INTEGRACIÓN DE Spectrum-E® A LA SUITE TESGestion® EMPLEANDO METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE

OSCAR FELIPE PÉREZ MONTAÑA

Trabajo de grado presentado como requisito para obtener el título de
INGENIERO ELECTRÓNICO

Director:
ESP. SANDY ENRIQUE AVELLA CELY

Universidad Pedagógica y Tecnológica De Colombia
Facultad Sede Seccional Sogamoso
Escuela De Ingeniería Electrónica
Noviembre de 2016

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

ESP. SANDY ENRIQUE AVELLA CELY

Director

_____

JURADO

_____

JURADO

Universidad Pedagógica y Tecnológica De Colombia
Escuela De Ingeniería Electrónica
Facultad Sede Seccional Sogamoso

Noviembre de 2016

Agradecimientos

En estas líneas quiero expresar mi más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo.

Primero agradezco a Dios por las bendiciones recibidas, por permitirme llegar a esta instancia de mi vida, por los triunfos y las dificultades que me han hecho fortalecer mi carácter, me han enseñado a levantarme con más ánimo y hacen que valore cada día más este momento.

Agradezco a la UPTC y a todos los docentes por sus enseñanzas, porque parte de lo que soy como profesional y como persona se lo debo a ellos, al personal administrativo por su gestión, atención y colaboración en las múltiples ocasiones que requerí de sus servicios. Igualmente quiero hacer un especial agradecimiento al Dr. Jorge Eduardo Londoño por su gestión en la Gobernación de Boyacá y por el programa de becas en convenio con la universidad.

Especial reconocimiento merece el interés mostrado hacia mi trabajo, por su acompañamiento y las sugerencias recibidas por parte del Esp. Sandy Enrique Avella. Extiendo mi agradecimiento al Ing. Daniel Rosas Tapia por brindarme la oportunidad de ser parte de la familia TES America, a los líderes del área de I+D, Juan León y Andrés Aguirre, por sus enseñanzas y el conocimiento compartido, y en general a todo el talento humano.

Hago extensiva mi gratitud a mis compañeros de la Escuela de Ingeniería Electrónica, especialmente a Ivan Alfonso y a Andrés Rojas porque junto a ellos me formé y crecí no solo en el ámbito profesional sino que también me marcaron como persona, me preparé para la vida aprendiendo de cada uno de ellos muchas cosas, gracias por su apoyo y colaboración.

A mi familia por el apoyo brindado, en especial a mi madre quien en medio de tantas dificultades me ha ayudado a conquistar esta meta, a mis hermanos por su apoyo en tantos aspectos y a mis demás familiares por sus consejos y enseñanzas. A mis amigos por la comprensión, paciencia y el ánimo recibidos.

A todos ellos, muchas gracias.

Dedicatoria

A mi padre fallecido:
Porque gran parte de lo que soy se lo debo a él,
desde mis primeros pasos él estuvo siempre a mi lado
siendo mi ejemplo a seguir, mi héroe y hoy aún sigue acompañándome,
desde el cielo me da fuerzas para seguir día a día.

A mi madre:
Quien ha hecho grandes sacrificios por mí y por mis hermanos
por sus valiosos consejos, porque siempre ha buscado
la manera de estar a mi lado velando y cuidando mis pasos.

"La INNOVACIÓN es lo que distingue
a un LÍDER de los demás"
Steve Jobs

Índice General

1. GENERALIDADES **1**
1.1. Introducción . . . 1
1.2. Identificación Del Problema . . . 2
1.3. Objetivos . . . 3

2. MARCO TEÓRICO **4**
2.1. Metodologías de Desarrollo . . . 4
2.1.1. Metodologías Tradicionales . . . 5
CMMI . . . 6
2.1.2. Metodologías Ágiles . . . 9
SCRUM . . . 12
2.2. Conceptos de Programación . . . 20
2.2.1. Programación Orientada a Objetos . . . 20
Conceptos de POO . . . 21
2.2.2. Bases de Datos . . . 22

3. DESARROLLO **25**
3.1. Simulación Enlaces Nuevos . . . 25
3.1.1. Capa de Presentación . . . 28
3.1.2. Capa de Datos . . . 29
3.1.3. Capa de Entidades . . . 30
3.1.4. Capa de Acceso a Datos . . . 31
3.1.5. Capa de Lógica . . . 33
3.2. Simulación Enlaces Existentes y Múltiples Enlaces . . . 36
3.2.1. Capa de Presentación . . . 36
3.2.2. Capa de Datos . . . 37
3.2.3. Capa de Acceso a Datos . . . 38
3.2.4. Capa de Lógica . . . 39
3.2.5. Simulación Múltiples Enlaces . . . 40

4. RESULTADOS **41**
4.1. Análisis y Resultados del los Sprints . . . 41
4.1.1. Sprint 1: Simulación Enlaces Nuevos . . . 41
4.1.2. Sprint 2: Simulación Enlaces Existentes y Múltiples Enlaces . . . 43
4.2. Análisis y Resultados del Desarrollo . . . 45

5. CONCLUSIONES Y TRABAJO FUTURO **49**

Índice Alfabético **50**

Lista de Acrónimos **51**

Bibliografía **52**

Índice de Tablas

2.1. Comparación de metodologías. 10
2.2. Elementos de SCRUM 13
2.3. Relación del SM con los demás roles 15

3.1. Columnas Tabla Resultados. 29
3.2. Propiedades clase Resultados. 30
3.3. Procedimientos Agregados a la capa DAL 33

4.1. Enlaces Simulados 47

Índice de Figuras

2.1. Componentes CMM 7
2.2. Ciclo de vida y costos de cambio 10
2.3. Marco de SCRUM 13
2.4. Formato historias de usuario 17
2.5. Sprint Burndown Chart 18
2.6. Modelamiento del mundo real a OOP 20
2.7. OOP vs Estructurada 21
2.8. SQL joins 24

3.1. Capturadores datos creación enlaces en TesGestion 25
3.2. Diagrama general sistema 27
3.3. Diagrama de capas 28
3.4. Navegador de simulación en Visual Studio 28
3.5. Tabla de resultados en base de datos 29
3.6. Diagrama de clases para los datos del enlace 31
3.7. Diagrama de flujo validación 36
3.8. Botones en ventana de enlaces 37

4.1. Burndown Chart Sprint 1 41
4.2. Tablero Scrum Sprint 1 42
4.3. Scrum Burndown Chart Sprint 2 43
4.4. Tablero Scrum Sprint 2 44
4.5. Navegador de Final Creación 45
4.6. Simulación en Spectrum-E 46
4.7. Enlaces simulados en el tiempo 47

Lista de Segmento de Códigos

3.1. Codigo SQL actualización tabla Enlaces 30
3.2. Instanciación Base de Datos 31
3.3. Comando de lectura de registros Base de Datos 31
3.4. Ejecución de procedimiento almacenado 32
3.5. Ejecución de una transacción SQL 32
3.6. Constructor ventana simulación 34
3.7. Click botón Simular 34
3.8. Click botón Anexar 34
3.9. Creación de botones en ventana de Enlaces 36
3.10. Actualización procedimientos almacenados 37
3.11. Creación DatosEnlace desde procedimiento almacenado 38
3.12. Constructor clase simulación para enlace existente 39

Capítulo 1

GENERALIDADES

1.1. Introducción

En los últimos años se ha presentado un importante crecimiento en la demanda del Espectro Radio-Eléctrico (ERE), el uso incrementado que se hace de las tecnologías basadas en las radiocomunicaciones y el enorme impacto social, económico, cultural y estratégico que estas generan, demuestran la importancia de este recurso así como de los procesos de gestión del mismo. Los avances tecnológicos han generado una diversidad de nuevas aplicaciones que requieren del uso de este recurso lo cual repercute en un mayor interés hacia el mismo. Al incrementarse la demanda se hace necesario un uso más eficiente del espectro y la implementación de procesos de gestión más eficaces. Tomando en cuenta lo anterior, se destacan los métodos de análisis de ingeniería y el tratamiento de los datos para poder soportar a la gran cantidad de posibles usuarios que pretenden acceder a ese recurso especialmente para la prestación de servicios de telecomunicaciones.

Como ejemplo para visualizar la necesidad de la gestión eficiente del espectro radioeléctrico, supongase el caso de un servicio como la radio. En el equipo receptor lo que se hace es sintonizar una frecuencia específica, por ejemplo 106.6 MHz, en caso de no existir control alguno de la frecuencia de transmisión de cada emisora, al intentar sintonizarla, se podría llegar a tener varias fuentes de señal adicionales a la de interés para el usuario, lo cual ocasionaría interferencia sobre esta y pérdida de la información, de este modo sería sumamente complicado recuperar el mensaje original; ahora tomando el caso de la telefonía móvil, el Internet u otro servicio de mayor impacto el problema sería aún más grave (Muñoz et al., 2015).

Se pueden distinguir tres tipos de interferencias desde el punto de vista de la gestión del espectro. El primer tipo son las llamadas interferencias admisibles, que son aquellas interferencias observadas o previstas que satisfacen los criterios cuantitativos de interferencia y de compartición que están presentes en el Reglamento de Radiocomunicaciones publicado por la International Telecommunication Union (ITU) o en acuerdos especiales previstos en dicho reglamento. Estas interferencias tienen la particularidad que el efecto sobre los distintos servicios que afectan son casi despreciables y por lo tanto no ponen en riesgo el buen funcionamiento de los distintos servicios, puesto que se han tenido en cuenta en la fase de diseño de los mismos.

En segundo lugar se encuentran las interferencias aceptadas, que son aquellas interferencias de mayor nivel que las definidas como admisibles, pero que son acordadas explícitamente entre dos o más administraciones. La particularidad de estas interferencias es que su efecto es tal que no degrada la prestación de los servicios en cuestión y típicamente simplifica la prestación de los mismos. Finalmente, las interferencias perjudiciales son aquellas que representan un riesgo considerable para el funcionamiento de algún servicio que usa el espectro radioeléctrico. En la práctica significa que se degrade u obstruya gravemente o interrumpa de forma repetida un servicio radio que funcione

de conformidad con la reglamentación que corresponda.

En Colombia los organismos encargados de la gestión manejo y regulación del ERE son:el Ministerio de las Tecnologías de la Información y las Comunicaciones (MinTIC), la Agencia Nacional del Espectro (ANE) y la Autoridad Nacional de Televisión (ANTV) (Comisión Nacional de Televisión (CNTV) antes del 2012) para el caso específico del espectro electromagnético atribuido al servicio de TV. Así pues, para poder hacer uso del espectro radio eléctrico se requiere permiso previo y expreso otorgado por el MinTIC, el cual debe adelantar procesos de selección objetiva, previa convocatoria pública para el otorgamiento de permisos para el uso del espectro radioeléctrico. En aquellos casos en los que prime la continuidad del servicio, esta entidad tiene la facultad de otorgar permisos temporales de uso del espectro de manera directa (Chamorro and Barbosa, 2011). Todo este proceso requiere tiempo de análisis de datos y estudios, tiempo que para los interesados, como los operadores de telefonía móvil por ejemplo, es importante y ellos requieren minimizar el riesgo de una respuesta negativa, por esto se ha decidido ejercer acciones sobre uno de los parámetros más importantes: las interferencias.

Para disminuir el riesgo de que se presenten interferencias al crear o modificar un enlace se realiza la integración del sistema de gestión documental que es empleado por los operadores con un sistema de simulación y comparación de enlaces que contiene la información de los enlaces existentes en la red nacional. Teniendo en cuenta que ningún operador tiene acceso directo a esta información y que ingresar la información en el gestor y luego en el simulador toma tiempo además de tornarse tedioso, la integración de los dos sistemas presenta una solución que promete muy buenos resultados y de gran valor para los usuarios.

Aprovechando el desarrollo que han tenido las metodologías ágiles y siendo un tema relativamente novedoso cuyos orígenes datan de mediados de la década de los 90 (Cadavid et al., 2013) y teniendo en cuenta que es apto para un entorno con requerimientos inestables como lo son muchos proyectos de software, se opta por emplear SCRUM como marco de trabajo para desarrollar el proyecto dadas sus características que permiten obtener resultados continuos en tiempos cortos y un incremento en el desempeño del equipo de desarrollo a través del tiempo. Por otra parte se busca presentar algunos resultados importantes sobre la implementación de la metodología de manera que sirva como referencia para trabajos futuros.

1.2. Identificación Del Problema

TES America Andina es una compañía que provee servicios especializados de ingeniería, consultoría y soluciones personalizadas a distintas entidades en la rama de las telecomunicaciones, entre ellas algunos operadores de telefonía móvil. Una de esta soluciones personalizadas es TESGestion®, un OSS (Operations Suport System) que busca la eficiencia operativa de las áreas técnicas, optimización en el manejo de información de las redes de enlaces, mediante procesos estandarizados, organizados y controlados, empleando información veraz, centralizada y oportuna.

Al integrar procesos y datos de las áreas técnicas, TESGestion® contribuye a una mayor organización y administración de la información de la topología de la red de enlaces, lo cual impacta positivamente el interior de la áreas operativas.

El proceso mediante el cual el MinTIC autoriza el uso del espectro radio eléctrico a un emisor en una zona y en unas frecuencias establecidas involucra mucho tiempo desde la radicación de la solicitud hasta la respuesta de este ente, como máximo 3 meses. Uno de los principales requisitos es garantizar que el enlace no interfiera con los enlaces del mismo operador ni con los enlaces de otros operadores: Si dos transmisores con las mismas características que están cubriendo la misma zona transmiten en la misma frecuencia y al mismo tiempo pueden provocar una interferencia perjudicial mutua; ello supone un desperdicio de ERE, puesto que no podría tener ninguna comunicación efec-

tiva. Si el enlace diseñado no cumple con este y otros requisitos será denegado, se deberá rediseñar, buscar frecuencias alternativas y en algunos caso hacer mediciones de ocupación y/o interferencia y el operador tendrá que esperar el siguiente proceso de selección objetiva o solicitar la adjudicación de frecuencias y volver a iniciarse todo el proceso de legalización; todo esto supone costos, recursos y tiempo que son valiosos.

Spectrum-E es un sistema de simulación de enlaces en línea ofrecido por la ANE. La integración de la plataforma de Spectrum-E a TESGestion permite simular el enlace planeado a implementar y realizar el respectivo análisis de interferencia con todos los enlaces (legalizados) de todos los operadores, cabe resaltar que ningún operador conoce la información detallada de los enlaces de otros operadores por lo cual Spectrum-E es una excelente opción, además en caso de haber interferencia presenta frecuencias alternativas para el enlace. Esto si bien es cierto no garantiza que todos los enlaces sean aceptados o no presenten interferencias representa una gran herramienta para depurar el diseño y minimizar el riesgo de que el enlace presente interferencias y sea rechazado.

Por otra parte, a pesar de que el operador puede solicitar acceso al aplicativo mediante la adjudicación de un usuario ante la ANE, se presenta el inconveniente de que no existe un solo documentador (persona encargada de registrar las actividades y actualizar la red en TesGestion) sino que son cientos de ellos y proporcionar acceso a todos estos documentadores a la información del operador en la ANE es arriesgado aún si no lo fuera sería engorroso diligenciar la información en TesGestion y luego hacer lo mismo en Spectrum-E.

Con lo expuesto se presenta la integración de los dos aplicativos como una excelente alternativa para dar solución a estos inconvenientes y mejorar el proceso tanto para el operador como para la entidad.

1.3. Objetivos

Objetivo General:

Integrar la plataforma Spectrum-E de la ANE a la suite TESGestion® para la simulación y validación de enlaces microondas.

Objetivos Específicos:

1. Realizar las actividades de capacitación sobre metodologías ágiles para el desarrollo de software, sistemas operativos y lenguajes de programación suministradas por TES America Andina LTDA.

2. Desarrollar la funcionalidad de simulación de enlaces planeados a instalar.

3. Desarrollar la funcionalidad de simulación de enlaces existentes.

4. Desarrollar la funcionalidad de simulación de múltiples enlaces.

Capítulo 2

MARCO TEÓRICO

2.1. Metodologías de Desarrollo

El primer concepto a tener claro es el de metodología, en cualquier proyecto es importante tener técnicas y herramientas actualizadas que nos permitan llevar a buen término cada actividad dentro del desarrollo de este, pero de nada sirven buenas notaciones y herramientas si no se cuenta con directivas para su aplicación, con base en lo anterior se denominará metodología a aquella disciplina que indicará qué métodos, técnicas y herramientas se deben emplear a lo largo de cada fase del ciclo de vida del proyecto. Una buena metodología debe tener ciertas características para que genere un impacto positivo y sea útil en el desarrollo del proyecto. Tales características se enumeran a continuación:

1. Ha de ser una metodología impersonal.
2. Debe requerir un tiempo de aprendizaje reducido.
3. Apoyos gráficos.
4. Debe emplear algún estándar de documentación.
5. Debe cubrir la mayor parte del ciclo de vida del proyecto, lo ideal es que cubra el total de este.
6. Debe simplificar el trabajo y aumentar la productividad, es decir que no sea una carga exagerada.
7. Que sea madura y evolucione.

