"	►	A igual a vacío
	G4N	A "tiene único" B A "tiene única" B	►	A tiene único dato B
	G5N G5R	IdA	►	Variable IdA igual a campo A
Descripción	D1N	<R1>B	►	Ventana "Nombre de HU" tiene botón <R1>
	D2N	<R1>B	►	Actor <R1> B, Significa que:

Tabla 6-29 Resumen de reglas de traducción (Criterios de aceptación).

Tipo	Código	Descripción	►	Resultado
Criterios de aceptación	CC1R	"Cuando se"	►	Actor
	CR1N	**Operación atómica Insertar:** Inserta B	►	Ventana "Nombre de HU" tiene campo B
		Operación atómica Editar: Edita B	►	
		Asignaciones o comparaciones: (“igual a”, “menor o igual que”, “mayor o igual que”, “mayor que”, “menor que”, “diferente a”) B	►	
	CR2N	**Operación atómica Seleccionar:** Seleccionar "Menú" B	►	Ventana "Nombre de HU" tiene Menú B
		Seleccionar "Lista" B	►	Ventana "Nombre de HU" tiene Lista B
		Seleccionar "Botón radio" B	►	Ventana "Nombre de HU" tiene Botón radio B
	CR3R	"Se debe cumplir que"	►	Significa que

Tabla 6-30 Resumen de reglas de traducción (Criterios de aceptación).

Tipo	Código	Descripción	►	Resultado
Criterios de aceptación	CR4R CR4N	<Condición> "SE" <Resultado>	►	<Condición> ventana "Nombre de HU" <Resultado> <SI>A <R1, AS, OP> B <ENTONCES Ventana "Nombre de HU"><R2, AS, OP>D <SI>A <R1, AS, OP> B Y E <R3, AS, OP> F <ENTONCES Ventana "Nombre de HU">C<R2, AS, OP>D <SI>A <R1, AS, OP> B O E <R3, AS, OP> F <ENTONCES Ventana "Nombre de HU">C<R2, AS, OP>D

Tabla 6-31 Resumen de reglas de traducción (Criterios de aceptación).

Tipo	Código	Descripción	►	Resultado
Criterios de aceptación	CR4R CR4N	**Posibles casos condiciones:** - "Se debe cumplir que" + "SI" + A <R1, AS, OP> B "SE" + <R2, AS, OP>D - "Se debe cumplir que" + "SI"+ A <R1, AS, OP> B Y E <R3, AS, OP> F <ENTONCES>C<R2, AS, OP>D - "Se debe cumplir que" + "SI"+ A <R1, AS, OP> B O E <R3, AS, OP> F <ENTONCES>C<R2, AS, OP>D	►	<Condición> ventana "Nombre de HU" <Resultado> <SI>A <R1, AS, OP> B <ENTONCES Ventana "Nombre de HU"><R2, AS, OP>D <SI>A <R1, AS, OP> B Y E <R3, AS, OP> F <ENTONCES Ventana "Nombre de HU">C<R2, AS, OP>D <SI>A <R1, AS, OP> B O E <R3, AS, OP> F <ENTONCES Ventana "Nombre de HU">C<R2, AS, OP>D

6.3.3 Agrupación de reglas para su aplicación

Dependiendo del orden en que se deben aplicar las reglas para generar el lenguaje controlado UN-LENCEP, se realizará una agrupación de las mismas, de la siguiente manera:

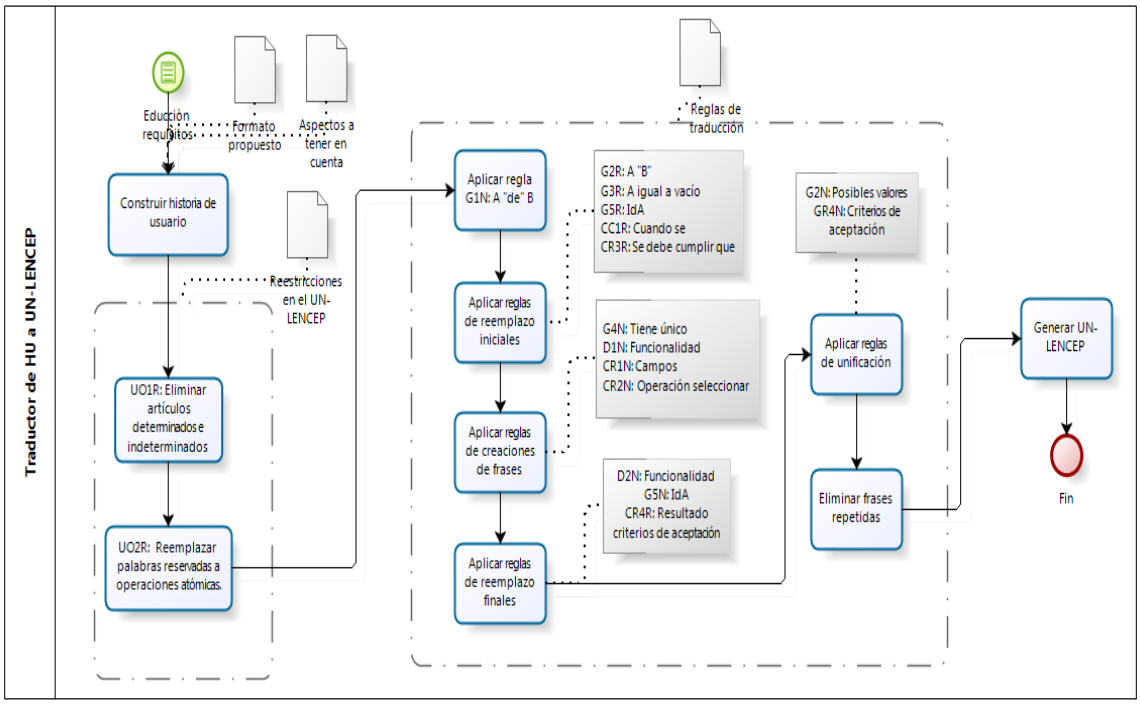
1. Reglas de restricciones de UN-LENCEP.
    - UO1R: Eliminar artículos determinados e indeterminados.
    - UO2R: Reemplazar palabras reservadas a operaciones atómicas.

2. Reglas de traducción de la historia de usuario.
    - G1N: Regla A “de” B.
    - Reglas de reemplazo iniciales.
        - G2R: A “B”.
        - G3R: A igual a vacío.
        - G5R: IdA.
        - CC1R: “Cuando se”.
        - CR3R: “Se debe cumplir que”.
    - Reglas de creaciones de frases.
        - G4N: Tiene único.
        - G5N: IdA.
        - D1N: Funcionalidad.
        - D2N: Funcionalidad.
        - CR1N: Campos.
        - CR2N: Operación seleccionar.
    - Reglas de reemplazo finales:
        - CR4R: Resultado criterios de aceptación.
    - Reglas de unificación:
        - G2N: Posibles valores.
        - GR4N: Criterios de aceptación.