Como se dijo anteriormente, al implementar una metodología se busca impactar positivamente el proyecto facilitando su desarrollo. A continuación se enuncian algunas de las numerosas ventajas que aporta la implementación de una metodología.

1. Desde el punto de vista de Gestión:
    - Facilita la tarea de planificación.
    - Facilita la tarea de control y seguimiento de un proyecto.
    - Mejorar relación coste beneficio.
    - Optimizar el uso de recursos disponibles.
    - Facilitar la evaluación de resultados y cumplimiento de objetivos.
    - Facilitar la comunicación efectiva entre usuarios y desarrolladores.
2. Desde el punto de vista de los ingenieros de software:

- Ayudar a la comprensión del problema
- Optimizar el conjunto y cada una de las fases del proceso de desarrollo
- Facilitar el mantenimiento del producto final
- Permitir la reutilización de partes del producto.

3. Desde el punto de vista del cliente o usuario:

   - Garantía de un determinado nivel de calidad en el producto final.
   - Confianza en los plazos de tiempos fijados en la definición del proyecto
   - Implementar el ciclo de vida que más se adecue a las condiciones y características del desarrollo.

2.1.1. Metodologías Tradicionales

Las metodologías tradicionales, como intuitivamente se percibe, fueron las primeras metodologías que surgieron para satisfacer la necesidad de crear productos de alta calidad. La filosofía y el enfoque de estas metodologías se basan en asegurar y detallar desde el principio los requerimientos del producto, en hacer una gestión predictiva para minimizar el incremento de costos por demoras, los costes de mantenimiento, los cambios de requerimientos y asegurar que el proyecto se desarrolle en los tiempos establecidos y que el producto entregado sea el que desde un principio se ideó. Se caracterizan por promover procesos basados en la planificación exhaustiva, esperando al final de cada proceso que el resultado sea determinista y predecible, esto se logra por medio de recolección y estudio de métricas de desarrollo en contextos de trabajo repetibles.

Se pueden citar las siguientes metodologías tradicionales:

1. CMMI (Capability Maturity Model Integration)

   Se apoya bajo los siguientes criterios:

   - La calidad de un producto o un sistema es consecuencia directa de los procesos empleados en su desarrollo.
   - Las organizaciones que desarrollan software presentan un atributo denominado madurez, cuya medida es proporcional a los niveles de capacidad e institucionalización de los procesos que emplean en su trabajo.

2. RUP (Rational Unified Process)

   Se hace un énfasis en cuanto a su adaptación a las condiciones del proyecto (mediante su configuración previa a aplicarse), realizando una configuración adecuada podría considerar como ágil.

3. MSF (Microsoft Solution Framework)

4. Win-Win Spiral Model

5. Inonix

A continuación se detalla la metodología CMMI, una de las más importantes y frecuentemente implementada:

CMMI

Reseña de CMM El *Modelo de Madurez de Capacidades* CMM por sus siglas en inglés, es un modelo cuyo propósito es la evaluación de la calidad de los procesos de una organización a la vez que suministra guías enfocadas a orientar a dicha organización en la mejora de los mismos, todo esto contribuye a desarrollar mejores productos. Aunque se desarrolló inicialmente para los procesos relacionados con el desarrollo e implementación de software en su evolución se ha extendido a las áreas de adquisición y de servicios.

Su origen se remonta hacia el año 1986 cuando el SEI (Software Engineering Institute), centro de investigación y desarrollo patrocinado por el Departamento de Defensa de los Estado Unidos y gestionado por la Universidad Carnegie-Mellon (Chrissis et al., 2011), inició a trabajar en los modelos de madurez de los procesos de desarrollo de software a petición del gobierno estadounidense con el objetivo de atacar los problemas que se generaban con el software que le suministraban otras empresas: fechas de entrega extendidas, incremento del costo, liberaciones con errores, etcétera (Paulk, 1993). Como resultado en el año siguiente publicó el primer documento con la primera definición de estos modelos.

CMM establece un conjunto de prácticas o procesos que son fundamentales y los agrupa en Áreas Clave de Proceso (KPA - Key Process Area). Para cada KPA se define un conjunto de buenas prácticas las cuales deben cumplir con las siguientes características:

- Estar definidas en un procedimiento documentado.
- Estar provistas (la organización) de los medios y formación necesarios.
- Deben ser ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas).
- Se deben medir.
- Se deben verificar.

Además de esto, las KPA's se clasifican dentro de cinco "niveles de madurez", de modo que una organización que implemente en todas su áreas y sean aplicadas por cada uno de sus elementos todas y cada una de las prácticas incluidas en un nivel y sus inferiores, se considera que ésta ha alcanzado ese nivel de madurez. Por madurez se entiende como un atributo de las organizaciones que mide el grado de desarrollo de un proceso, así que en la medida que una organización ejecute procesos homogéneamente implantados, definidos, conocidos y ejecutados por los miembros de la misma la organización será más o menos madura. Los niveles de madurez se emplean para describir un camino evolutivo recomendado para que una organización que busque mejorar sus procesos de desarrollo de productos o servicios (Mary Beth Chrissis, 2012). Los niveles se describen a continuación:

1. *Inicial:* En este nivel no se dispone de un ambiente estable para el desarrollo y mantenimiento del software. No es útil emplear técnicas correctas de ingeniería si los esfuerzos se ven afectados por la falta de planificación; en las organizaciones de este nivel el éxito de los proyectos se basa la mayoría de las veces en el esfuerzo personal, aunque a menudo se producen fracasos y casi siempre retrasos y sobrecostos. No se puede predecir el éxito o fracaso de los proyectos.
2. *Repetible:* Las organizaciones que están en este nivel se caracterizan por implementar unas prácticas institucionalizadas de gestión de proyectos, emplean métricas básicas y un cierto seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente.
3. *Definido:* En este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detalladas y un nivel más avanzado de métricas en los procesos. Se implementan técnicas de revisión por pares (peer reviews).

4. *Gestionado:* Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.

5. *Optimizado:* La organización completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

La Figura 2.1 muestra los componentes de CMM los cuales se describen más adelante:

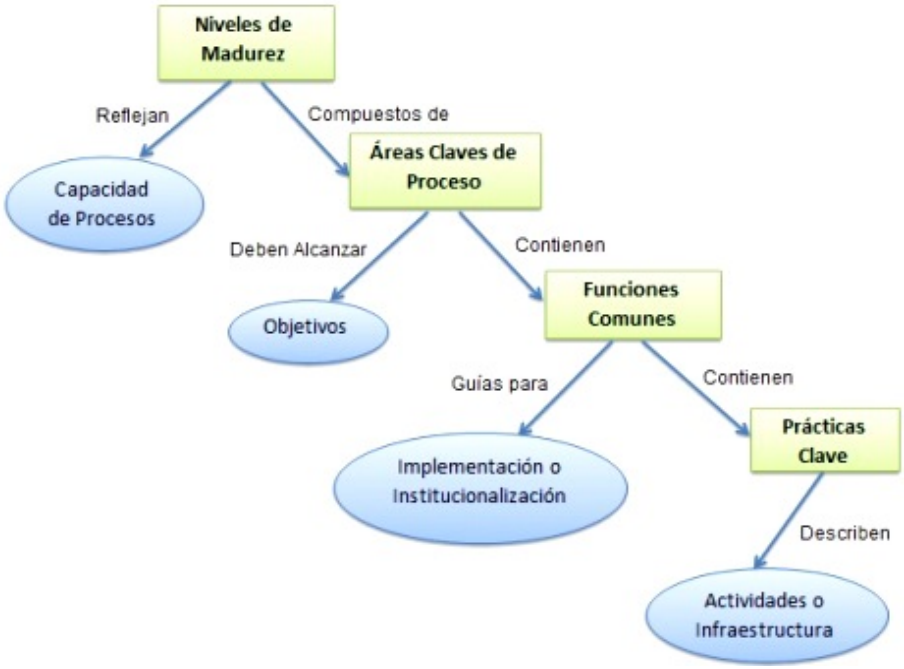


Figura 2.1: Componentes CMM

* Niveles de Madurez: Es una plataforma de evolución definida orientada a obtener un proceso de software maduro. Los cinco niveles conforman la estructura de más alto nivel de CMM.

* Capacidades del Proceso: Describen los resultados esperados que pueden lograrse a través de determinado proceso , además brindan un medio de predicción de las salidas esperadas del próximo proyecto de encarado por la organización.

* Áreas de Proceso Claves (KPA): Cada una identifica un conjunto de actividades relacionadas, que deben desarrollarse conjuntamente por los miembros de la organización involucrados, para alcanzar los objetivos importantes enfocados a establecer las Capacidades del Proceso en determinado nivel de madurez. Se diseñaron de tal forma que una KPA reside en un único nivel de madurez. Por ejemplo: una de las Áreas Clave de Proceso para el Nivel 2 es Planificación de Proyectos de Software.

* Prácticas Claves (KP): cada KPA se describen en Prácticas Claves deben implementarse para satisfacer los objetivos de esa Área de Procesos Clave. Describen la infraestructura y las actividades que apoyan la implementación e institucionalización efectiva del KPA. Ejemplo: una KP para el KPA Planificación de Proyectos de Software es:"El plan de desarrollo del proyecto de software es desarrollado de acuerdo a un procedimiento documentado".

* Funciones Comunes: Las prácticas claves se agrupan en cinco secciones de características comunes: Requisitos a Realizar, Habilidad para Realizar, Actividades Realizadas, Mediciones

y Análisis, y Verificación de la Implementación. Las características comunes son atributos que indican si la implementación y la institucionalización de un Área Clave de Proceso es efectiva y repetible. Las Funciones Comunes de Actividades Realizadas describen las actividades de implementación. Las otras cuatro describen los factores de institucionalización, que hacen que un proceso sea parte de la cultura de la organización.

* Objetivos: Representan el alcance, los límites, y la intención de cada KPA. Por Ejemplo: un objetivo de Planificación de Proyectos de Software es "Las estimaciones del software son documentadas para su uso en la planificación y seguimiento del proyecto de software".

Con el paso del tiempo y la aplicación del modelo en distintas áreas se crearon diversas versiones para cada una: People CMM (P-CMM), Software Acquisition CMM (SA-CMM), Security Systems Engineering CMM (SSE-CMM), trusted CMM (T-CMM), Systems Engineering CMM (SE-CMM), Integrated Product Development CMM (IPD-CMM). Algunas organizaciones implementaban varios modelos de forma paralela lo que derivó en la oportunidad y necesidad de integrarlos para hacer más sencilla su implementación. Así surge CMMI que integra CMM-SW, SE-CMM e IPD-CMM.

CMMI se puede implementar de dos formas: escalonada y continua. El primero se centra en la madurez de la organización presentando los 5 niveles de madurez de CMM, pero se cambian los nombres a los niveles 2 y 4; mientras que el segundo se enfoca en la mejora y control de las capacidades que se clasifican en 6 niveles que van de 0 a 5 y se presenta en orden de la siguiente manera:

- 0- Incompleto: El proceso no se realiza o no se logran los objetivos.
- 1- Ejecutado: El proceso se ejecuta y se logra su objetivo.
- 2- Gestionado: Además de ejecutarse, el proceso se planifica, se revisa y se evalúa para comprobar que cumple los requisitos.
- 3- Definido: Además de ser un proceso "gestionado" se ajusta a la política de procesos que existe en la organización, alineada con las directivas de la empresa.
- 4- Cuantitativamente Gestionado: Además de ser un proceso definido se controla utilizando técnicas cuantitativas.
- 5- Optimizado: Además de ser un proceso cuantitativamente gestionado, se revisa sistemáticamente

2.1.2. Metodologías Ágiles

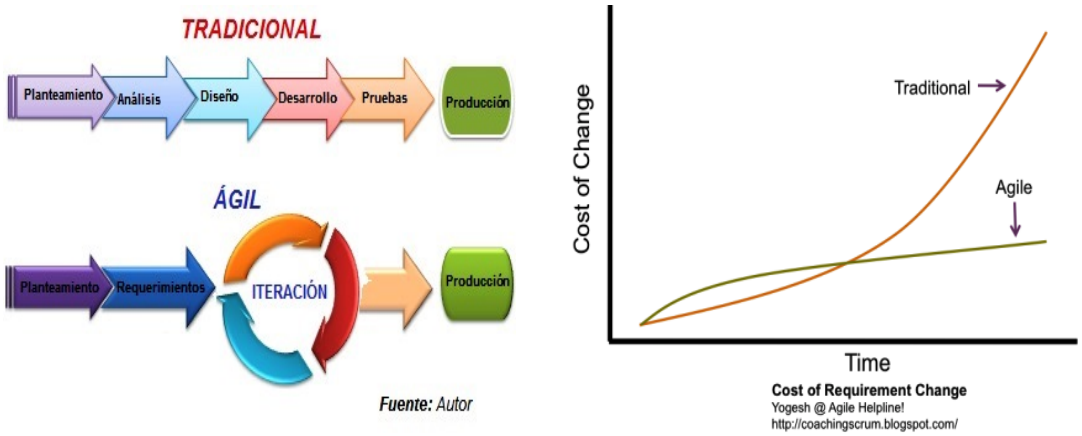
El ambiente de trabajo de la gran mayoría de las empresas de desarrollo de software, y en general dedicadas al desarrollo del conocimiento, presenta grandes diferencias respecto al que originó la gestión de proyectos predictiva (tradicional), debido a la rápida y constante evolución de los mercados se requieren estrategias enfocadas a la entrega temprana y continua de productos con resultados tangibles, y a la respuesta ágil y flexible. Ahora se construye el producto mientras se modifican y aparecen nuevos requisitos, pues el cliente parte de una visión medianamente clara de su producto, sin embargo, el grado de innovación que requiere, su evolución y adaptación a los cambios generados por la dinámica del sector de su negocio, le impiden predecir con detalle cómo será el resultado final. Podría llegar a afirmarse que ya no se generan "productos finales", en vez de estos se entregan productos en una dinámica de continua evolución y mejora.

Dado que las metodologías tradicionales son muy rígidas y "pesadas", pues se dedican tiempo y recursos en el desarrollo de una planificación exhaustiva y en las que se busca minimizar los cambios debido al costo eleva que implica solucionar los errores detectados o introducir modificaciones cuanto más avance el proyecto (Boehm et al., 1981), se hacen inapropiadas en un mercado dinámico. Las metodologías ágiles nacen de la necesidad de adaptarse a un entorno variante e inestable, que inherentemente implica el cambio y la continua evolución rápida y en el que se requieren entregas tempranas que satisfagan las necesidades cambiantes de los usuarios y lógicamente del cliente. Es así que las metodologías ágiles surgen para dar soluciones a los principales inconvenientes que presentan las metodologías tradicionales:

- Se invierten cantidades considerables de tiempo y esfuerzo en la fase inicial (planificación), cuyo resultado debe: ser una "fotografía exacta" de lo que desea el cliente (quien generalmente no tiene claro lo que desea), minimizar al máximo los cambios de requerimientos por el costo que involucra; además el cliente es quien debe acarrear los costos de un nuevo requerimiento y en esta fase no ve más que papeles.
- Se requiere de una documentación excesiva, que no siempre es útil y que inherentemente genera un gasto de tiempo y esfuerzo que no genera valor al producto y el desarrollo se hace más lento.
- Se tiende a acumular los riesgos y dificultades a medida que se avanza en el desarrollo del producto, lo cual puede ocasionar retrasos en la entrega.
- Es una metodología burócrata donde no hay "contacto" cercano y continuo con el cliente.
- Las dificultades o nuevos requerimientos aparecen en las fases finales que es donde el producto es testeable y en donde es "visible" para el cliente, generando costos elevados y retrasos en los tiempo de entrega.

En la Tabla2.1 se presenta un cuadro comparativo entre las metodologías ágiles y las tradicionales. La Figura.2.2(a) compara los ciclos de vida de ambas metodologías mientras que la Figura.2.2(b) presenta las curvas de comportamiento de los riesgos y costos de cambios a lo largo del ciclo de vida de desarrollo en cada metodología.

De la Figura.2.2(b) se puede concluir que el costo de cambios en los requerimientos empleando metodologías tradicionales se incrementa conforme el proyecto avanza y crece de forma casi exponencial siendo excesivamente elevado finalizando el proyecto, este comportamiento se genera a causa de la necesidad de replantear muchas cosas, se pierde una cantidad importante de los recursos invertidos en las fases anteriores y se debe volver a hacer el análisis y la planificación. Mientras que con las metodologías ágiles el costo de cambio no es tan elevado y es casi constante durante el ciclo de vida del proyecto debido a la adaptación y preparación para el cambio al no dedicar demasiados recursos en la planificación exhaustiva.



(a) Ciclos de vida ágil y tradicional

(b) Costo de cambios en ambas metodologías

Figura 2.2: Ciclo de vida y costos de cambio

Tabla 2.1: Comparación de metodologías.

Ágiles	Tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a cambios
Reglas impuestas internamente por el equipo	Reglas impuestas externamente
Proceso poco controlado con mínimas normas	Proceso muy controlado, numerosas normas
Contrato flexible, indeterminado o incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	El cliente es ajeno al desarrollo, solo interactúa mediante reuniones

El pilar de las metodologías ágiles son: los valores y los principios del Manifiesto Ágil que se firmó entre el 11 y el 13 de febrero de 2001 durante una reunión desarrollada en el resort Lodge en Snowbird en Utha(USA)(Uribe and Ayala, 2007). En dicha reunión 17 críticos, empresarios y desarrolladores de software convocados por Kent Beck para debatir y entender las ideas y enfoques de cada uno, como resultado nació el *Manifiesto Ágil* cuyos valores se citan a continuación:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> **Individuals and interactions over** processes and tools
> **Working software over** comprehensive documentation
> **Customer collaboration over** contract negotiation
> **Responding to change over** following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more."

En el manifiesto se indica que se valoran más las personas que los procesos sin indicar que éstos no sean necesarios, se valora más la adaptación y la agilidad que la documentación y la planificación rigurosa sin dejarlas de lado por completo. Además de los valores se acordaron 12 principios que se listan a continuación

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4. *Business people and developers must work together daily throughout the project.*
5. *Build projects around motivated individuals.Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*
8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous attention to technical excellence and good design enhances agility.*
10. *Simplicity–the art of maximizing the amount of work not done–is essential.*
11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Al igual que con las metodologías tradicionales, existen diversas metodologías ágiles, a continuación se describen brevemente XP y Crystal para luego profundizar en SCRUM.

- XP: Desarrollada por Kent Beck con el fin de guiar equipos de desarrollo de software pequeños o medianos, entre dos y diez desarrolladores, en ambientes de desarrollo con requerimientos imprecisos o cambiantes (Beck, 2000). XP se basa en cinco valores: Simplicidad, Comunicación, Retroalimentación, Respeto y Coraje (Jeffries, 2011). Estos valores, a su vez, son la base para la definición de sus principios. De ellos, los fundamentales son: la retroalimentación rápida, asumir simplicidad, el cambio incremental, la aceptación del cambio y el trabajo de calidad (Beck, 2000).

  A partir de estos valores y principios se desprenden las prácticas de esta metodología cuyo enfoque es darle solución a las actividades básicas de un proceso de desarrollo, esto es: escribir código, realizar pruebas, escuchar (planear) y diseñar. Las prácticas de XP incluyen: planning game, pequeñas entregas, diseño simple, programación en pareja, pruebas, refactoring, integración continua, propiedad común del código, paso sostenible, cliente en sitio, metáfora y estándares de código (Cadavid et al., 2013).

- Crystal: Es un conjunto de metodologías basadas en los conceptos de RUP que se compone por Crystal Clear, Crystal Yellow, Crystal Orange y Crystal Red; el número de personas implicadas releja el nivel de opacidad del color en el nombre, un mayor tamaño del proyecto y, por lo tanto, la necesidad de mayor control en el proceso (Abrahamsson et al., 2003). Su filosofía concibe el desarrollo como un juego cooperativo de invención y comunicación con el objetivo principal de entregar software útil, que funcione, y en adición preparar el próximo juego.

  El centro de los valores compartidos por los miembros de la familia Crystal están en las personas y en la comunicación. Sus principios indican que: el equipo puede reducir trabajo

intermedio en la medida que produce código con mayor frecuencia y utiliza mejores canales de comunicación entre las personas (Cadavid et al., 2013). Crystal define dos reglas principales, la primera indica que los ciclos donde se crean los incrementos no deben exceder cuatro meses; la segunda, que es necesario realizar un taller de reflexión después de cada entrega para afinar la metodología.

SCRUM

Es un marco de trabajo que se basa en el empirismo y la experticia, su origen se dio gracias al estudio desarrollado por los profesores Horotaka Takeuchi e Ikujiru Nonaka a inicios de 1980 quienes analizaron el proceso de desarrollo de productos tecnológicos de las principales industrias de sector: Canon, Honda, Nec, Hewlett-Packard, entre otras, y cuyos resultados se publicaron en 1986 (Takeuchi and Nonaka, 1986). Como resultado de este análisis Takeuchi y Nonaka lograron identificar la similitud entre la forma de trabajo de los equipos de estas organizaciones y el avance en formación SCRUM propia del Rugby, de ahí que ese nuevo framework adopte este nombre.

SCRUM es apropiado en entornos de desarrollo con requerimientos inestables, no muy bien definidos y en los que se requieres rapidez y flexibilidad. Entre los años 1993 y 1995 se conforma el primer equipo SCRUM y se formaliza este framework gracias a Jeff Sutherland y Ken Schwaber (Deemer et al., 2010).

I. El Modelo:

La Figura2.3 muestra gráficamente el marco de Scrum con sus elementos: Roles, Artefactos y Eventos. El modelo es iterativo incremental y se desarrolla en ciclos de trabajo denominados *Sprints*, los cuales son limitados en tiempo (*Time boxing*), tienen una duración fija y se realizan uno tras otro. Los tres pilares (Schwaber et al., 2013) sobre los que se soporta la gestión y el control del proceso son:

- ***Transparencia:*** Todos aquellos comprometidos con el resultado del desarrollo deben poder visualizar los elementos importantes de proceso. Estos elementos deben estar bajo un estándar común, de tal modo que quienes lo visualicen entiendan la misma idea o concepto.
- ***Inspección:*** Se deben inspeccionar los artefactos y el progreso frecuentemente a fin de detectar variaciones, esta inspección no debe interferir con el trabajo por esto sería ideal que las realizaran inspectores expertos.
- ***Adaptación:*** Si como resultado de la inspección se detectan elementos que se desvían de la aceptación y que el producto no estará en concordancia con la idea inicial y/o que se corre el riesgo de ser rechazado, se deben hacer los ajustes necesarios para minimizar mayores desviaciones. Para ello se emplean las reuniones de inspección y adaptación: Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review, Sprint Retrospective Meeting.

Para desarrollar el producto se parte de una visión general de lo que el cliente desea obtener y a partir de ella se especifican y detallan los items más prioritarios, los cuales se incluirán en el Sprint y a los cuales el equipo se *compromete* a entregar al finalizar el Sprint. Durante el transcurso del sprint se desarrollan reuniones diarias de corta duración con el objeto de examinar el trabajo desarrollado el día anterior, el trabajo para el día presente y los posible inconvenientes que se tengan y actualizar las gráficas empleadas para visualizar el "estado" del sprint.

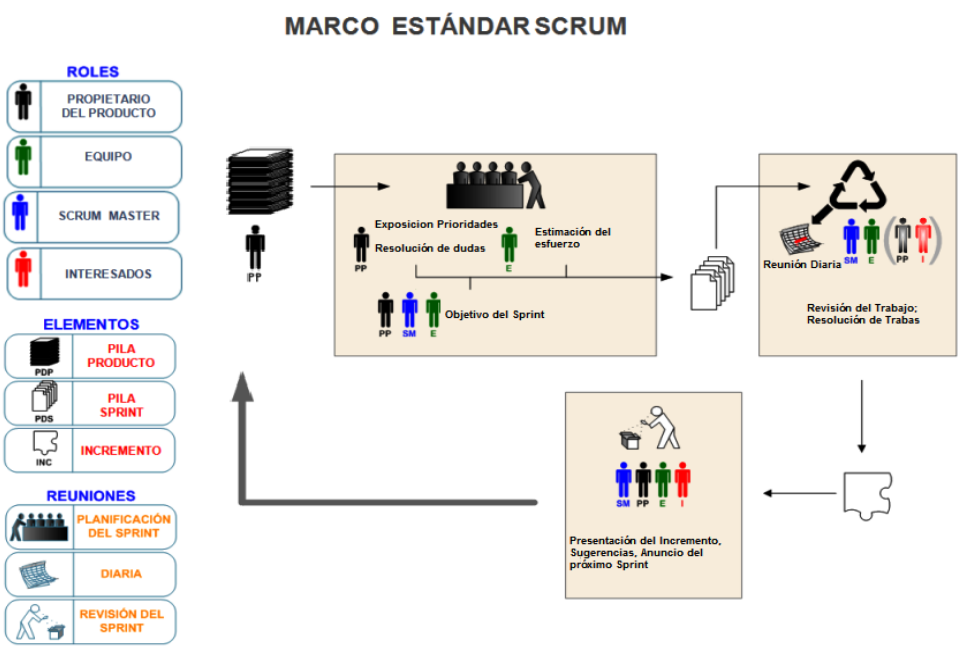


Figura 2.3: Marco de SCRUM

La gestión de la evolución del proyecto se realiza en base al trabajo pendiente por realizar (Palacio, 2014), para lo cual se llevan a cabo reuniones de revisión al finalizar cada sprint en las que se hace la presentación de la nueva funcionalidad (incremento) ante el cliente y los interesados de quienes se recibe la retroalimentación que se tiene en cuenta en los próximos sprints.

Tabla 2.2: Elementos de SCRUM

Elementos de Scrum	
Roles	Product Owner Scrum Team Scrum Master stakeholders
Artefactos	Product Backlog Sprint Backlog User Stories Sprint Tasks Sprint Burndown Chart Sprint
Eventos	Sprint Planning Meeting Backlog Refinement Meeting Daily Scrum Meeting Sprint Review Meeting Sprint Restospective Meeting

La Tabla.2.2 presenta los elementos agrupados en roles, artefactos y eventos que serán descritos con detalle a continuación:

II. Roles:

Los roles se clasifican entre comprometidos e involucrados de acuerdo a la fábula del cerdo y la gallina (Keith, 2010). Los comprometidos (cerdos) son: el Scrum Team, el Scrum Master y el Product Owner aunque en ciertos aspectos puede ser considerado como gallina (involucrado), entre estos roles no existe ningún nivel de jerarquía; mientras que los involucrados (gallinas) son los

Stakeholders.

A. Scrum Team (ST): El equipo scrum está compuesto por los individuos que desarrollan el producto trabajando en conjunto. Es importante para su eficiencia que cada miembro siga un objetivo en común, se adhiera a las misma normas y que muestre respeto a los demás. Los equipos desde su inicio pasan por diferentes etapas que se describen el el modelo Tuckman: Formación, Tormenta, Normalización y Ejecución (Tuckman, 1965).Los Scrum Teams siempre tienen las siguientes características (Kniberg, 2007):

- Los miembros del equipo comparten las mismas normas y reglas.
- El equipo de Scrum en su conjunto es responsable de la entrega.
- El Equipo Scrum está facultado.
- Está trabajando tan autónomo, como le sea posible.
- El Equipo Scrum es auto-organizado.
- Las habilidades dentro del equipo Scrum se equilibran.
- Un equipo Scrum es pequeño y no tiene sub-equipos.
- Las personas dentro del equipo Scrum trabajan a tiempo completo en el equipo.
- Las personas están localizadas en un mismo sitio.

Reglas y Normas : El entorno define algunas de las normas que cada miembro del Scrum Team debe seguir, algunas de estas se desarrollan durante la fase de Normalización, y son muy importantes pues de no existir los miembros del equipo tendrían que perder constantemente valioso tiempo para cambiar entre los diferentes sistemas de valores y conjuntos de reglas. Ejemplos de tales normas y reglas son: hora y lugar de la reunión diaria de Scrum, la definición de ***Hecho*** **(DoD)** utilizada para decidir si el trabajo está terminado o no, directrices de codificación, herramientas a utilizar, etc.

Responsabilidad: El Scrum Team en conjunto es responsable de la entrega comprometida en el tiempo y con la calidad definida. Un buen resultado o un fracaso nunca se atribuye a un solo miembro, siempre el resultado es del Scrum Team.

Empoderamiento y Organización Propia: El Scrum Team tiene que estar facultado para definir lo que se compromete a entregar al final del sprint, cómo los resultados esperados han sido desglosarse en tareas, qué tarea llevará a cabo y en qué orden se realizan. Sólo si el Scrum Team está facultado para decidir estas cosas va a trabajar con la más alta motivación y rendimiento posible.

Conjunto equilibrado de habilidad: Las personas dentro del Scrum Team tienen habilidades y enfoques especializados. Sin embargo, para lograr el mejor rendimiento posible sería óptimo tener un conjunto equilibrado de habilidades, así Scrum Team será capaz enfrentarse a retos siempre cambiantes y actuar de manera autónoma. Un Scrum Team debe ser multidisciplinario (desarrolladores, probador, arquitectos, etc), y cada miembro del equipo debe aprender un poco de la especialización de cada uno, por ejemplo, un desarrollador también debería poder realizar o escribir pruebas. Una norma muy importante es que dentro de Scrum no se clasifica por habilidades sino que todos comparten el mismo título “Scrum Team Member”.

Tamaño: Los Scrum Teams son pequeños. El tamaño ideal es de 7 +/- 2 personas. Si hay más gente la sobrecarga de la comunicación se hace muy grande y el equipo debe dividirse en varios Scrum Teams. Estos Scrum Teams deben coordinarse y comunicarse entre sí, pero deben trabajar de forma independiente.

Locación: Para minimizar la sobrecarga de comunicación innecesaria cada Scrum Team debe coubicarse. Si el trabajo tiene que ser repartidos en varias ubicaciones, deben crearse Scrum Teams independientes.

Responsabilidades del Equipo Scrum: El Scrum Team y cada uno de los miembros tienen ciertas responsabilidades que deben cumplirse: Tienen que desglosar los requisitos (historias de usuario) en elementos más pequeños (tareas), estimarlas y distribuirlas,en otras palabras, esto significa que tienen que crear el Sprint Backlog; además deben participar en la reunión diaria (Scrum Daily Meeting), actualizar el estado y los esfuerzos que quedan para sus tareas para permitir la creación del Burndown Chart y desarrollar el incremento.

B. Scrum Master (SM): Es el entrenador del equipo, ayuda a que se consiga el máximo rendimiento del Scrum Team. Es el responsable de asegurar que Scrum es entendido y adoptado, su principal función es asegurar que el Scrum Team trabaja ajustándose a la teoría, prácticas y reglas de Scrum.

El Scrum Master debe ser un líder que está al servicio del Scrum Team, ayuda a las personas externas al equipo Scrum a entender qué interacciones pueden ser de ayuda y cuáles no, busca que se modifiquen estas interacciones de tal forma que el Scrum Team pueda concentrar sus esfuerzo en el desarrollo del producto. Además es el encargado de hacer la logística y liderar las diferentes reuniones (eventos) de Scrum. El SM se relaciona con los demás roles de scrum y con la organización como se muestra en la Tabla 2.3(Schwaber et al., 2013).

Tabla 2.3: Relación del SM con los demás roles

Relación del SM con	
Product Owner	Le ayuda a encontrar técnicas para gestionar la lista de producto de manera efectiva Busca que el PO conozca como ordenar la lista de producto para maximizar el valor Facilitar los eventos de Scrum según se requiera o necesite. Entender y practicar la agilidad
Scrum team	Guia al equipo de desarrollo en ser auto-organizado y multifuncional Eliminar impedimentos para el progreso del Equipo de desarrollo Facilitar los eventos Scrum Guiar al equipo de desarrollo en el entorno de organizaciones en las que scrum aún no ha sido adoptado o entendido Ayuda a al equipo a crear productos de alto valor.
Organización	Liderar y guiar a la organización en la adopción de scrum Planificar las implementaciones de scrum en la organización Motivar cambios que incrementen la productividad del equipo Trabajar con otros scrum masters para incrementar la efectividad de la aplicación de scrum en la organización

C. Product Owner (PO): El dueño del producto es un individuo miembro de la organización que requiere el software y es el responsable de conducir el proyecto desde la perspectiva del negocio. Es quien debe comunicar una visión clara del producto y definir sus características principales, priorizando los requerimientos del cliente para que el proceso de desarrollo se centre en aquellos requerimientos necesarios para la organización. En todo momento el PO debe contribuir con el equipo para remover todas las dudas que surgen acerca de los requerimientos. Por esta razón, es necesario que el lugar de trabajo del PO esté en el mismo espacio físico que el equipo.