6.3.3.1 Modelo de procesos de aplicación de reglas

Dependiendo de las reglas anteriormente mencionadas y realizando una secuencia de aplicación de las mismas, en la Figura 6-6, se presenta un diagrama del proceso a seguir para la traducción de las historias de usuario a UN-LENCEP.

Figura 6-6 Modelo de procesos de las reglas



6.3.3.2Ejemplo de aplicación de las reglas

Tabla 6-32 Ejemplo de aplicación de las reglas.

<table>
<tr><th colspan="3">Historia de Usuario</th><th>UN-LENCEP</th></tr>
<tr><td>Código</td><td colspan="2">1</td><td rowspan="8">Página web tiene sesión.
Usuario tiene nombre.
Mensaje puede tener los valores: "Usuario inició sesión satisfactoriamente", "Falta usuario o contraseña", "Usuario o contraseña incorrectos".
Ventana "Iniciar sesión" tiene botón iniciar sesión.
Ventana "Iniciar sesión" tiene campo usuario.
Ventana "iniciar sesión" tiene campo contraseña.
Usuario inicia sesión, significa que:
Variable Idusuario igual a campo usuario.
Si usuario ingresa nombre y usuario ingresa contraseña, entonces ventana "iniciar sesión" lista mensaje igual a "Usuario inició sesión correctamente".
Si campo nombre igual a vacío o campo contraseña igual a vacío, ventana "iniciar sesión" lista mensaje igual a "Falta usuario o contraseña".
SI variable Idusuario diferente a campo usuario Y variable Idcontraseña diferente a campo contraseña, ventana</td></tr>
<tr><td>Nombre</td><td colspan="2">Iniciar sesión</td></tr>
<tr><td>Actor</td><td colspan="2">Usuario</td></tr>
<tr><td>Descripción</td><td colspan="2">“Como usuario quiero iniciar sesión de la página Web que sirva para utilizar la funcionalidad específica del sistema para el usuario registrado."</td></tr>
<tr><td rowspan="3">Criterios de aceptación</td><td>Condición</td><td>Resultado</td></tr>
<tr><td>Cuando se inicia sesión de la página Web</td><td>Se debe cumplir que si usuario ingresa nombre de usuario y usuario ingresa contraseña, se presenta mensaje "Usuario inició sesión satisfactoriamente".</td></tr>
<tr><td>Cuando se inicia sesión de la página Web</td><td>Se debe cumplir que si campo nombre de usuario no tiene valor o campo contraseña no tiene valor, se presenta mensaje "Falta usuario o contraseña".</td></tr>
</table>

Tabla 6-33 Continuación ejemplo de aplicación de las reglas.

Historia de Usuario			UN-LENCEP
Criterios de aceptación	Cuando se inicia sesión de la página Web	Se debe cumplir que Idusuario diferente a usuario Y Idcontraseña diferente a contraseña, se presenta mensaje "Usuario o contraseña incorrectos".	"iniciar sesión" lista mensaje igual a "Usuario o contraseña incorrectos".

La traducción del ejemplo anterior, se da a partir del uso de las siguientes reglas en el orden descrito, para cada parte de la historia de usuario.

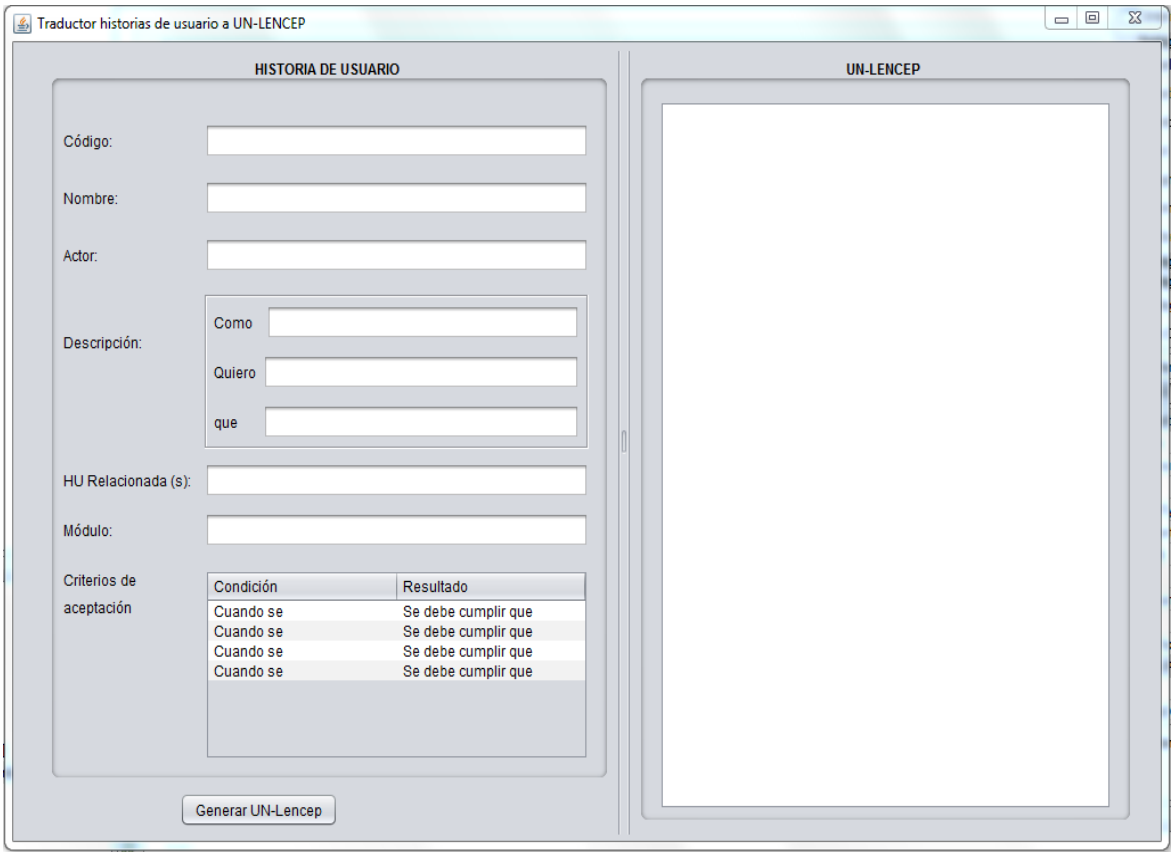
- Para toda la historia de usuario:
  - UO1R: Eliminar artículos determinados e indeterminados.
  - UO2R: Reemplazar palabras reservadas a operaciones atómicas.
  - UO3R: Reemplazar verbos infinitivos a tercera persona
  - G1N: Regla A “de” B.
- Descripción:
  - D1N: Funcionalidad.
- Criterios de aceptación:
  - G3R: A igual a vacío.
  - G5R: IdA.
  - CC1R: “Cuando se”.
  - CR3R: “Se debe cumplir que”.
  - CR1N: Campos.
  - G2N: Posibles valores.
  - D2N: Funcionalidad.
  - G5N: IdA.
  - GR4N: Criterios de aceptación.