Las responsabilidades del PO (que puede ser interno o externo a la organización) son:

* Gestionar el Product Backlog: Crear las historias de ususario, priorizar el PB y reorganizarla (antes de cada sprint).

* Ser el representante de todas las personas interesadas (stakeholders) en los resultados del proyecto (internas o externas a la organización, promotores del proyecto y usuarios o consumidores finales del producto) y actuar como interlocutor único ante el equipo, con autoridad para tomar decisiones

* Definir los objetivos del producto o proyecto.

* Dirigir los resultados del proyecto y maximizar su ROI:

  - Es el propietario de la planificación del proyecto y por tanto se deben respetar sus decisiones: crea y mantiene la lista priorizada con los requisitos necesarios para cubrir los objetivos del producto o proyecto, conoce el valor que aportará cada requisito y calcula el ROI a partir del coste de cada requisito que le proporciona el equipo.
  - Reparte los objetivos/requisitos en iteraciones y establece un calendario de entregas.
  - Antes de iniciar cada iteración replanifica el proyecto en función de los requisitos que aportan más valor en ese momento, de los requisitos completados en la iteración anterior y del contexto del proyecto en ese momento (demandas del mercado, movimientos de la competencia, etc.).

* Colaborar con el equipo para planificar, revisar y dar detalle a los objetivos de cada iteración.

* Participar en la reunión de planificación de iteración, proponiendo los requisitos más prioritarios a desarrollar, respondiendo a las dudas del equipo y detallando los requisitos que el equipo se comprometer a hacer.

* Estar disponible durante el curso de la iteración para responder a las preguntas que puedan aparecer.

* Participar en la reunión de demostración de la iteración, revisando los requisitos completados.

D. Stakeholders SH: Son las personas interesadas en el producto, como: clientes, usuarios, patrocinadores, etc. Su rol no es obligatorio pero tiene una gran importancia ya que aportan al Product Owner muchas ideas a partir de las cuales él crea su visión del producto. Se consideran "gallinas" pues están involucrados en el desarrollo mas no comprometidos. No es conveniente que interactúen directamente con el Scrum Master ni mucho menos con el Scrum Team ya que pueden generar "ruido" y distracciones.

III. Artefactos:

Los artefactos de Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, que es necesaria para asegurar que todos tengan el mismo entendimiento del artefacto.

A. Product Backlog (PB): La pila de producto es la lista de requerimientos que le dan valor de negocio al producto, contiene las funcionalidades y/o requisitos que el producto irá adquiriendo con cada iteración. Es gestionada por el Product Owner quien debe crearla y ordenarla por prioridad. Las características principales del PB son:

- Contendrá los requerimientos del producto expresados en historias de usuario(*User Stories*).
- Se gestiona entre cada sprint exclusivamente por parte del PO y nunca durante el transcurso de un Sprint.
- Visible a todos, incluidos los stakeholders.
- Evoluciona constantemente, es un artefacto vivo.

Antes de iniciar cualquier Sprint es necesario definir los objetivos y tener la lista de requerimientos definida, no en extenso detalle pero sí con la información suficiente para que el equipo pueda desarrollarla. Los elementos de las listas se llaman historias de usuario (*User Stories*) y se expresan en terminología del cliente (Kniberg, 2007).

B. User Stories (US): Las historias de usuario son los items del Product Backlog y contienen la información de cada funcionalidad a desarrollar. Estas historias tiene los siguiente campos como mínimo.

- ***ID:*** Un identificador único para facilitar el seguimiento.
- ***Nombre:*** Título descriptivo de la historia.
- ***Descripción:*** Una descripción corta, no más de 15 palabras, lo suficientemente clara para que el PO comprenda de qué se trata y que se distinga de las demás. Ejemplo: *"Como usuario quiero consultar el reporte de notas en una página web"*.
- ***Prioridad:*** La importancia que le da el PO a la historia de acuerdo al valor de negocio que representa para él. Entre más alto el valor, más importante es la historia.
- ***Estimación:*** La valoración que el Scrum Team da en cuanto el ***esfuerzo*** que se requiere para desarrollar la histora. Sus unidades son puntos de esfuerzo, la estimación absoluta no es importante, es más importante la relatividad de la estimación: que una historia de 2 puntos dure la mitad que una de 4(Kniberg, 2007).
- ***Cómo Probarlo:*** Una descripción a alto nivel de cómo se demostrará la historia desarrollada. Es una breve descripción de un test.
- ***Notas:*** Se agrega información adicional como fuentes de información, clarificación, etc. Normalmente muy breve

La Figura.2.4 muestra un ejemplo del formato de las User Stories

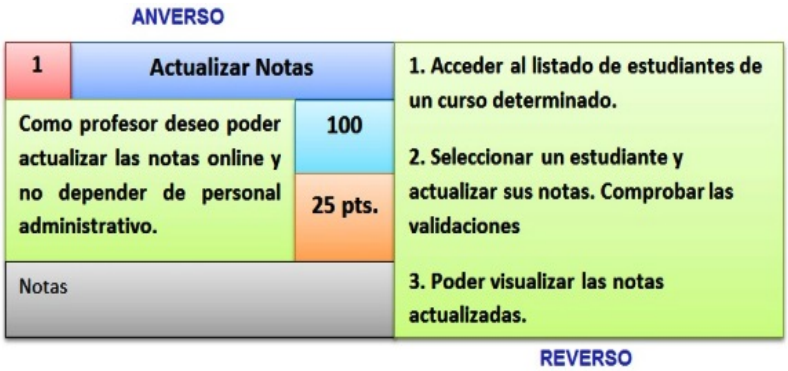


Figura 2.4: Formato historias de usuario

C. Sprint Backlog (SB): La Pila de Sprint está compuesta por las User Stories que el Product Owner y el Scrum Team negocian durante la Sprint Planning Meeting y este último se compromete a entregar al finalizar el sprint. Se expresa en una lista de tareas (Tareas del Sprint (TS)). Su alcance idealmente es fijo y es visible a todo el equipo. Es gestionado por el Scrum Team durante la Daily Scrum Meeting.

D. Sprint Tasks (TS): Son la descomposición de las historias de usuario en elementos más pequeños que especifican cómo alcanzar el qué de las User Stories y son identificadas por el Scrum Team durante la Sprint Planning Meeting. Son las actividades deben realizarse para desarrollar la historia de usuario a la cual pertenecen. Se expresan en lenguaje técnico del equipo y cada persona se *compromete* a trabajar sobre la tarea que el voluntariamente escoja y/o que sea asignada.

E. Sprint Burndown Chart (SBC): El gráfico de trabajo realizado es una herramienta que ayuda a visualizar y hacer seguimiento al trabajo realizado y el trabajo pendiente por hacer. Su gestión y actualización se realiza en la DSM por parte del ST. Es importante saber interpretar la gráfica y actuar ante los resultados y variaciones que esta presente. La Figura.2.5 muestra un ejemplo del Sprint Burndown Chart

Figura 2.5: Sprint Burndown Chart

IV. Los Eventos:

A. Product Backlog Refinement Meeting (PBRM): La mayoría de las US deben refinarse, ya que son grandes y poco comprendidas. En la reunión de refinamiento del PB, el equipo estima la cantidad de esfuerzo que se debe invertir para completar los ítems del PB proporcionando información técnica para ayudar al PO a priorizarlos. Las historias que requieren esfuerzos demasiado grandes se dividen y se clarifican, teniendo en cuenta temas tanto de negocio como técnicos. El PO ayuda al equipo resolviendo dudas e inquietudes con el fin de que todos los miembros del ST tengan la misma idea de cada historia de usuario. Las estimaciones de esfuerzo se realizan por diversos métodos, el más popular y más empleado es el *Planning Poker*.

A veces, un subconjunto del equipo, junto con el PO y los SH, descomponen y dividen los ítems del PB antes de involucrar a todo el equipo en la estimación. Un SM experimentado puede ayudar al equipo a identificar finas rebanadas verticales de trabajo que tienen valor comercial, mientras promueve una definición rigurosa de "terminado" que incluye las pruebas adecuadas y refactorización.

B. Sprint Planning Meeting (SPM): Antes de empezar un Sprint, el PO, el SM y el ST tienen una Reunión de Planificación del Sprint SPM que se divide en dos partes (Deemer et al., 2010). En la primera parte se definen los objetivos del Sprint, la duración de este y se negocian qué ítems del PB intentarán convertir en producto funcional durante el Sprint, en resumen se responde ¿ qué puede ser terminado en este Sprint ?. El PO es el responsable de declarar cuáles son los ítems más

importantes para el negocio. El equipo es responsable de seleccionar la cantidad de trabajo que se comprometen a realizar sin acumular deuda técnica, lo cual es una práctica clave en Scrum: decidir qué cantidad de trabajo se compromete a realizar basándose en su experiencia y conocimiento tanto técnico como funcional del equipo, en vez de que sea un trabajo asignado por alguien más; el SM es el moderador y el encargado de hacer que la reunión fluya y no se estanque pues es una reunión limitada en tiempo (*timeboxing*). El equipo "toma" el trabajo desde el PB hacia el SB. En la segunda parte se responde a la pregunta ¿ Cómo se conseguirá hacer el trabajo necesario para entregar el incremento?; para ello se dividen las US en las Sprint Task en base a las necesidades más detalladas del PO y de los conocimientos técnicos del ST.

C. Daily Scrum Meeting (DSM): La reunión diaria de Scrum tiene un bloque de tiempo de 15 minutos en la cual el Scrum Team sincroniza sus actividades y crea un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde el último Scrum Diario y haciendo una proyección acerca del trabajo que podría completarse antes del siguiente.

El Scrum Diario se realiza todos los días en el mismo lugar y a la misma hora, con la misma duración. Durante la reunión, cada miembro del Scrum Team responde a las siguientes preguntas planteadas por el SM:

- ¿ Qué hizo ayer que ayudó al ST a lograr el Objetivo del Sprint?
- ¿ Qué hará hoy para ayudar al ST a lograr el Objetivo del Sprint?
- ¿ Ve algún impedimento que evite que el ST o yo logremos el Objetivo del Sprint?

Otro propósito que tiene el DSM es evaluar el progreso hacia el Objetivo del Sprint y la tendencia que tiene este progreso de cara a la culminación del trabajo pendiente en el Sprint Backlog. Además se debe actualizar el SBC.

D. Sprint Review Meeting (SR): En Scrum la inspección y adaptación son pilares para ver y saber lo que está pasando y luego evolucionar en base a la retroalimentación, en la repetición de ciclos. La Revisión del Sprint es una actividad para este propósito, es un tiempo para que el PO conozca lo que qué está sucediendo con el producto y con el equipo (es decir, una revisión del Sprint); y para que el equipo se entere lo que sucede con el PO y el mercado. Así pues, un elemento vital es una profunda conversación entre el ST y el PO para conocer la situación y obtener retroalimentación. La revisión incluye una demostración de lo que el equipo construido durante el Sprint, pero no es el objetivo principal.

E. Sprint Retrospective Meeting (SRM): La reunión de retrospectiva del Sprint es muy importante pues es donde se analiza lo que ocurrió durante el Sprint. Es la reunión siguiente a la del Sprint Review y antes de la siguiente planeación de Sprint, durante la reunión se discuten los siguientes puntos:

- Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas.
- Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras.
- Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo.

Esta reunión es fundamental para ayudar al equipo a mejorar, pues del examen interno se detectan los aspectos a mejorar y se crean compromisos que adquiere cada miembro para con el resto del equipo y la organización. Esta reunión es manejada por el Scrum Master quien debe buscar que cada miembro comprenda su importancia y además sobrepasar las barreras psicológicas mencionadas por (James, 2010).

2.2. Conceptos de Programación

En esta sección se expondrán los conceptos y fundamentos de programación empleados para desarrollar el proyecto, como lo son la programación orientada a objetos (Object Oriented Programming (OOP)) que es el paradigma empleado, las bases de datos y programación multihilos.

2.2.1. Programación Orientada a Objetos

La programación orientada a objetos nace como la mayoría de las tecnologías a partir de una necesidad. En este caso surge como solución a las limitaciones de su predecesora la programación estructural y a la *crisis del software*, aunque no ha sido la solución definitiva a esta última. En la programación estructural se diseñan y manejan los programas como grupos de elementos funcionales de código reutilizable llamados procedimientos que definen parámetros de entrada y salida y que además se invocan (llaman) unos a otros modificando datos almacenados en determinadas posiciones de memoria. (Velarde et al., 2006)

El inconveniente de la programación estructural radica en la dificultad y limitación en la reutilización de código, pues únicamente los procedimientos se pueden reutilizar y es muy complicado haceros genéricos y flexibles al ser estructuras de datos abstractas y que generalmente requieren distintas implementaciones. Esto no quiere decir que no sea útil, de hecho en aplicaciones sencillas es más fácil emplear la programación estructural.(Nakov, 2013)

Otro problema que se tiene al emplear la programación estructural es que debe existir una fuerte comunicación, entendimiento y sincronización entre los desarrolladores, si se tiene en cuenta el hecho que cada uno de ellos estará manipulando funciones separadas que pueden trabajar sobre tipos de datos mutuamente compartidos, luego los cambios de un desarrollador deben reflejarse en el trabajo del resto del equipo. Además no modela muy bien el mundo real al separar datos de funciones.

Desde la década de 1960 se venía trabajando en las ideas y principios bases de la OOP con Simula, luego en la década de los 70's aparece Small talk, no fue hasta la década de los 80's que este paradigma iba tomando fuerza entre los programadores debido a la aparición de C, C++ y se extendió a BASIC y Pascal. Seguidamente en Eiffel (ahora JAVA) y a partir del año 98 se desarrolla la arquitectura de objetos distribuidos (Kendal, 2014). La Figura.2.7 muestra gráficamente la comparación de la programación estructurada y la Programación Orientada a Objetos.

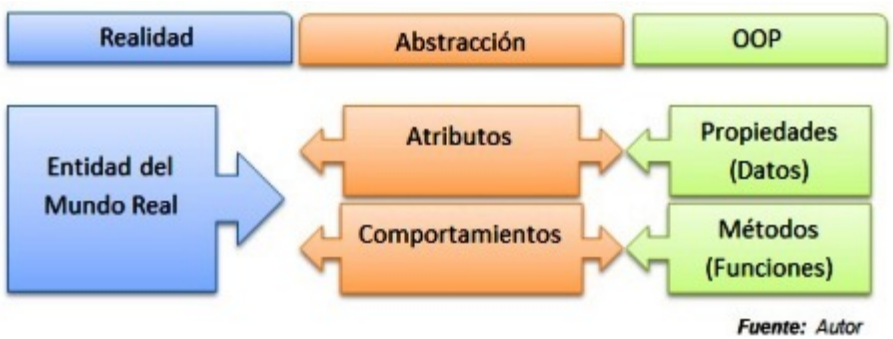
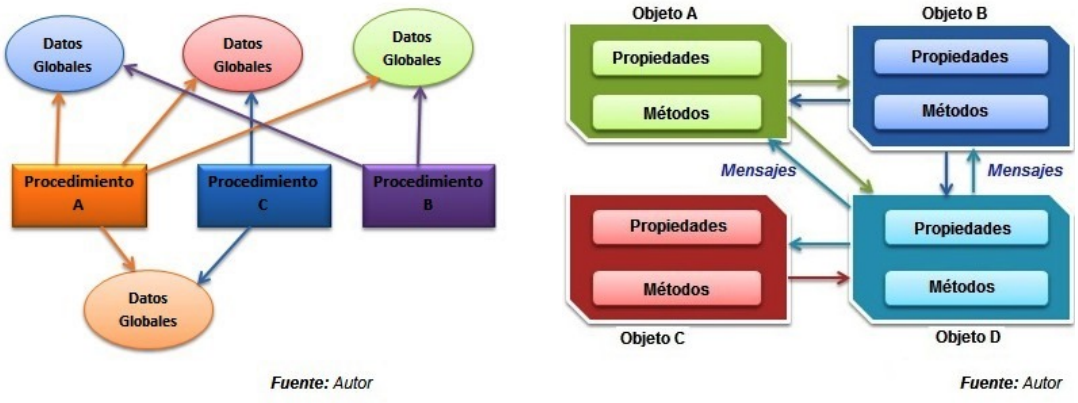


Figura 2.6: Modelamiento del mundo real a OOP

La idea en la OOP es pasar de ajustar el mundo al lenguaje, a ajustar el lenguaje a lo que es el mundo real, a modelarlo. Se deja de ver todo como cajas negras (procedimientos) a las que entran y salen datos manejados "aisladamente" para adoptar un modelo de entidades (objetos) que poseen atributos y comportamientos, de manera tal que ya no se manejan datos y funciones por separado,

sino que se integran y son parte de cada objeto *(encapsulamiento)* y por lo tanto cada objeto es responsable de sus datos y funciones. Los objetos interactúan entre sí mediante mensajes para producir un resultado, lo cual modela mejor el comportamiento de las cosas en el mundo real y más fácil modelar los problemas al emplear un paradigma más intuitivo. La Figura.2.6 ilustra el mapeo de las entidades del mundo real a la OOP.



(a) Programación Estructurada (b) OOP

Figura 2.7: OOP vs Estructurada

Conceptos de POO

A continuación se presenta una descripción de los principales elementos y características más importantes que se manejan en OOP

Objeto: Conceptualmente describe una entidad del mundo real que posee atributos y comportamientos; los primeros son sus características, por ejemplo, en un coche su marca, modelo, velocidad etc; mientras que los segundos son las acciones que ejecutan ante algún estímulo, como un coche aumenta su velocidad al pisar el pedal del acelerador. Además de estos componentes, un objeto tiene una identidad que lo diferencia de los demás objetos. Formalmente se define un objeto es una instancia de una clase que tiene un ciclo de vida dentro del programa.

Clase: Una clase es una plantilla o modelo que se usa para definir los atributos y métodos que tendría un objeto. De manera más formal se define como la abstracción de las características y comportamientos de alguna entidad que se está modelando en el programa.

Abstración: En la OOP, no es esencial reproducir todas las características ni comportamientos de una entidad en el programa, pues la cantidad de información sería inmanejable y la gran mayoría sería información superflua (Clark and Sanders, 2011). Es la abstracción la propiedad por la cual se tiene en cuenta solo aquellas características más relevantes y se deja de lado aquellas que no lo son desde el punto de vista del problema para así obtener una visión clara del mismo.

Encapsulamiento: Significa reunir en el mismo nivel de abstracción los elementos pertenecientes a una entidad y así tener una buena estructuración. Se emplea conjuntamente con la *ocultación* que consiste en proteger las propiedades de un objeto para que no sean modificadas por quien no deba, de modo tal que el estado del objeto debe ser responsabilidad de este y lo modifica por medio de sus procedimientos.

Herencia: Con la herencia se generan nuevas clases a partir de clases base existentes, lo cual facilita el rehuso de código. Booch la define como: "una clasificación u ordenamiento de la abstracción". Por ejemplo: a partir de la abstracción animal, se generan las abstracciones aves y mamíferos las cuales además de tener los atributos y comportamientos de animal, especifican los propios.

Polimorfismo: Está relacionado con el hecho de que existen comportamientos similares (en resultados) pero con implementaciones distintas. Por ejemplo: para ir de una cuidad A a una ciudad B los aviones se desplazan por aire, mientras que los automóviles van por las carreteras. Avión y automóvil son abstracciones de medios de transporte y ejecutan la misma función desplazarse, pero de distintas maneras.

Relaciones Entre Clases: Dado que la POO busca modelar el mundo real, en la ejecución de un programa se debe modelar el trabajo conjunto de los diferentes objetos, como sucede en el mundo real. Es por esta razón que las clases se relacionan entre sí de diversas formas. Los principales tipos de relaciones que se presentan son:

- *Generalización:* Es quizá la relación más importante ya que hace fuerte uso de la herencia. Uno de los motivos de las relaciones entre clases es el hecho de poseer propiedades comunes. Las clases con propiedades comunes se "agrupan" en superclases. Una superclase representa una generalización de las subclases. Ejemplo: La clase Animal es una generalización de las subclases Perro y Gato.

- *Especialización:* Una subclase que hereda de una superclase es una especialización de esa clase padre. Ejemplo: Las clases Perro y Gato son especializaciones de la superclase Animal.

- *Agregación:* Se presenta cuando hay objetos compuestos. La agregación de objetos permite describir modelos del mundo real que se componen de otros modelos. Este es un concepto que se utiliza para expresar tipos de relaciones entre objetos parte-de (part-of) o tiene-un (has-a). Es útil cuando una clase *Todo* se compone de otras clases *Parte*, aunque si quitamos alguna de las partes entonces el todo seguirá funcionando normalmente. Ejemplo: Una escuela tiene Biblioteca y Cafetería, aunque se quite la cafetería sigue siendo una escuela.

- *Asociación:* Permirte asociar objetos que colaboran entre sí (un objeto usa información de otro). Esta no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro. En esta asociación se emplea la cardinalidad que indica la cantidad de objetos usados: uno a uno, uno a muchos, muchos a uno.

- *Composición:* Es una relación similar a la Agregación con la diferencia que si una de las clases *Parte* falta, la clase *Todo* se vería afectada.

2.2.2. Bases de Datos

Se puede afirmar que una base de datos es un conjunto estructurado de datos que modela entidades del mundo real y sus interrelaciones, almacenados para su posterior uso y consulta. Sus orígenes se remontan a la Antigüedad donde ya existían bibliotecas y registros de datos de censos y cosechas. El uso de los sistemas de gestión de bases de datos se desarrolló a partir de la necesidad de manejar y consultar grandes cantidades de información generadas por las nuevas industrias (Chapple, 2009).

Existen diversos tipos de bases de datos que surgieron dependiendo del tipo de estructura de información que se quería gestionar con el paso del tiempo y de las aplicaciones para tal propósito. Los tipos de bases de datos más destacados son:

- *Relacional:* Se fundamenta en la lógica de predicados y la teoría de conjuntos. Emplea las relación entre tablas que se podrían considerar como conjuntos de datos llamados tupas. Sus orígenes datan desde 1970 cuenado el Dr. Edgar Frank Codd publica el documento "A

Relational Model of data for Large Shared Data". Las bases de datos relacionales ofrecen flexibilidad estructural, pues las aplicaciones escritas para esas bases de datos son más fáciles de mantener que las aplicaciones hechas para bases de datos jerárquicas o de red. Esa misma flexibilidad estructural permite recuperar combinaciones de datos que es posible que no pueda haber previsto en el momento del diseño de la base de datos.

- *Jerárquico:* Era utilizado por los primeros SGBD, desde que IBM lo definió para su Sistema Administrador de Información en 1970. También se conoce como modelo de árbol. La información se organiza con un jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo padre / hijo. Existen una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre). Está en deshuso.

- *En Red:* organiza la información en registros (también llamados nodos) y enlaces. En los registros se almacenan los datos, mientras que los enlaces permiten relacionar estos datos. Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre. En este modelo se pueden representar perfectamente cualquier tipo de relación entre los datos (aunque el Codasyl restringía un poco las relaciones posibles), pero hace muy complicado su manejo.

- *Orientado a Objetos:* La programación orientada a objetos permite cohesionar datos y procedimientos, haciendo que se diseñen estructuras que poseen datos (atributos) en las que se definen los procedimientos (operaciones) que pueden realizar con los datos. En las bases orientadas a objetos se utiliza esta misma idea. A través de este concepto se intenta hacer frente a las limitaciones de las relacionales. Por ejemplo el problema de la herencia (el hecho de que no se puedan realizar relaciones de herencia entre las tablas), tipos definidos por el usuario, disparadores (triggers) almacenables en la base de datos, soporte multimedia...

- *Objeto Relacionales:* Son un híbrido entre las relacionales y las orientadas a objetos. La mayoría de las últimas versiones de bases de datos y sistemas gestores son objeto-relacionales y gran parte de las bases de datos relacionales se ha migrado a este modelo, como lo es el caso de Oracle, SQL, Informix, entre otras (Chapple, 2009).

  En las bases de datos relacionales u objeto relacionales, se manejan dos conceptos clave: *entidades* que son la representación en el modelos de los objetos o elementos reales del problema o sistema que se desea modelar y las *relaciones* que son los elementos de asociación entre las entidades. Como se mencionó anteriormente, se emplea teoría de conjuntos, así pues, las tablas se relacionan a través de una o más columnas con otras tablas. En estas bases de datos se manejan uniones e intersecciones, de estas últimas existen diversos tipos que se presentan en la Figura.2.8 y que permiten obtener información de varias entidades relacionadas.

  Los principales elementos que presentan las bases de datos en SQL son:

  Tablas: Una tabla se utiliza para almacenar la información y representar algo que de lo cual se desea almacenar datos. Una tabla puede utilizarse para representar a las personas, lugares, cosas o ideas sobre los cuales necesita ser almacenada la información. Las tablas son básicamente estructuras que tienen instancias (filas) de un concepto (como una persona) y la información de los atributos relevantes de dicho concepto en un contexto determinado (columnas) (Davidson and Moss, 2012).

  Vistas: Una vista en una tabla virtual (se comporta como una tabla pero no tiene independencia existencial) que permite obtener información específica de una o más tablas. Desde una vista no se puede modificar la información directamente. los objetivos de una vista pueden ser varios: esconder las tablas originales, limitar el acceso a los datos (seguridad), simplificar y/o optimizar la extracción de datos, etc (Elmasri et al., 2002).

Procedimientos Almacenados: Las instrucciones en sql una vez ejecutados se pierden, por esto es necesario tener bloques de código nombrados que se puedan almacenar y ejecutar cuando se requieran. Un procedimiento es un segmento de código que puede llamarse por el nombre que se ha dado y ejecutar acciones sobre la base de datos. El procedimiento puede tener cero o varios parámetros de entrada y/o salida.

Triggers: Son bloques de código asociados a una tabla que se ejecutan cuando ocurre algún evento sobre dicha tabla.

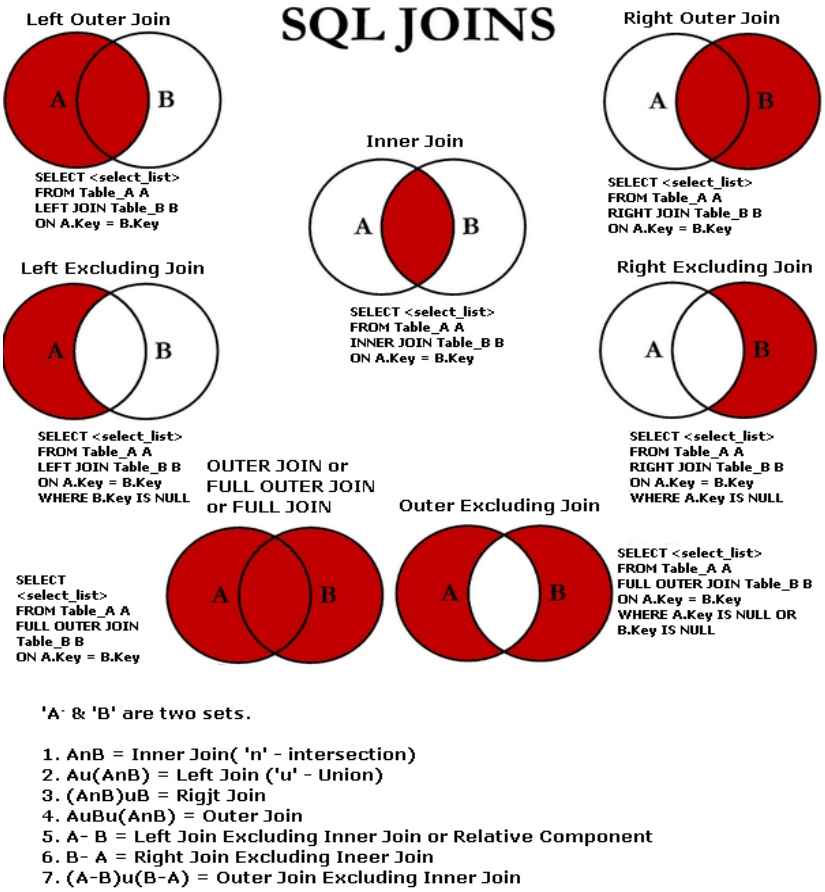


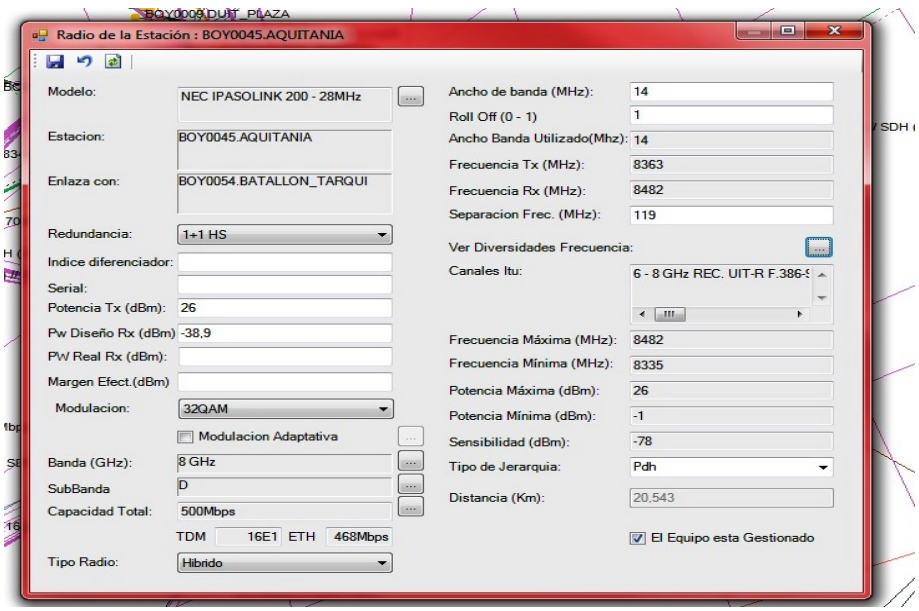
Figura 2.8: SQL joins

Capítulo 3

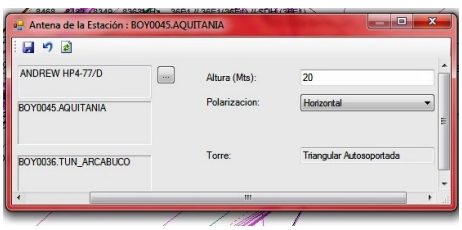
DESARROLLO

3.1. Simulación Enlaces Nuevos

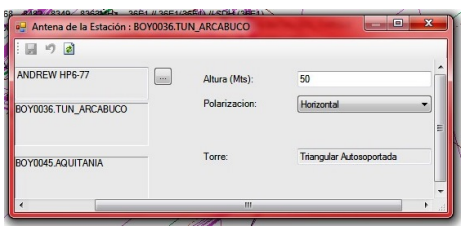
Para crear un enlace en TesGestion se deben seleccionar las estaciones origen y destino desde el mapa para luego proceder a ingresar los datos de radios (Figura 3.1(a)), antenas y torres (Figuras 3.1(b) y 3.1(c)) en unos navegadores creados para tal fin y finalmente crear el enlace.



(a) Navegador datos radios



(b) Datos antena estación A



(c) Datos antena estación B

Figura 3.1: Capturadores datos creación enlaces en TesGestion

El primer paso es agregar un navegador para que no permita crear el enlace sin tener los resultados de la simulación en el cual se deben incluir los botones respectivos para la gestión de la simulación y los resultados, así como una opción para aquellos usuarios con permisos especiales la cual permita crear el enlace sin simular.

La simulación en Spectrum-E se desarrolla a partir de la información de unas redes de enlaces existentes, que en este caso son las redes instaladas y aprobadas por el MinTIC, y los enlaces a simular, que son creados, cargados y guardados en la base de datos de Spectrum-E por el usuario para luego ser consultados por Spectrum-E y realizar la simulación. Para este caso, los datos deben cargarse en el simulador desde TesGestion por los motivos expresados en la identificación del problema y por tratarse de información confidencial y que debe ser manejada cuidadosamente se ha determinado realizar una etapa de exploración e investigación para determinar la forma de pasar estos datos desde TesGestion hacia el simulador.

El desarrollo de este módulo en TesGestion, se planificó en un Sprint con dos historias: "Simular enlaces planeados a instalar" y "Almacenar resultados en base de datos". Para lo cual se revisó lo que era necesario para desarrollarla y se obtuvo el siguiente diagnóstico para sacar una lista de tareas. Así pues, nuestras historias fueron descompuestas en las siguientes tareas:

- Crear navegador de simulación de enlaces.
- Explorar qué datos se necesitan para desarrollar la simulación y la manera de pasarlos desde TesGestion.
- Crear base de datos para almacenar los resultados.
- Crear acceso a base de datos para gestionar resultados.
- Crear procedimientos almacenados en base de datos para gestión de resultados.
- Crear procedimientos para hacer conexión al simulador y pasar información.
- Desarrollar algoritmo de validación y verificación de resultados.
- Crear algoritmo para adjuntar resultados a legalización.
- Modificar base de datos de TesGestion para asociar archivos a enlaces.