6.4 Prototipo funcional y casos de estudio.

6.4.1 UN-HULEN: Traductor de historias de usuario a UN-LENCEP

Con el prototipo funcional UN-HULEN se traducen historias de usuario al lenguaje controlado UN-LENCEP, buscando generar código automáticamente. Esto es posible porque el resultado que se genere en UN-LENCEP (que es el lenguaje textual de los esquemas preconceptuales) constituye la entrada de la herramienta CASE UNC-PSCoder (Chaverra, 2011), que permite diagramar los esquemas, obtener código fuente y generar diagramas de clases, casos de uso y entidad-relación. En la Figura 6-7 se presenta una imagen del prototipo funcional.

Figura 6-7 Traductor de historias de usuario a UN-LENCEP.



El prototipo funcional, tiene la opción de ingresar la historia de usuario en el formato predefinido. Una vez se tenga la historia de usuario, se da *click* en el botón “Generar UN-

LENCEP", que transformará la historia de usuario a UN-LENCEP, por medio de las reglas predefinidas.

6.4.2 Caso de estudio

6.4.2.1Descripción del dominio: Gestión de cuentas de correo

Las historias de usuario pueden variar dependiendo de la construcción del analista. Debido a esto, se debe tener presente que cada analista puede especificar una HU dependiendo de su interpretación de la necesidad del cliente.

Una compañía de publicidad tiene un servidor de correo interno, para un dominio específico, lo cual proporciona a sus trabajadores, permitir enviar mensajes de un usuario a otros, independiente del dominio que usen los demás usuarios. Sin embargo, se vieron en la necesidad de poder gestionar diferentes dominios para una misma cuenta de correo, debido a que sus proveedores o clientes, proporcionaban cuentas internas para relacionarse con la empresa de publicidad.

Para esto se requiere, que el administrador del servidor de correo tenga las diferentes opciones:

- Administrar los dominios de las cuentas de correo (Parametrizar).
- Gestionar las cuentas de correo.
- Consultar las cuentas de correo parametrizadas.

Para ello, se desea que, cuando se consulte una cuenta de correo, se permita gestionarla en la misma pantalla.

Para este ejemplo se cuenta con dos historias de usuario, una para Administrar los dominios de las cuentas y otra para consultar las cuentas de correo parametrizadas en dónde se tendrá la opción de gestionarlas (Véase la Tabla 6 –34 y Tabla 6-35).

Tabla 6-34 Historia de usuario "Administrar dominios".

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Código</td><td colspan="2">001</td></tr>
<tr><td>Nombre</td><td colspan="2">Administrar dominios</td></tr>
<tr><td>Actor</td><td colspan="2">Administrador de servicios</td></tr>
<tr><td>Descripción</td><td colspan="2">Como Administrador de servicios quiero administrar correo de dominio el servidor de correo de la compañía pueda utilizar.</td></tr>
<tr><td>HU Relacionada(s):</td><td colspan="2">002 - Consultar cuentas de correo</td></tr>
<tr><td>Módulo</td><td colspan="2">Gestionar cuentas de correo</td></tr>
<tr><td rowspan="4">Criterios de aceptación</td><td>Condición</td><td>Resultado</td></tr>
<tr><td>Cuando se cambia Lista Capacidad Storage de correo</td><td>Se debe cumplir que Lista Capacidad storage igual a "20000" O Lista Capacidad storage igual a "50000" O Lista Capacidad storage igual a “100000”.</td></tr>
<tr><td>Cuando se elige botón guardar</td><td>Se debe cumplir que si Dominio no tiene valor, se presenta mensaje "El campo Dominio es obligatorio."</td></tr>
<tr><td>Cuando se elige botón guardar</td><td>Se debe cumplir que si Cantidad máxima de correo no tiene valor, se presenta mensaje "El campo Cantidad máxima es obligatorio."</td></tr>
</table>

Tabla 6-35 Continuación historia de usuario "Administrar dominios".

Historia de Usuario		
	Condición	**Resultado**
Criterios de aceptación	Cuando se elige botón guardar	Se debe cumplir que si Usuario de dominio no tiene valor, se presenta mensaje "El campo Usuario es obligatorio."
	Cuando se elige botón guardar	Se debe cumplir que si Password de dominio no tiene valor, se presenta mensaje "El campo Password es obligatorio."
	Cuando se elige botón guardar	Se debe cumplir que Si IdDominio igual a Dominio y IdcantidadMaxCorreo igual a Cantidad máxima de correo y IdcantidadMínCorreo igual a Cantidad mínima de correo y Idusuario igual a Usuario y Idpassword igual a Password se guarda dominio.

Para la anterior historia de usuario se genera el siguiente UN-LENCEP:

Dominio tiene correo.

Dominio tiene usuario.

Dominio tiene password.

Correo tiene servidor.

Correo tiene cantidad máxima.

Correo tiene cantidad mínima.

Correo tiene Capacidad storage.

Capacidad storage puede tener los valores: "10000", "20000", "50000".

Mensaje puede tener los valores: "El campo Dominio es obligatorio", "El campo Cantidad máxima es obligatorio", "El campo Cantidad mínima es obligatorio.", "El campo Usuario es obligatorio", "El campo Password es obligatorio."
Ventana "Administrar dominios" tiene Lista Capacidad Storage.
Ventana "Administrar dominios" tiene campo Dominio.
Ventana "Administrar dominios" tiene campo Cantidad máxima.
Ventana "Administrar dominios" tiene campo Cantidad mínima.
Ventana "Administrar dominios" tiene campo Usuario.
Ventana "Administrar dominios" tiene campo Password.
Ventana "Administrar dominios" tiene botón guardar.
Administrador de servicios edita Lista Capacidad Storage, Significa que:
Lista Capacidad storage igual a "20000" O Lista Capacidad Storage igual a "50000" O Lista Capacidad storage igual a "10000".
Administrador de servicios administra dominio, Significa que:
Variable IdDominio igual a campo Dominio.
Variable IdcantidadMaxCorreo igual a campo Cantidad máxima.
Variable IdcantidadMínCorreo igual a campo Cantidad mínima.
Variable Idusuario igual a campo Usuario.
Variable Idpassword igual a campo Password.
Administrador de servicios selecciona botón guardar, Significa que:
Si Dominio igual a vacío, entonces ventana "Administrar dominios" lista mensaje igual a "El campo Dominio es obligatorio".
Si Cantidad máxima igual a vacío, entonces ventana "Administrar dominios" lista mensaje igual a "El campo Cantidad máxima es obligatorio".
Si Cantidad mínima igual a vacío, entonces ventana "Administrar dominios" lista mensaje igual a "El campo Cantidad mínima es obligatorio".
Si Usuario igual a vacío, entonces ventana "Administrar dominios" lista mensaje igual a "El campo Usuario es obligatorio".
Si Password no tiene valor, entonces ventana "Administrar dominios" lista mensaje igual a "El campo Password es obligatorio".
Si variable IdDominio igual a campo Dominio y variable IdcantidadMaxCorreo igual a campo Cantidad máxima y variable IdcantidadMínCorreo igual a Cantidad mínima y

variable Idusuario igual a campo Usuario y variable Idpassword igual a campo Password entonces ventana "Administrar dominios" inserta dominio.