Luego de desarrollar la exploración sobre el simulador, se pudo identificar que la manera más práctica y sencilla de hacer el paso de los parámetros del enlace a simular, es a través del URL. Los parámetros que se requieren para desarrollar la simulación por cada estación del enlace son:

- Nombre de la estación.
- Longitud.
- Latitud.
- Elevación (m.s.n.m.).
- Altura a la que se encuentra la antena.
- Frecuencia de TX.
- Polarización.
- Potencia nominal.
- Ganancia.

- Pérdidas de RX.
- Pérdidas de TX.
- Pérdidas adicionales.

La Figura 3.2 describe la operación del sistema que se ha diseñado.

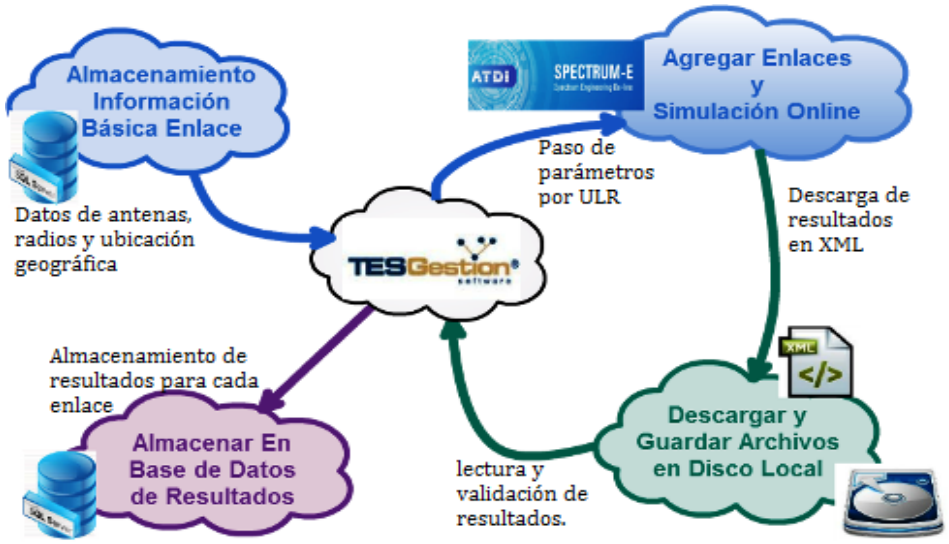


Figura 3.2: Diagrama general sistema

Gran parte de estos datos se obtienen de los capturadores que se presentan en la Figura 3.1, y otros, que dependen de las características de los modelos de radio o de la antena se obtiene a través del llamado a un procedimiento almacenado que retorna la información del modelo del equipo que se ha seleccionado en el respectivo capturador. En el código es importante mantener la arquitectura en capas con la que se ha desarrollado todo el sistema y mantener el principio de *separation of concerns* que básicamente indica que cada capa debe cumplir únicamente con sus responsabilidades.La Figura 3.3 muestra el diagrama de la arquitectura de capas y a continuación se describen las capas que la componen:

- Capa de datos: Repositorio de datos, es decir el servidor de datos donde se almacena la base de datos.
- Capa de entidades: Contiene cada una de las tablas de la base de datos convertida en clases con sus respectivas propiedades y es común a las demás capas.
- Capa de acceso a datos: Será la interfaz entre el programa y la base de datos, se encarga de que se ejecuten todas las sentencias de consulta, inserción, modificación y eliminación de datos. Interactúa con las capas de datos, entidades y de lógica.
- Capa de lógica de negocio: En esta capa se implementan todas las reglas del negocio, es decir lo que se debe cumplir antes de realizar alguna acción. Interactúa con las capas de presentación, acceso a datos y entidades
- Capa de presentación: Es la capa donde se implementa toda la interfaz gráfica que actúa con el usuario, la capa de lógica y de entidades.

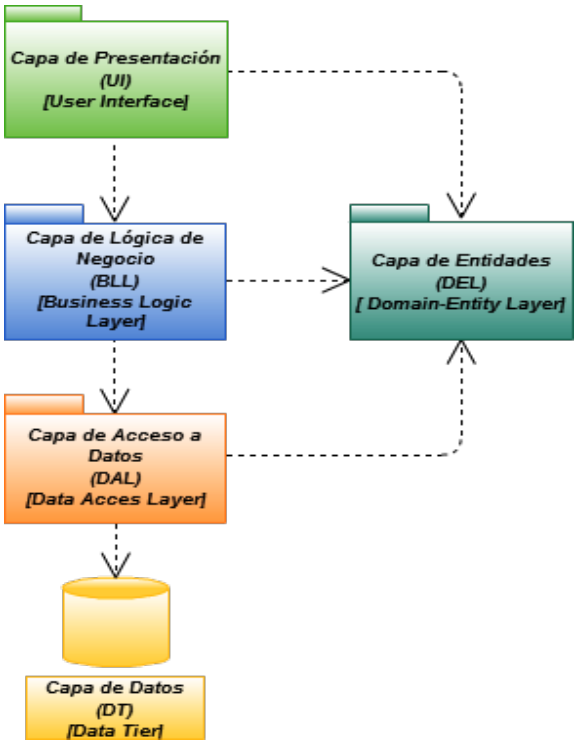


Figura 3.3: Diagrama de capas

3.1.1. Capa de Presentación

Se creó un formulario con un botón para iniciar la simulación, otro para anexar los resultados que se obtienen, otro para hacer el retorno a los capturadores donde el usuario podrá modificar los parámetros del enlace y otro para crear el registro del enlace y los elementos asociados a este en la base de datos. De igual forma se agregó un Checkbox para habilitar a usuarios con permisos especiales sobre la herramienta la opción de crear el enlace sin necesidad de desarrollar la simulación. Debido a políticas de confidencialidad no es posible mostrar el código implementado de manera explícita, así que en la Figura 3.4 se presenta el formulario en el entorno de Visual Studio con algunas propiedades.

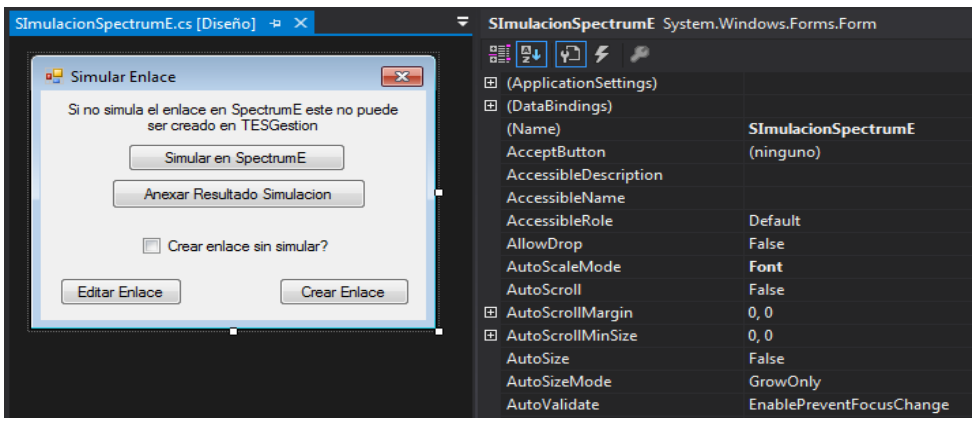


Figura 3.4: Navegador de simulación en Visual Studio

A cada botón se le asoció un evento onClick el cual se hace el llamado a un método determinado de la capa de lógica para que haga la ejecución de la tarea respectiva.

3.1.2. Capa de Datos

En el servidor de datos se creó una base de datos de resultados de simulaciones a fin de no sobrecargar la base de datos principal, se creó una tabla la cual almacenará la información del identificador único del enlace, la información de cada archivo de resultados en formato binario, y el respectivo nombre de cada archivo con una descripción adicional. La Tabla 3.1 muestra los campos de la tabla que almacenará los resultados y el tipo de dato que soporta cada uno, mientras que la Figura 3.5 presenta la tabla en un diagrama de base de datos en SQL.

Tabla 3.1: Columnas Tabla Resultados.

Columna	Tipo Dato	Descripción
IdEnlace	Uniqueidentifer	Identificador único del enlace simulado
Simulacion1	VarBinary	Archivo resultado interferencia hacia la red en formato binario
Simulacion2	VarBinary	Archivo resultado interferencia desde la red en formato binario
Archivo1	Varchar	Nombre archivo resultado interferencia hacia la red
Archivo2	Varchar	Nombre archivo resultado interferencia desde la red

Además de la tabla, se agregaron a la base de datos los procedimiento almacenados que permiten hacer las operaciones de CRUD (Create, Read, Update, Delete) de registros almacenados en la tabla de simulaciones. En cuanto a los procedimientos asociados a la inserción de registros en tabla de enlaces se realizó la respectiva actualización para que se incluya la información relacionada a este nuevo módulo.

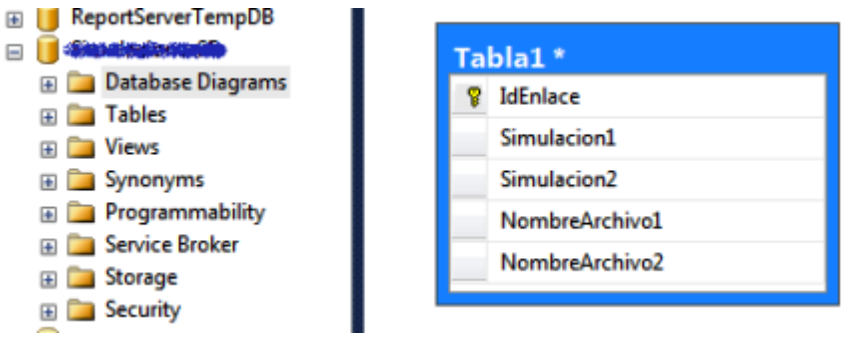


Figura 3.5: Tabla de resultados en base de datos

Por otra parte, en la tabla de enlaces se agregó una columna de tipo bolean que especifica si el enlace tiene asociados los archivos de simulación, esta columna no debe soportar valores nulos y el valor por defecto debe ser false. El Segmento de Código 3.1 presenta el algoritmo desarrollado en SQL para dicha actualización de la tabla de Enlaces; al tener un valor por defecto, los registros ya existentes se inicializan con este por lo cual no es necesario hacer un volcado de información a una nueva tabla.

```
-- SE CREA UNA TRANSACCION
BEGIN TRANSACTION
GO

ALTER TABLE dbo.EnlacesMicroondas SET (LOCK_ESCALATION = TABLE)
GO

-- SE AGREGA UNA COLUMNA DE TIPO BIT CON VALOR POR DEFECTO = 0
ALTER TABLE dbo.EnlacesMicroondas ADD
  TieneSimulacionSE BIT NOT NULL DEFAULT 0
GO

COMMIT --APLICA EL CAMBIO
```

Segmento de Código 3.1: Codigo actualización tabla de enlaces en SQL

3.1.3. Capa de Entidades

Como se mencionó anteriormente, la capa de entidades es la representación de las tablas de la base de datos en forma de clases, así pues se genera la clase resultados, la clase que estará compuesta por las propiedades públicas que se muestran en la Tabla 3.2

Tabla 3.2: Propiedades clase Resultados.

Propiedad	Tipo	Descripción
IdEnlace	System.Guid	Identificador único del enlace simulado
Simulacion1	byte[]	Array de bytes del archivo resultado de interferencia hacia la red
Simulacion2	byte[]	Array de bytes del archivo resultado de interferencia desde la red
NombreArchivo1	string	Nombre archivo resultado interferencia hacia la red
NombreArchivo2	string	Nombre archivo resultado interferencia desde la red

Para mantener la información durante todo el proceso de creación se cuenta con una clase que contiene todos los datos del enlace a crear cuyo diagrama se muestra en la Figura 3.6; de esta manera mientras la instancia de dicha clase exista se mantendrá la información invariante a menos que el usuario la modifique a través de los capturadores; lo anterior permite al usuario, en caso que la simulación resulte fallida, poder regresar al capturador y modificar la información necesaria para volver a realizar la simulación, sin necesidad de volver a diligenciar todos los campos.

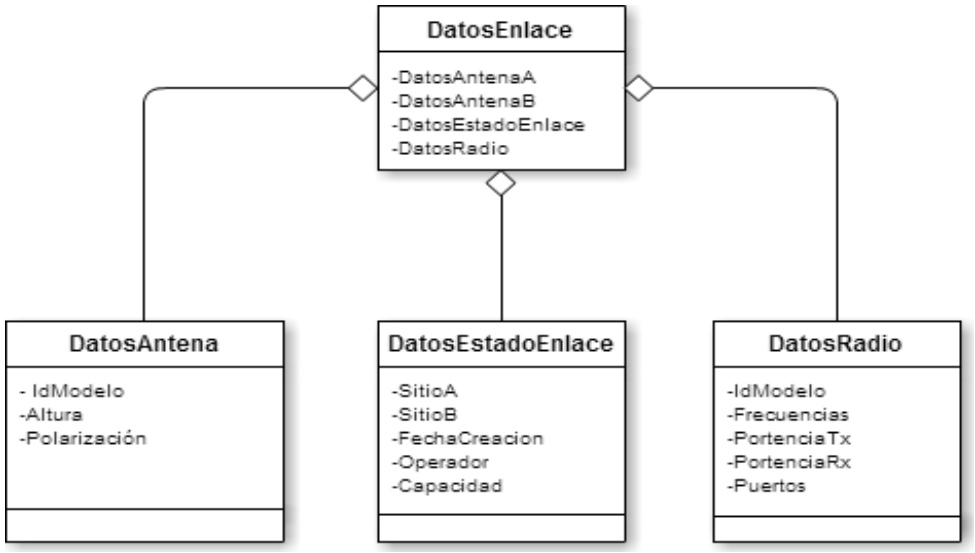


Figura 3.6: Diagrama de clases para los datos del enlace

3.1.4. Capa de Acceso a Datos

En esta capa se cuenta con los siguientes elementos:

- Database: Representación de la base de datos en el código en C#. La instancia de este elemento se creó a partir de la cadena de conexión que contiene la ruta del servidor de base de datos, el nombre de la base de datos y los parámetros de seguridad que requiere para realizar la conexión con el motor de base de datos con la sintaxis presentada en el Segmento de Código 3.2:

```
private static Database _dataBase = new Microsoft.Practices.EnterpriseLibrary.Data.Sql.
    SqlDatabase("ConnectionString");
```

Segmento de Código 3.2: Instanciación Base de Datos en C

- DbCommand: Representa una instrucción SQL o un procedimiento almacenado que se van a ejecutar en un origen de datos. Para leer datos generados por un procedimiento almacenado se emplea el Segmento de Código 3.3:

```
public static Type ExceuteProcedimiento(Parametro 1, Parametro 2, ... , Parametro n)
{
 using (DbCommand command = _dataBase.GetStoredProcCommand("StoredProcedure"))
 {
   //Para agregar los parametros de entrada al procedimiento almacenado
   _db.AddInParameter(command, "NombreParametro", DbType, ValorParemetro);

   //Para leer los resultados de una consulta con un DataReader:
   using (IDataReader reader = _db.ExecuteReader(command))
   {
     while (reader.Read())
         {
             //se crean los objetos en memoria a partir de los datos del reader
         }
```

```
    }
 }
}
```

Segmento de Código 3.3: Comando de lectura de registros de Base de Datos

Para la ejecución de procedimientos de inserción, actualización o eliminación de registros se hace necesario emplear una transacción para obtener un control de estas operaciones y poder tomar decisiones en caso de que ocurra alguna falla. Para estos casos se emplean la sintaxis del algoritmo del Segmento de Código 3.4.

```
public static Type ProcedimientoEjecutarTransaccion(Parametro 1, Parametro 2, ... ,
   Parametro n, transaccion)
{
 using (DbCommand command = _dataBase.GetStoredProcCommand("StoredProcedure"))
 {
   //Para agregar los parametros de entrada al procedimiento almacenado
   _db.AddInParameter(command, "NombreParametro", DbType, ValorParemetro);

   if (transaccion != null) _db.ExecuteNonQuery(command, transaccion);
              else _db.ExecuteNonQuery(command);

   //Si se necesita algun parametro del procedimiento en base de datos se trae y se
    crea el objeto respectivo con la siguiente linea:
   object o = _db.GetParameterValue(command, "ValorParametro");
   if (o == DBNull.Value) return null;
   return (Type)o;
 }
}
```

Segmento de Código 3.4: Ejecución de procedimiento almacenado en base de datos

- DbTransaction: Representa una transacción SQL en código y se usa en operaciones de inserción, actualización o eliminación de registros en la base de datos. La sintaxis para su implementación se presenta en el Segmento de Código 3.5:

```
public static void Procedimiento(Parametro1, Parmetro2,...,Parmetro n)
 {
   try
   {
     using (DbConnection conexion = _db.CreateConnection())
     {
       conexion.Open();//abre la conexion con la Base de datos
       using (DbTransaction transaccion = conexion.BeginTransaction(IsolationLevel.
    Snapshot))//crea la transaccion
       {
         try
         {
           //Se ejecuta el procedimiento
           ProcedimientoEjecutarTransaccion(Parametro1, Parmetro2,...,Parmetro n,
    transaccion)
           transaccion.Commit();//Aplica los cambios en Base de Datos
         }
         catch(Exception ex)
         {
```

```
            transaccion.Rollback();//Rechaza los cambios en Base de Datos
            throw ex;
          }
        }
      }
    }
    catch (SqlException ex)
    {
      //Si es posible se reintenta la operacion
      Procedimiento(Parametro1, Parmetro2,...,Parmetro n)
    }
  }
```

Segmento de Código 3.5: Ejecución de una transacción desde C

Los bloques Try/Catch permiten controlar excepciones y evitar que la ejecución del programa se detenga abruptamente y se pueda realizar el manejo de éstas ya sea notificando del error o anulando la operación. La instrucción *Commit* lo que hace es aplicar los cambios generados en base de datos mientras que la instrucción *Rollback* permite deshacer los cambios que se hallan generado con la transacción. Vale la pena indicar que una transacción puede contener más de una operación de inserción, actualización o modificación. Por otra parte, la instrucción *Using* permite hacer el Dispose (eliminación) de los elementos en memoria para optimizar el manejo de la misma.

La Tabla 3.3 presenta los procedimientos agregados en la capa de Acceso a Datos y se describe su propósito.

Tabla 3.3: Procedimientos Agregados a la capa DAL

Procedimiento	Descripción
GetLoginSE	Permite obtener los datos de login del usuario para el acceso al simulador
UpdateLoginSE	Permite actualizar los datos de login del usuario para el acceso al simulador
GetCaracteristicasRadio	Permite obtener los datos del radio seleccionado
GetCaracteristicasAntena	Permite obtener los datos de la antena seleccionada
InsertArchivosSE	Permite insertar los archivos de resultados de las simulaciones en la base de datos y asociarlos al respectivo enlace. Como un enlace solo puede tener un par de archivos se revisa si el enlace ya tiene archivos asociados y los elimina
GetArchivosSE	Permite obtener los archivos asociados a un determinado enlace si éstos existen en la base de datos

3.1.5. Capa de Lógica

Como se mencionó, se creó un navegador encargado de gestionar la simulación, la asociación de los resultados al respectivo enlace y además permitir a usuarios con permisos especiales poder crear enlaces sin simular. Cada botón está subscrito a un evento *onClick* mediante el cual se llama a un procedimiento determinado. Se destaca el hecho de que este navegador se diseñó de tal forma que puede ser llamado desde el capturador de datos del enlace como desde el navegador de enlaces existentes. El constructor de esta ventana se muestra en el Segmento de Código 3.6

```
public SImulacionSpectrumE(DatosCreaEnlace datosEnlace, Coordenada Coordenada1,
    Coordenada coordenada2, Usuario usuario)
{
  // Aqu se asignan los atributos de la clase SImulacionSpectrumE necesarios para
     desarrollar las simulacion por cada estacion:
 Parametro1 = datosEnlace.Parametro1
 Parametro2 = datosEnlace.Parametro2
 .
 .
 .
 Parametron = datosEnlace.Parametron

 //Si tiene permisos especiales se habilita la opcin de crear enlace sin simular
 if (usuario.TienePermisos) CheckBox.Enabled = true;
           else CheckBox.Enabled = false;
}
```

Segmento de Código 3.6: Constructor ventana simulación

Al llamarse al método de simulación a través del respectivo evento se reúne la información técnica relevante para la simulación con la cual se genera la respectiva URL que contiene los parámetros necesarios para crear una entidad de enlace a simular en Spectrum-E. Al mismo tiempo se crea un proceso que abre el navegador de internet en el cual se direcciona la página dispuesta por la ANE para la simulación en línea; para dar acceso al simulador se emplea un registro de login cuyos datos se guardan en la base de datos de TesGestion. El código asociado al click del botón simular se presenta en el Segmento de Código 3.7.

```
private void Simular()
{
  Process.Start("http://ane.gov.co/");//Abre la pgina de la ANE
  Thread.Sleep(2500);//Tiempo de espera 2.5 segundos
  string linksimulador = "http://simulacion.ane.gov.co:8088/se/portal/ane/loginproc.php?
     ";//link del simulador
  linksimulador = AgregarParametros(linksimulador);//Agrega los parametros a la URL
  Process.Start(linksimulador);//Abre el simulador importandole un nuevo enlace a la red
      de enlaces propuestos
  anexararchivo.Enabled = true; //Habilita el boton para la carga de archivos
}
```

Segmento de Código 3.7: Código del evento Click al botón Simular

Luego de haber importado la red a simular en Spectrum-E, corresponde al usuario ejecutar tanto la simulación de interferencia del enlace hacia la red existente como de la red existente hacia el enlace propuesto y descargar los archivos generados por el simulador con los resultados de cada simulación. Estos archivos se generan en un formato específico (XML) y deben ser almacenados en el ordenador del usuario. Una vez obtenidos los resultados los archivos son leídos e interpretados por TesGestion a través de ciertos procedimientos creados para extraer la información detallada y específica. El Segmento de Código 3.8 corresponde al código de carga de los archivos

```
private void Anexar()
{
  OpenFileDialog dialogoCarga = new OpenFileDialog();//Venatana de carga de archivos
  dialogoCarga.Filter = "Archivos XML|*.xml";//Solo archivos xml
  dialogoCarga.Multiselect = true;//Multiple seleccion
```

```
  //delegado para validar la carga de dos archivos
  dialogoCarga.FileOk += delegate(object s, CancelEventArgs ev)
  {
    if (dialogoCarga.FileNames.Count() != 2)
    {
      MessageBox.Show("Debe seleccionar 2 archivos, el de interferentes y el de
     interferidos.");
      ev.Cancel = true;
    }
  };

  if (dialogoCarga.ShowDialog() == DialogResult.OK)
  {
    filenames = dialogoCarga.FileNames;
    if (VerificarArchivoSpectrumE(filenames))//Si los archivos son correctos y no
     reportan errores
    {
      crearenlace.Enabled = true;
    }
    else
    {
      MessageBox.Show("Los archivos de la simulacion tienen resultados fallidos, por
     favor cambie los datos del enlace");
      filenames = null;
    }
  }
}
```

Segmento de Código 3.8: Código del evento Click al botón Anexar

Con la información interpretada, se procede a realizar la validación para verificar que no se registran reportes de interferencias y permitirle al usuario crear el respectivo enlace en la base de datos de TesGestion y asociar los archivos al enlace en una Base de Datos creada para tal fin. En caso que se encuentre algún reporte negativo, el usuario deberá regresar al navegador inicial y modificar la información relevante que considere necesaria para que el enlace no presente interferencias. Un punto importante y que se consideró esencial para facilitar el trabajo al usuario es mantener la información durante todo el tiempo, para que el usuario no tenga que volver a diligenciarla.

Los archivos generados por el simulador vienen en formato XML y contienen la información de los resultados del análisis del enlace propuesto contra los enlaces existentes. El archivo se compone por un encabezado, una raíz que contiene la descripción del tipo de simulación y los elementos o nodos que contienen la información del resultado de la simulación. El proceso de validación consiste en leer los nodos y buscar los elementos reportados como fallidos, además de asegurarnos que el archivo contenga todos los elementos y que no se reporten interferencias.

La Figura 3.7 muestra el algoritmo de verificación, en el cual primero se valida el formato del archivo, que su estructura sea correcta y corresponda al formato generado por el simulador, luego se valida que hayan elementos que leer y que existan elementos reportados como correctos y luego que no exista ningún elemento en el que se indique que hay interferencias. Una vez se han validado los resultados y no se reportan interferencias, se procede a crear el nuevo registro del enlace en la base de datos de TesGestion y se crea el respectivo registro del archivo binarizado en la base de datos de resultados. Se emplean dos bases separadas para no sobrecargar la base de datos de TesGestion con la información de los archivos.

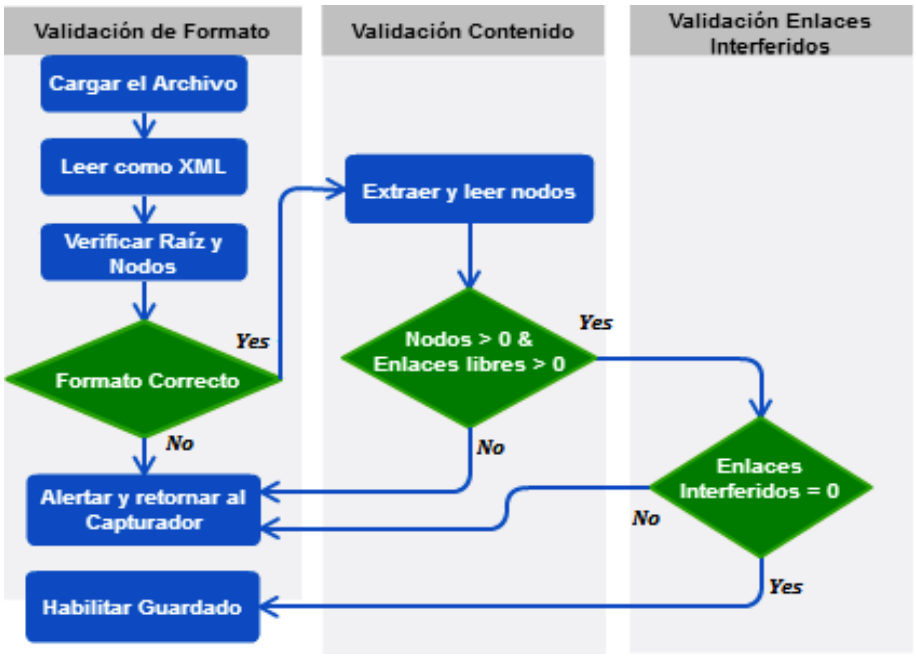


Figura 3.7: Diagrama de flujo validación

3.2. Simulación Enlaces Existentes y Múltiples Enlaces

Un elemento importante que el cliente solicitó fue el poder realizar la simulación con los enlaces existentes y desarrollar la simulación de múltiples enlaces. Así pues, se planeó el Sprint correspondiente con una historia: Permitir simulación de enlaces existentes y múltiples enlaces.

Para esta historia se definen las siguientes tareas:

- Crear botón y método para vincular ventana de simulación con navegador de enlaces.
- Crear acceso a datos para extraer información de la Base de Datos.
- Modificar Navegador de enlaces para agregar información de simulación.
- Crear métodos para anexar y adjuntar archivos a enlaces.
- Investigar cómo se ejecuta la simulación en batch en Spectrum-E.
- Crear procedimiento para aplicar simulación en batch.

3.2.1. Capa de Presentación

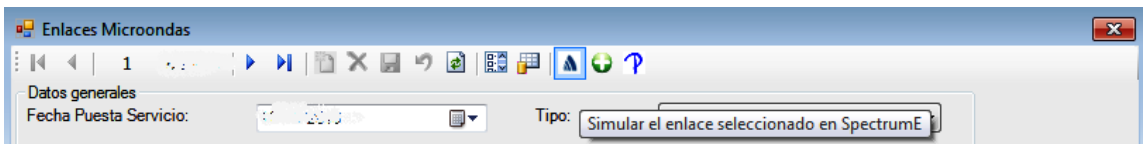
En el navegador de enlaces microondas existentes se agregaron dos botones, uno para ejecutar la simulación y otro para anexar resultados de simulaciones ya realizadas. Cada botón tiene sus respectivos eventos asociados desde los cuales se ejecutan los procedimientos creados en el desarrollo de la función de simulación de enlaces nuevos con la variación de la forma en que se inicializa la ventana de simulación, es decir se emplea un constructor diferente.

```
//se crean los botones
botonSimular = new ToolStripButton();
botonAnexarResultados = new ToolStripButton();
//se definen las propiedades
```

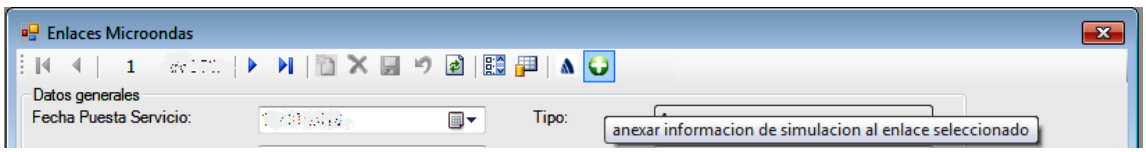
```
botonSimularSpectrumE.Enabled = true;
.
.
.
botonSimularSpectrumE.Text = "Simular el enlace seleccionado en SpectrumE";//Muestra
  la funcion del boton
botonSimularSpectrumE.Click += new EventHandler(SimularEnlace);//Llama

botonAnexarResultados.Enabled = true;
.
.
.
botonAnexarResultados.Text = "anexar informacion de simulacion al enlace seleccionado"
  ;
botonAnexarResultados.Click += new EventHandler(anexarinformacion);
```

Segmento de Código 3.9: Creación de botones en ventana de Enlaces

El Segmento de Código 3.9 genera los botones que se presentan en las Figuras 3.8(a) y 3.8(b) respectivamente.



(a) BotonSimular



(b) BotonAnexar

Figura 3.8: Botones en ventana de enlaces

3.2.2. Capa de Datos

Se actualizaron los procedimientos almacenados de actualización, consulta y eliminación, así como las vistas que se relacionan con los registros de la tabla de enlaces microondas con el fin de incluir la información correspondiente a este nuevo módulo. Para ello se usa el Segmento de Código 3.10

```
ALTER PROCEDURE [dbo].[Nombre_Procedimiento]
@ParametroX Type
AS
SET NOCOUNT ON

SELECT
  Id, ..., TieneSimulacion--Se indica si tiene simulacion
FROM
  NombreTabla_Vista
Where
  NombreTabla_Vista.ColumnaX=@ParametroX

--Procedimientos de Insercion
ALTER PROCEDURE [dbo].[NombreProcedimiento]
```

```
@Parametro1 Type,@Parametro2 Type,...,@TieneSimulacion BIT
AS
SET NOCOUNT ON
INSERT INTO Nombre_Tabla (Columna1, Columna2,...,TieneSimulacion)
VALUES (@Parametro1,@Parametro2,...,@TieneSimulacion)

--Procedimientos de Actualizacion
ALTER PROCEDURE [dbo].[NombreProcedimiento]
@Parametro1 Type, @Parametro2 Type, ..., @Parametro_n Type, @TieneSimulacion BIT
AS
SET NOCOUNT ON

IF EXISTS (Select [Columna1] FROM Nombre_Tabla WHERE [Columna1] = @Parametro1 AND [
    Columna2] = @Parametro1 AND...[Columnan] = @Parametro_n)
BEGIN
  UPDATE
    [Nombre_Tabla]
  SET
    [Columna1] = @Parametro1, [Columna2] = @Parametro1, ..., [Columnan] = @Parametro_n,
    [TieneSimulacion] = @TieneSimulacion
  WHERE
    [Columna1] = @Parametro1 AND [Columna2] = @Parametro1 AND...[Columnan] =
    @Parametro_n
  IF @@ERROR <> 0 BEGIN RAISERROR ('Error modificando el registro', 16, 1) RETURN END
END
--El registro no se encontro
ELSE BEGIN RAISERROR ('No se encontro el Registro o alguien lo ha modificado', 16, 1)
    RETURN END

--Procedimientos de Eliminacion
ALTER PROCEDURE [dbo].[NombreProcedimiento]
@Parametro1 Type, @Parametro2 Type, ..., @Parametro_n Type
AS
SET NOCOUNT ON
DELETE FROM [BaseSimulaciones].[dbo].[TablaSimulaciones]
where TablaSimulaciones.IdEnlace = @Parametro1

DELETE FROM [BaseTesGestion].[dbo].[Nombre_Tabla]
where Nombre_Tabla.IdEnlace = @Parametro1 and Nombre_Tabla.Columna2=@Parametro2 and...
    Nombre_Tabla.Columnan=@Parametro_n
```

Segmento de Código 3.10: Actualización procedimientos almacenados

Además, se creó un procedimiento almacenado que retorna la información del enlace a partir del identificador único de éste. Dicho procedimiento permite reunir la información necesaria para generar la simulación.