El esquema preconceptual creado a partir del UN-LENCEP generado se presenta en la Figura 6-8 y Figura 6-9.

Figura 6-8 Esquema preconceptual "Administrar Dominios".

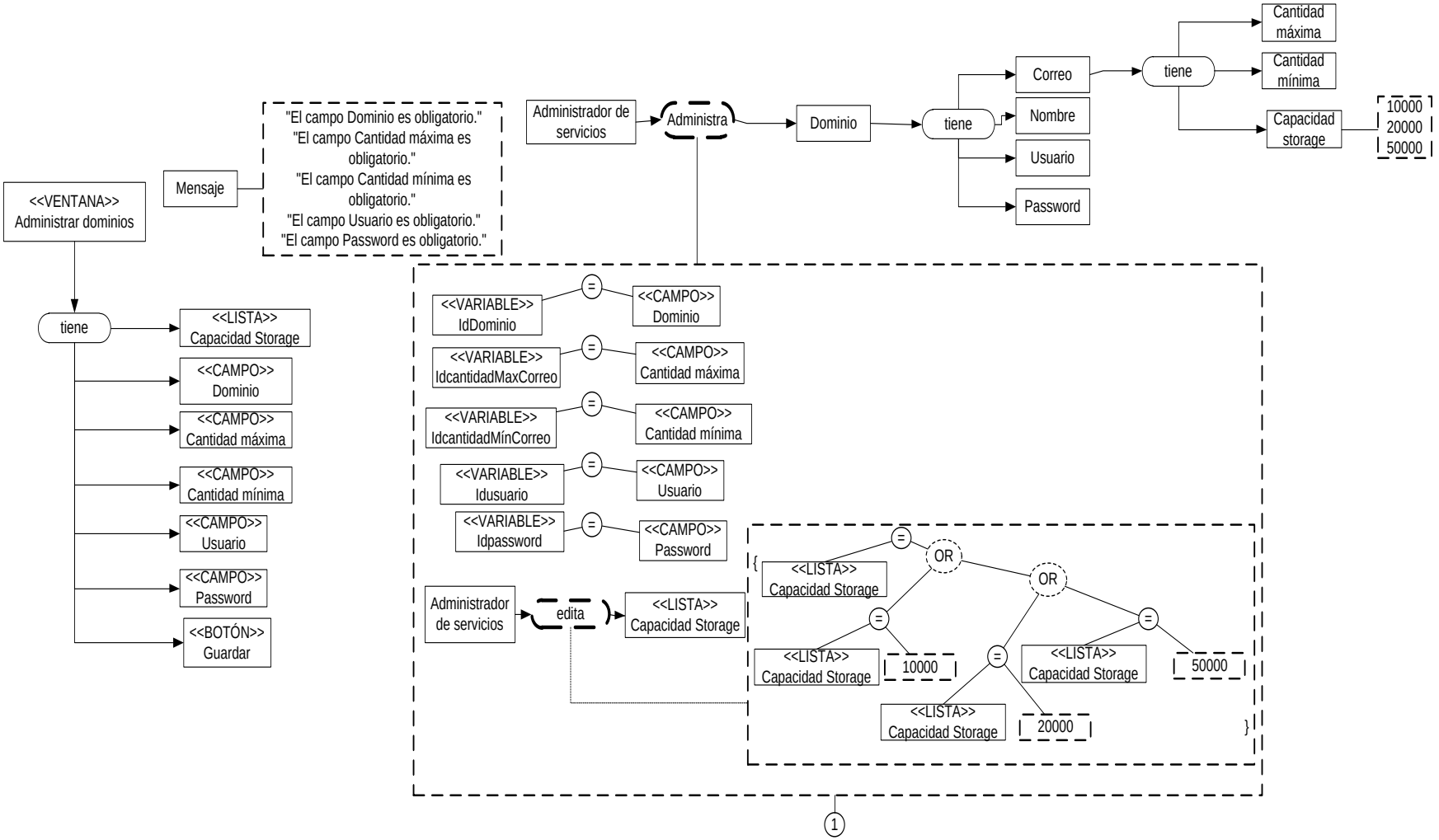
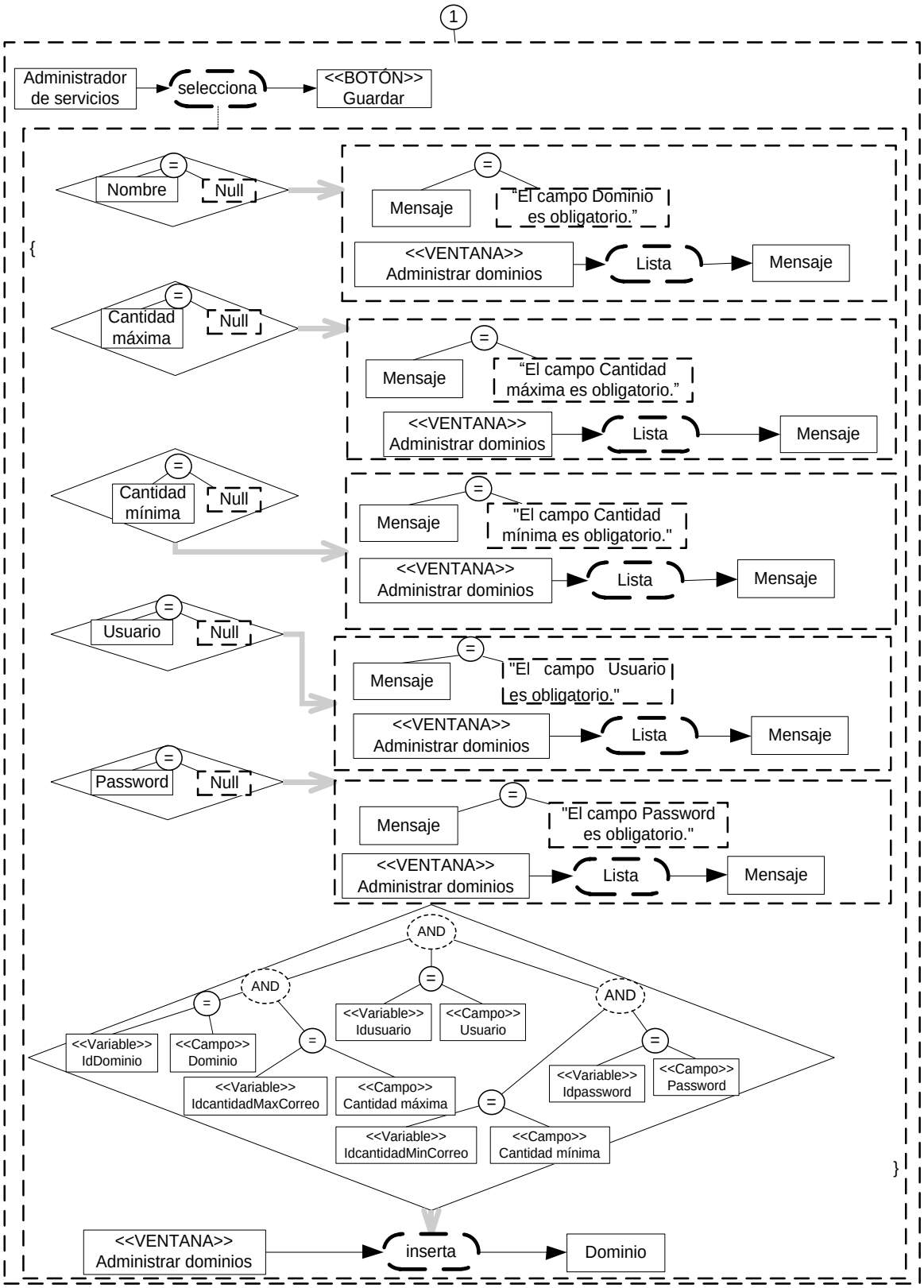


Figura 6-9 Continuación especificación "Administrar Dominios" en EP.



- Consultar cuentas de correo.

Tabla 6-36 Historia de usuario "Consultar cuentas de correo".

Historia de Usuario		
Código	002	
Nombre	Consultar cuentas de correo	
Actor	Administrador de servicios	
Descripción	Como Administrador de servicios quiero consultar cuenta de correo que se encuentre en el sistema de la compañía para gestionarla.	
HU relacionadas	001 – Administrar cuentas de correo.	
Módulo	Gestionar cuentas de correo	
Criterios de aceptación	**Condición**	**Resultado**
	Cuando se consulta cuenta de correo,	se debe cumplir que si campo Usuario es vacío y campo Dominio es vacío, se presenta mensaje "Debe ingresar un usuario o un dominio"
	Cuando se consulta cuenta de correo,	Se debe cumplir que si Idusuario es igual a Usuario o Iddominio es igual a Dominio, se visualiza Tabla "Listado de cuentas de correo registradas".
	Cuando se visualiza tabla "Listado de cuentas de correo registradas",	Se debe cumplir que checkbox Seleccione es vacío.
	Cuando se visualiza tabla "Listado de cuentas de correo registradas",	Se debe cumplir que Idusuario igual a Usuario Idnombredecuenta igual a Cuenta, Iddominiosea igual a Dominio, IdNit igual a Nit, Idestado igual a Estado y Idobservación sea igual a Observación.

Tabla 6-37 Continuación historia de usuario "Consultar cuentas de correo".

Historia de Usuario		
Criterios de aceptación	**Condición**	**Resultado**
	Cuando se consulta cuenta de correo,	Se debe cumplir que si estado de cuenta igual a "Pendiente confirmación" o estado de cuenta igual a "Activa" se presenta Campo Observación igual a vacío.
	Cuando se consulta cuenta de correo,	se debe cumplir que si estado de cuenta igual a "Pendiente confirmación" o estado de cuenta igual a "Activa" se presenta botón Borrar Cuenta Y se presenta botón Borrar Dominio
	Cuando se elige botón Borrar Cuenta	se debe cumplir que si Administrador de servicios chequea el checkbox Seleccione, se presenta campo Observación
	Cuando se cambia Observación	Se debe cumplir que, Idobservación diferente a vacío
	Cuando se elige botón Borrar Cuenta	se debe cumplir que si Observación diferente a vacío, se cambia Estado de cuenta igual a "Borrado" y borra cuenta de correo
	Cuando se elige botón Borrar Dominio,	se debe cumplir que si estado de cuenta diferente a "Borrado", se presenta mensaje "El dominio no puede ser eliminado, ya que tiene por lo menos una cuenta Activa"
	Cuando se elige botón Borrar Dominio,	se debe cumplir que si estado de cuenta igual a "Borrado", se borra dominio

Para la anterior historia de usuario, se genera el siguiente UN-LENCEP:

Correo tiene cuenta.
Cuenta tiene estado.
Dominio tiene correo.
Tabla igual a "Listado de cuentas de correo registradas".
Estado puede tener los valores: "Activa", "Pendiente confirmación", "Borrado".
Ventana "Consultar cuentas de correo" tiene botón Consultar.
Ventana "Consultar cuentas de correo" tiene campo Usuario.
Ventana "Consultar cuentas de correo" tiene campo Cuenta.
Ventana "Consultar cuentas de correo" tiene campo Dominio.
Ventana "Consultar cuentas de correo" tiene campo Nit.
Ventana "Consultar cuentas de correo" tiene campo Estado.
Ventana "Consultar cuentas de correo" tiene campo Observación.
Ventana "Consultar cuentas de correo" tiene Checkbox Seleccione.
Ventana "Consultar cuentas de correo" tiene botón Borrar cuenta.
Ventana "Consultar cuentas de correo" tiene botón Borrar dominio.
Administrador de servicios consulta cuenta, significa que:
Variable Iddominio igual a campo Dominio.
Variable Idusuario igual a campo Usuario.
Variable Idnombredecuenta igual a campo Cuenta.
Variable IdNit igual a campo Nit.
Variable Idestado igual a campo Estado.
Variable Idobservación igual a campo Observación.
SI campo Usuario igual a vacío y campo Dominio igual a vacío, entonces Ventana "Consultar cuentas de correo" lista mensaje igual a "Debe ingresar un usuario o un dominio".
Si Variable Idusuario igual a campo Usuario o variable Iddominio igual a campo Dominio, entonces Ventana "Consultar cuentas de correo" lista Tabla "Listado de cuentas de correo registradas".
Si Variable Idusuario igual a campo Usuario o Variable Iddominio igual a campo Dominio, entonces Ventana "Consultar cuentas de correo" lista cuenta igual a .LAP.

Si estado igual a "Pendiente confirmación" o estado igual a "Activa" entonces Ventana "Consultar cuentas de correo" lista Campo Observación igual a vacío.
Si estado igual a "Pendiente confirmación" o estado igual a "Activa" entonces Ventana "Consultar cuentas de correo" lista botón Borrar Cuenta Y Ventana "Consultar cuentas de correo" lista botón Borrar Dominio.
Administrador de servicios lista tabla "Listado de cuentas de correo registradas", significa que:
Checkbox Seleccione igual a vacío.
Variable Idusuario igual a campo Usuario, Variable Idnombredecuenta igual a campo Cuenta, Variable Iddominio igual a campo Dominio , Variable IdNit igual a campo Nit, Variable Idestado igual a campo Estado y Variable Idobservación igual a campo Observación.
Administrador de servicios selecciona botón Borrar Cuenta, significa que:
Si Administrador de servicios selecciona checkbox Seleccione, entonces Ventana "Consultar cuentas de correo" lista campo Observación.
Administrador de servicios edita observación, significa que:
Variable Idobservación diferente a vacío.
Administrador de servicios selecciona botón borrar cuenta, significa que:
Si Observación diferente a vacío, entonces Ventana "Consultar cuentas de correo" edita estado igual a "Borrado" y Ventana "Consultar cuentas de correo elimina cuenta.
Administrador de servicios selecciona botón Borrar Dominio, significa que:
Si estado diferente a "Borrado", entonces Ventana "Consultar cuentas de correo" lista mensaje igual a "El dominio no puede ser eliminado, ya que tiene por lo menos una cuenta Activa".
Si estado igual a "Borrado", entonces Ventana "Consultar cuentas de correo" elimina dominio.

El esquema preconceptual creado a partir del UN-LENCEP generado se presenta en la Figura 6-10, Figura 6-11 y Figura 6-12.

Figura 6-10 Esquema preconceptual "Consultar Cuentas".

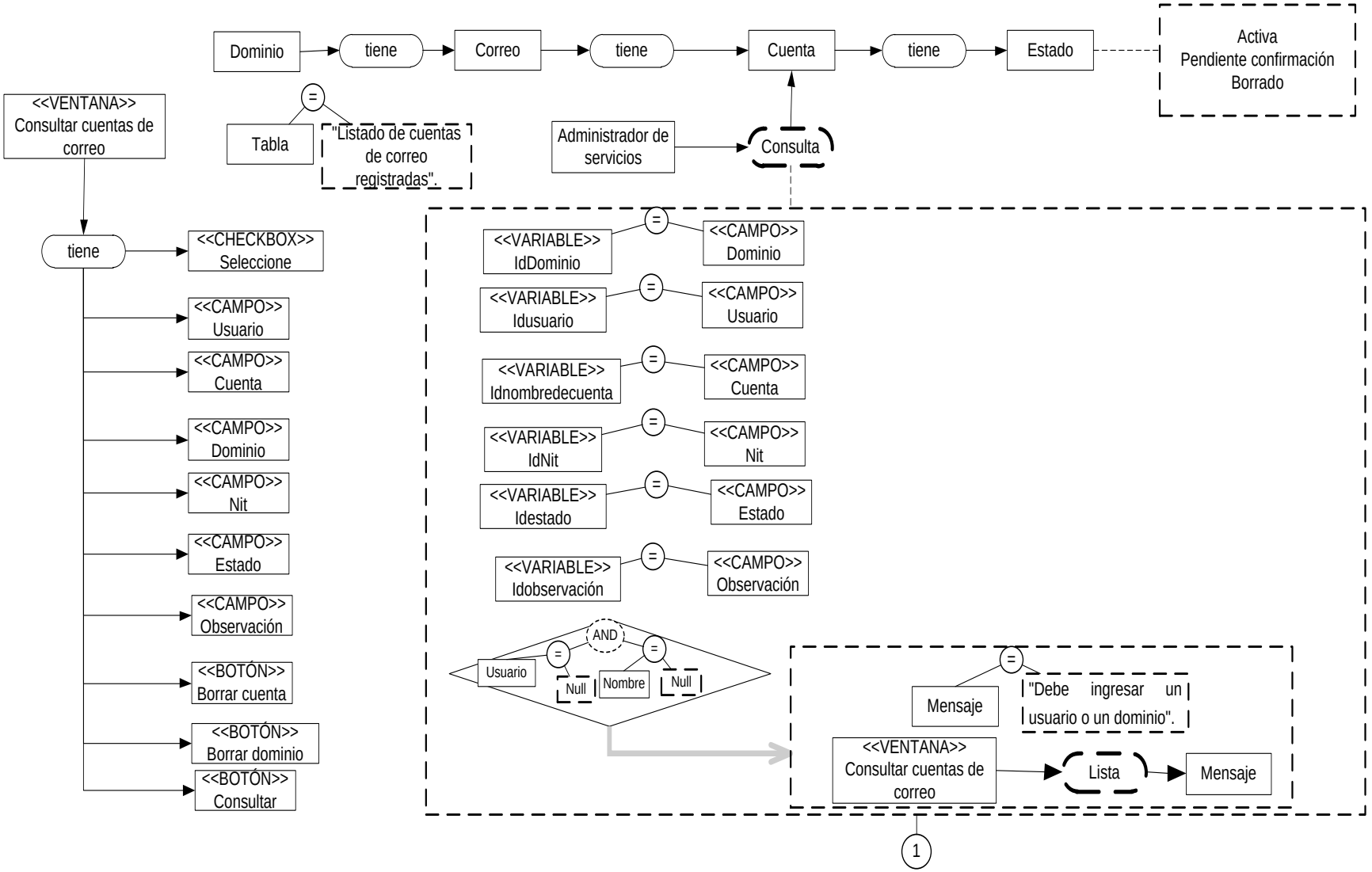


Figura 6-11 Continuación especificación "Consultar Cuentas" en EP.

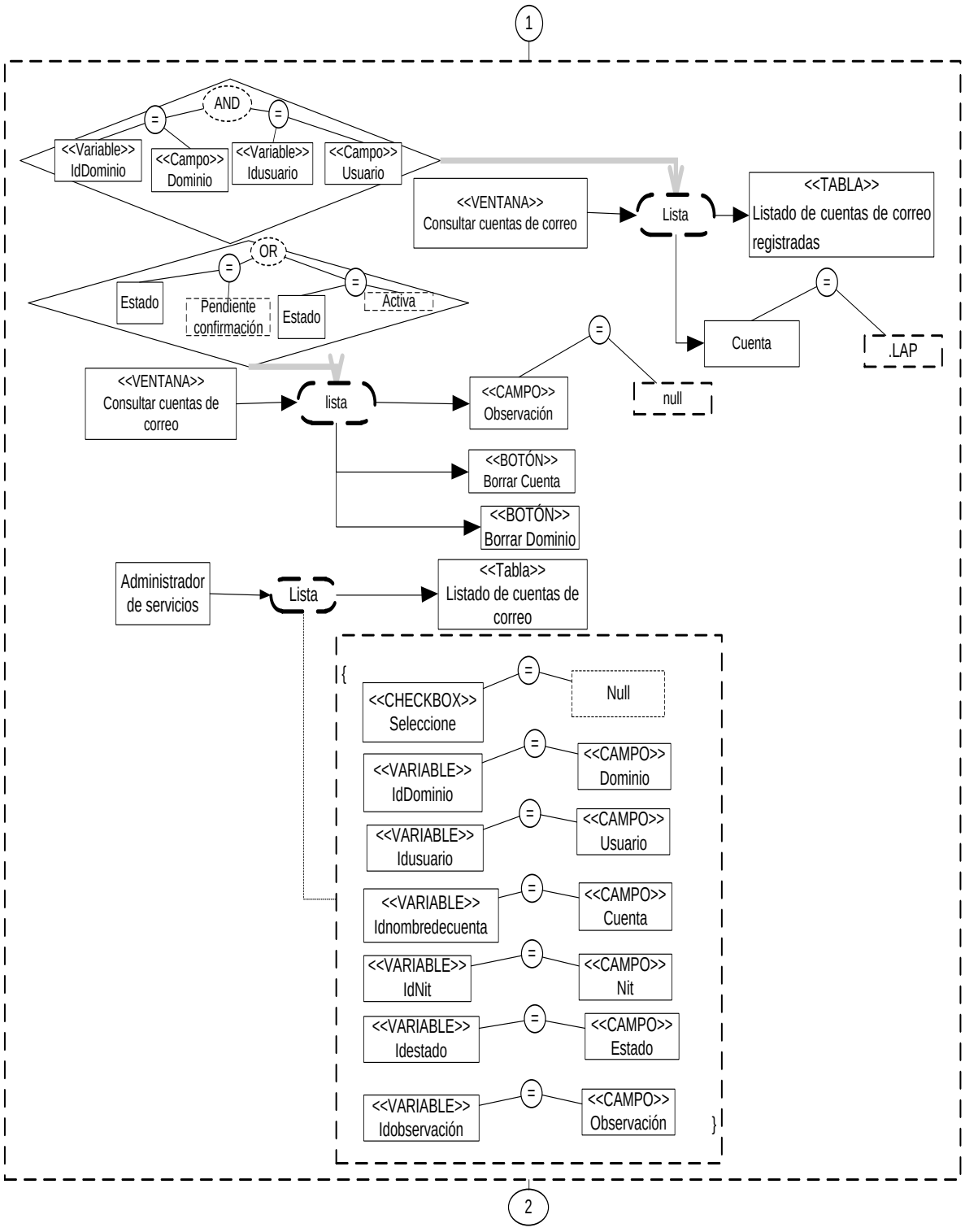
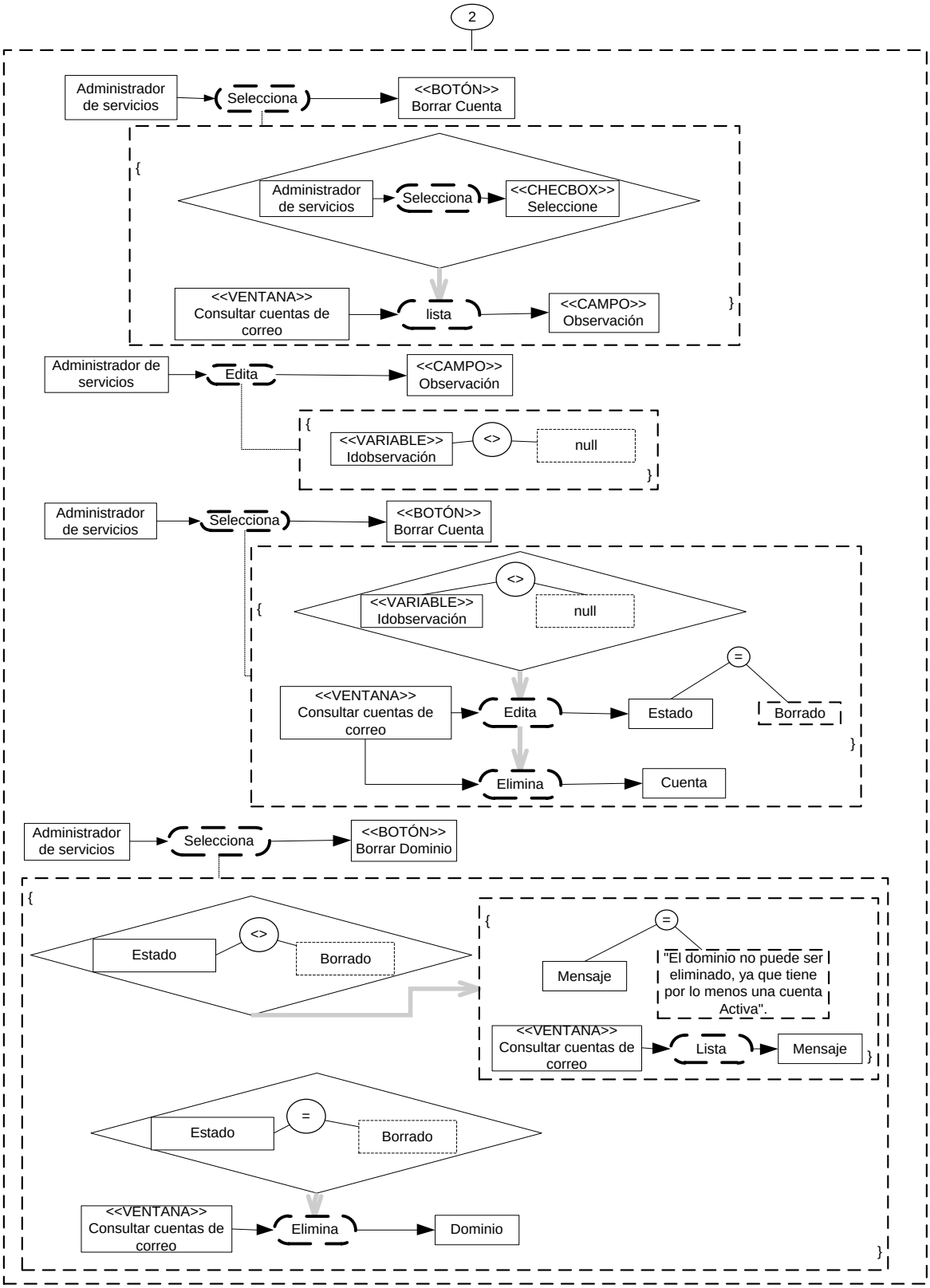


Figura 6-12 Continuación especificación "Consultar Cuentas" en EP.



7. Conclusiones y recomendaciones

7.1 Conclusiones

Las metodologías ágiles, al ofrecer una alternativa de flexibilidad en los procesos de desarrollo, optaron por dejar a un lado gran cantidad de información requerida para el soporte de una aplicación. Con la traducción de historias de usuario a UN-LENCEP, se logra disminuir el tiempo requerido para la construcción de código a la hora de su implementación, ya que el UN-LENCEP como lenguaje de los esquemas preconceptuales genera código automáticamente y a raíz del mismo una documentación robusta. Esto se debe a trabajos y aportes anteriores como lo es la herramienta CASE UNC-PSCoder (Chaverra, 2011).

La definición de reglas de traducción garantiza que la información sea consistente desde el proceso de adquisición de requisitos hasta su implementación. No obstante, se requiere que el analista de requisitos realice una educción completa y una especificación de las historias de usuario de una manera inambigua.

La especificación de una estructura definida para las historias de usuario, mejora la comunicación entre los analistas de requisitos y los interesados, ya que se define un lenguaje controlado que es el UN-LENCEP que elimina las ambigüedades de las historias de usuario, proporcionando un lenguaje común entre ambas partes.

7.2 Recomendaciones

A pesar de los aportes realizados en esta Tesis, aún se presentan limitantes de los cuales se pueden realizar diferentes investigaciones y continuar un trabajo en pro de la mejora continua de nuestra problemática. A continuación se presentan algunas de ellas:

- Definir nuevas reglas que permitan la traducción de plurales a UN-LENCEP.

- Definir reglas que permitan la especificación de comas (,) sin necesidad de conjunciones y disyunciones en UN-LENCEP.

- Definir nuevas reglas que permitan identificar los tipos de datos, tamaño de caracteres y valores por defecto de los conceptos en el lenguaje UN-LENCEP.

- Construir una aplicación robusta para la traducción de las historias de usuario que se pueda comunicar directamente con la herramienta CASE UNC-PSCoder y así generar código directamente desde la especificación de las historias de usuario.

7.3 Publicaciones relacionadas

Título	Ontología Para La Definición De Un Formato Estándar Para Las Historias De Usuario.
Autores	Gloria Lucía Giraldo Gómez Carlos Mario Zapata Katerine Villamizar

Resumen:

Las historias de usuario son especificaciones realizadas por un interesado, sobre las características que debe tener un sistema. Diversos intentos por utilizar metodologías de desarrollo ágil, usan historias de usuario como especificaciones de requisitos para la construcción de un producto de software. En dichos intentos por lograr estas especificaciones, se emplean historias de usuario sin un estándar definido, lo que hace

que una historia de usuario pueda ser o “muy compleja” que el interesado no la entienda o “muy sencilla” que no pueda construir un producto de software a partir de ella. Por esta razón, en el presente artículo se propone la construcción de una ontología como especificación del conocimiento que permita generar un formato estándar para la creación de historias de usuario de un dominio específico.

Título	Mejora de historias de usuario y casos de prueba de metodologías ágiles con base en TDD *Improved user stories and test cases based Agile TDD*
Autores	Katerine Villamizar Suaza John Jairo Tabares García Carlos Mario Zapata Jaramillo

Resumen:

Las historias de usuario se utilizan en las metodologías ágiles para especificar los requisitos de una aplicación de software. El desarrollo dirigido por pruebas (TDD por sus siglas en inglés) es una técnica usada en las metodologías ágiles que consiste en generar pruebas unitarias automáticas basadas en las historias de usuario. Sin embargo, esta técnica presenta dificultades en la confiabilidad de las pruebas funcionales integrales y en la especificación de las historias de usuario. Diversos investigadores proponen nuevas técnicas y metodologías para mejorar las pruebas funcionales de software y las historias de usuario, pero no solucionan directamente la problemática de la técnica TDD. Por ello, en este artículo se propone una mejora a las historias de usuario que integre elementos de los casos de prueba. Esta mejora se ejemplifica con un caso de estudio.

Referencias

Amaro, S., Valverde, J. (2007). *Metodologías Ágiles*. Universidad Nacional de Trujillo. Perú.

Beck, K. (2000) Extreme Programming Explained. Embrace Change. Pearson Education. Addison Wesley.

Blé, J., C. y Colaboradores (2010, Ene.). *Diseño Ágil con TDD*. (1ra Ed.). Creative Commons.

Canós, J. H., Letelier P., Penadés, M. C. (2008). Métodologías Ágiles en el Desarrollo de Software. DSIC -Universidad Politécnica de Valencia Camino de Vera s/n, 46022 Valencia.

Chaverra, John J. (2011). *Generación automática de prototipos funcionales a partir de esquemas preconceptuales.* (Tesis de maestría). Universidad Nacional de Colombia. Colombia.

Citón, M. L. (2006). *Método Ágil Scrum Aplicado Al Desarrollo De Un Software De Trazabilidad* (Tesis). EC. Consultado el 22 de Noviembre del 2013. (Formato PDF) Universidad De Mendoza Facultad de Ingeniería Ingeniería en Informática. 2006. Disponible en la url: http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L01ldG9kb3NfQWdpbGVzL01ldG9kb19BZ2lsX1NjcnVtLnBkZg%3D%3D&cidReset=true&cidReq=II0162004

Cohn, M. (2004). *User Stories Applied. For agile software development*. Addison Wesley. Boston.

Cockburn, A. (2000, Jun.). *Characterizing People as Non-Linear, First-Order Components in Software Development*. Presented at the 4th International Multi-Conference on Systems, Cybernetics and Informatics, Orlando, EE.UU.

Cooper, R. y Sawaf, A. (2005). *La Inteligencia Emocional aplicada al liderazgo y a las organizaciones*. Colombia: Editorial Norma.

Echeverry T. L. M., Delgado C. Luz E. (2007, Oct.). *Caso práctico de la metodología XP. Al desarrollo de software*. (Tesis de ingeniería). Universidad tecnológica de Pereira. Facultad de Ingeniería. Pereira.

Gutiérrez, J. J., Escalona M. J., Mejías M. y Torres J. (2005). Pruebas del Sistema en programación Extrema. Departamento de Lenguajes y Sistemas Informáticos. University of Sevilla. Art. Nro. 96. Javahispano.org.

Isaacson Walter. (2011). Steve Jobs. Simon & Schuter. New York. Pág 564.

Jeffries, R., Anderson, A., Hendrickson, C. (2001). *Extreme Programming Installed*. Addison-Wesley.

Jeffries, Ron. (2001, 30 Ago). *Essential XP: Card, Conversation, and Confirmation*. XP Magazine.

Lamsweerde, A., Dardenne, A., Fickas S. (1993). Goal-directed requirements acquisition. USA:ElSeVIER. Science of Computer Programming, Vol. 20. p. 3-50.

Leffingwell, D. (2012). Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. (1ra Ed.). United States: Addison Wesley.

Letelier P., Penadés, Mª C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación. Facultad de Informática. Universidad Politécnica de Valencia. Técnica Administrativa, Buenos Aires ISSN 1666-1680.

Meneses G., Peñalver, G. A., (2010, May.). SXP, Metodología Ágil Para El Desarrollo De Software. Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.

Noy, N. F., McGuinness, D. L. (2001). "OntologyDevelopment 101: A Guide toCreatingYourFirstOntology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March.

Sánchez P. E. A., Caños L. J. (2010). Mejorando la gestión de historias de usuario en eXtreme Programming. Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia Camino de Vera s/n 46022. Valencia - España.

Schenone, M. H. (2004). *Diseño de una Metodología Ágil de Desarrollo de Software*. (Tesis de Grado en Ingeniería en Informática). Facultad de Ingeniería. Universidad de Buenos Aires.Fiuba.

Sommerville, I. (2005). Integrated Requirements Engineering. *IEEE Software,* 22 (1), 16-23, January/February.

Yagüe, A. y Garbajosa, J. (2009) "Las pruebas en metodologías ágiles y convencionales: papeles Diferentes". Universidad Politecnica de Madrid (UPM). Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 3, No. 4.

Zapata, C. M. (2007). *Definición de un esquema preconceptual para la obtención automática de esquemas conceptuales de UML*. (Tesis de doctorado). Universidad Nacional de Colombia, Sede Medellín.

Zapata, C. M. Ruiz, L. M. y Villa, F. A. (2007). *UNC-Diagramador una herramientas upper CASE para la obtención de diagramas UML desde esquemas preconceptuales.* Revista Universidad EAFIT, vol. 43, No. 147, pp. 68-80. Colombia.

Zapata, C. M, Arango I. F., Gelbukh, A. (2007). UN–Lencep: Obtención Automática de Diagramas UML a partir de un Lenguaje Controlado. Colombia.

Zapata, C. M. Garcés, G. L. (2008, Dic.). Generación Del Diagrama De Secuencias De Uml 2.1.1 Desde Esquemas Preconceptuales. Revista EIA, ISSN 1794-1237 Número 10, p. 89-103.

Zapata, C. M, Chaverra M. J.J., Zapata, B. (2010). Generación Automática de Código a Partir del Lenguaje Controlado UN-Lencep. CISCI.