3.2.3. Capa de Acceso a Datos

Se creó un método para crear una instancia de la clase DatosEnlace a partir de un procedimiento almacenado que relaciona las diversas tablas que contienen dicha información, el procedimiento recibe como parámetro el identificador único del enlace. El Segmento de Código 3.11 es el correspondiente a dicho método

```
private static DatosEnlace GetDatosEnlaceMicroondasById(Guid id)
{
  DatosEnlace datosEnlace = new DatosEnlace();
  using (DbCommand GetEnlaces = _db.GetStoredProcCommand("NombreProcedimiento"))
```

```
  {
    _db.AddInParameter(GetEnlaces, "Id", DbType.Guid, id);
    using (IDataReader reader = _db.ExecuteReader(GetEnlaces))
    {
      while (reader.Read())
      {
        datosEnlaces.Atributo1 = reader.GetString((int)EnumDatosEnlaceMicroondas.
    Atributo1);
        datosEnlaces.Atributo2 = reader.GetString((int)EnumDatosEnlaceMicroondas.
    Atributo2);
        .
        .
        .
        datosEnlaces.Atributo_n = reader.GetDouble((int)EnumDatosEnlaceMicroondas.
    Atributo_n);
      }
    }
  }
  return datosEnlace;
}
```

Segmento de Código 3.11: Creación DatosEnlace desde procedimiento almacenado

3.2.4. Capa de Lógica

Dado que el procedimiento de generación de la URL es el mismo para ambos casos (enlaces nuevos y existentes) luego de inicializar el navegador, empleando el constructor del Segmento de Código 3.12, se procede a hacer el llamado del método que genera dicha URL y que abre el simulador de la ANE donde se cargarán los datos respectivos para que el usuario pueda ejecutar la simulación y descargue los archivos de resultados. Dichos archivos pueden ser cargados a través del botón asignado para tal tarea en el navegador de enlaces.

```
public SImulacionSpectrumE(Guid? IdEnlaceExistente, DataRow datosEnlace, bool
    soloAnexarArchivos)
{
  InitializeComponent();//Inicializa la parte grfica
  DatosEnlace datosEnlaceExistente = AccesoDatosDinamico.GetDatosEnlaceMicroondasById(
    IdEnlaceExistente.Value);
  //  Aqu  se asignan los atributos de la clase SImulacionSpectrumE necesarios para
    desarrollar las simulacion por cada estacion:
  Parametro1 = datosEnlaceExistente.Parametro1
  Parametro2 = datosEnlaceExistente.Parametro2
  .
  .
  .
  Parametron = datosEnlaceExistente.Parametron
}
```

Segmento de Código 3.12: Constructor clase simulación para enlace existente

Este constructor tiene un parámetro de entrada de tipo booleano que permite establecer si se van a cargar los archivos o si se va a desarrollar la simulación, de tal manera que cuando se da clic en el botón Simular este parámetro debe ser false mientras que si se emplea desde el botón adjuntar este parámetro debería ser true.

Luego de obtener los resultados de las simulaciones, se procede a realizar la carga de los mismo a TesGestion, esto mediante el método asociado al botón de adjuntar archivos, el cual inicializa un objeto XmlReader por cada archivo y se carga con la información respectiva. Cada objeto se envía al método de validación para asegurar que las simulaciones están en el formato correcto y que no se reportan interferencias en ningún sentido, de ser así se hace el llamado al método diseñado para insertar los respectivos registros en la base de datos de resultados y actualizar la asociación del enlace con la respectiva simulación, en caso contrario se indica al usuario que existen reportes de interferencias por lo cual no es posible adjuntar los archivos.

3.2.5. Simulación Múltiples Enlaces

Finalmente, para la simulación de múltiples enlaces, se desarrolló un proceso de inspección en el simulador de SpectrumE para analizar la forma de ejecutar la simulación de múltiples enlaces. El código está generado en lenguaje PHP y contiene diversas versiones personalizadas de archivos para los diferentes países que lo emplean ya que en cada país la gestión y regulación del espectro es distinta. En la versión que pertenece a la ANE, se habilita la opción de selección múltiple en la página de selección de ítems a simular para que el usuario pueda elegir los enlaces que entrarán en el proceso de simulación.

Se toman los identificadores de los enlaces y se pasan por medio de un array a un procedimiento que emplea un iterador para desarrollar la simulación de cada enlace propuesto contra la red de enlaces existentes y contra los propuestos restantes y genere un único resultado. Por razones de confidencialidad no es posible presentar ni detallar la implementación de estos cambios ya que el código es propiedad intelectual de ATDI que es la empresa que desarrolló el simulador.

Capítulo 4

RESULTADOS

4.1. Análisis y Resultados del los Sprints

4.1.1. Sprint 1: Simulación Enlaces Nuevos

Luego de haber desarrollado las nuevas funcionalidades, se tienen los siguientes resultados:

En el primer sprint, se calculó un esfuerzo de 60 puntos, planeados a desarrollar en 15 días, la Figura 4.1 muestra el tablero de Scrum con las historias, tareas y el BurndownChart.

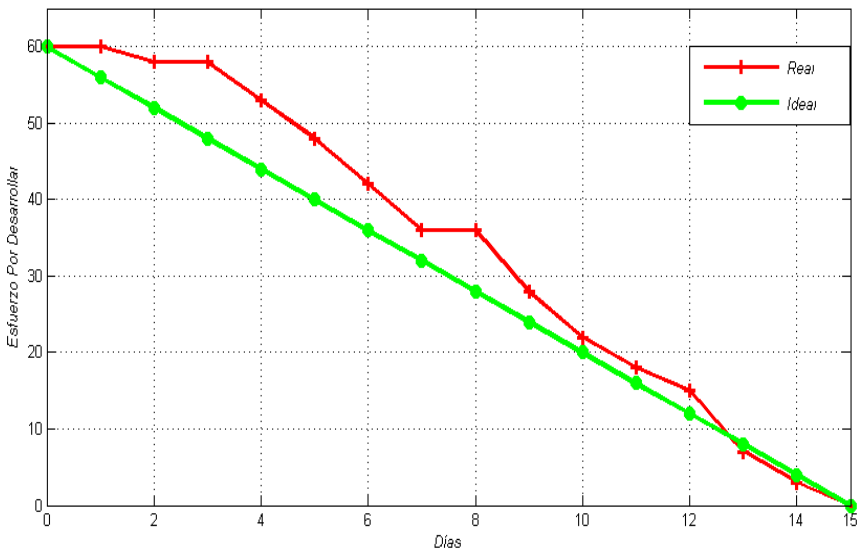


Figura 4.1: Burndown Chart Sprint 1

De la Figura 4.1 se puede afirmar que al inicio del Sprint aparentemente no se obtuvo ningún avance, esto se debió al hecho de que alguna o más tareas no se completaron en el transcurso de día ya que el esfuerzo que implicaba desarrollarla era muy alto para ejecutarlo en este periodo. Así mismo la gráfica indica que se presentaron impedimentos o situaciones externas al Sprint que afectaron la productividad del equipo los cuales fueron reportados al Scrum Master quien actuó en consecuencia para mantener al equipo enfocado en el desarrollo de la funcionalidad eliminando los impedimentos y manejando dichas situaciones externas, estas medidas permitieron a partir del tercer día, habiendo desarrollado la tarea más larga que consistió en analizar la forma en que el simulador recibiría los datos, obtener un progreso notorio con una velocidad de desarrollo casi constante que al séptimo día permite acercarse a la cantidad ideal de trabajo desarrollado. Aun así, en este punto se observa

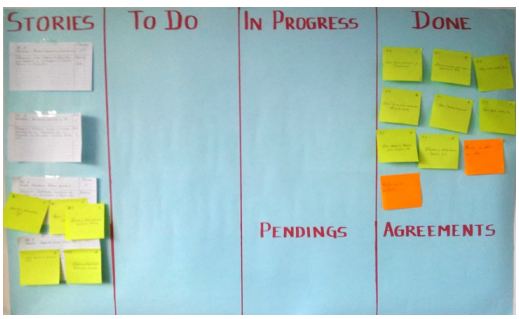
un aparente estancamiento, esto se debió a la necesidad de incluir una tarea que no fue planeada desde el inicio.

En los días siguientes se observa un ligero atraso con el cronograma situación que es normal y refleja que las historias y las tareas se estimaron un poco por debajo del esfuerzo real, sin embargo esta situación no es alarmante ni implica un riesgo total para el Sprint pues es consecuencia del margen de error que resulta de usar las series (Scrum Cards) para la estimación y finalmente en los últimos dos días se consigue finalizar el Sprint con todas las tareas desarrolladas para obtener como resultado un incremento funcional totalmente testeado y posteriormente presentado para recibir la retroalimentación respectiva en la reunión del Sprint Review.

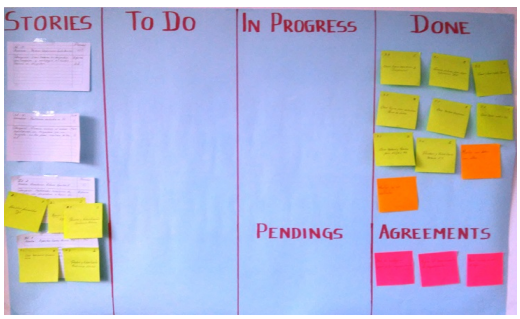
La Figura 4.2 presenta el tablero de Scrum en diferentes etapas del Sprint: al inicio (Figura 4.2(a)) con las historias en la parte izquierda, todas las tareas por hacer lógicamente, a mitad de transcurso del Sprint (Figura 4.2(b)) con diversas tareas finalizadas, otras en desarrollo y otras por hacer. Finalmente se presenta el tablero al final del Sprint (4.2(c)) con todas las tareas desarrolladas y sin ningún pendiente.

(a) Tablero Scrum al inicio del Sprint

(b) Tablero Scrum en la mitad del Sprint



(c) Tablero Scrum al final del Sprint



(d) Tablero Scrum al final del Sprint Retrospective

Figura 4.2: Tablero Scrum Sprint 1

Finalmente en la Sprint Retrospective Meeting se analizaron las diferentes situaciones y eventos que se presentaron durante el transcurso del Sprint; se analizaron las cosas buenas destacándolas para mantener al equipo animado y se presentaron estrategias para mantenerlas, aquellos factores que desanimaron al equipo y/o que no fueron muy satisfactorias así como las cosas que definitivamente afectaron negativamente a los miembros del equipo para luego analizarlas y plantear estrategias para resolverlas y para evitar que se vuelvan a presentar. El resultado de esta reunión son diversos compromisos adquiridos por los miembros del equipo y el Scrum Master y que se reflejan en la Figura4.2(d).

4.1.2. Sprint 2: Simulación Enlaces Existentes y Múltiples Enlaces

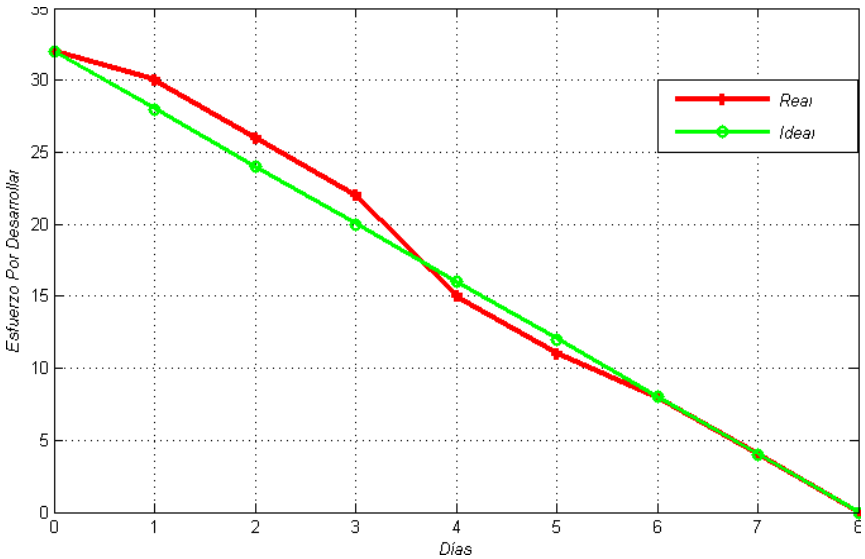


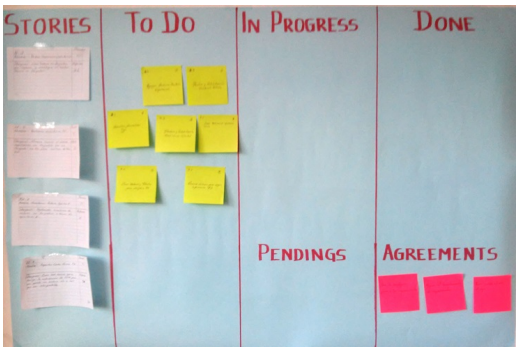
Figura 4.3: Scrum Burndown Chart Sprint 2

La Figura 4.3 presenta el Sprint Burndown Chart para el segundo sprint que se planeó con 32 puntos de historia para desarrollarlos en 8 días. De este gráfico se puede afirmar que tiene un comportamiento muy usual de un aparente atraso en los primeros días, pero luego se observa una recuperación del ritmo de trabajo, con una buena velocidad de desarrollo que permite incluso estar por debajo de la curva, sin embargo este comportamiento no es ni positivo ni negativo pues no es considerable la diferencia entre la estimación y la realidad del trabajo desarrollado sino que se puede interpretar nuevamente como el error normal de la estimación; de la gráfica se puede inferir que la estimación de las tareas fue bastante cercana a la realidad y que el equipo adquirió un ritmo de trabajo más estable y constante en comparación con el Sprint anterior.

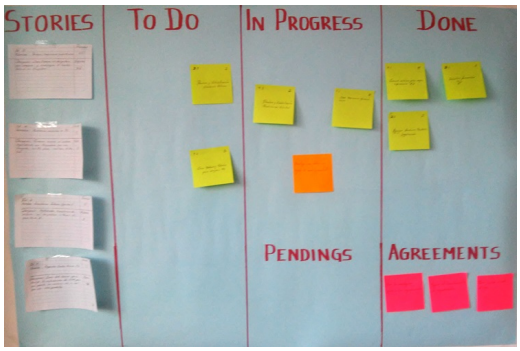
Esta mejora en el desempeño del equipo es consecuencia de las lecciones aprendidas en el Sprint previo y a los compromisos adquiridos por el equipo, las mejoras en la estimación y planeación de las tareas son posibles al diagnóstico que se hace en las reuniones de restrospectiva y la experiencia misma que han adquirido los integrantes del equipo. Esta evolución es consecuencia de la aplicación de la metodología que permite a intervalos cortos hacer diagnóstico del desempeño del equipo y obtener una retroalimentación del trabajo desarrollado por parte de las personas externas y del mismo equipo de desarrollo.

Dado que la gráfica del SBC depende también de las tareas que se hayan desarrollado, el análisis se realiza a diario junto con el tablero de Scrum donde se registra y se hace el seguimiento al trabajo desarrollado, el que se está trabajando y el que se falta por desarrollar para poder hacer una correcta interpretación del estado del Sprint y que el Scrum Master pueda tomar las acciones necesarias para asegurar el éxito de la iteración.

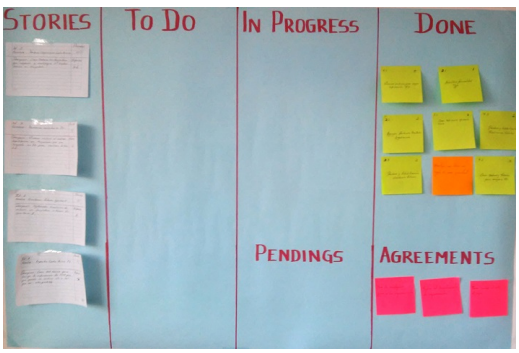
Al igual que con el Sprint 1, la Figura 4.4 presenta el tablero de Scrum en diferentes etapas del Sprint. Por último, luego de desarrollar el análisis del desarrollo del sprint durante en la Sprint Retrospective Meeting se obtuvo como resultado una serie de compromisos adquiridos por los miembros del equipo y el Scrum Master y que se reflejan en la Figura4.2(d) que buscan mantener las actitudes y características positivas del equipo y mejorar las características negativas para seguir madurando como equipo y ser cada vez más fuertes.



(a) Tablero Scrum al inicio del Sprint



(b) Tablero Scrum en la mitad del Sprint



(c) Tablero Scrum al final del Sprint

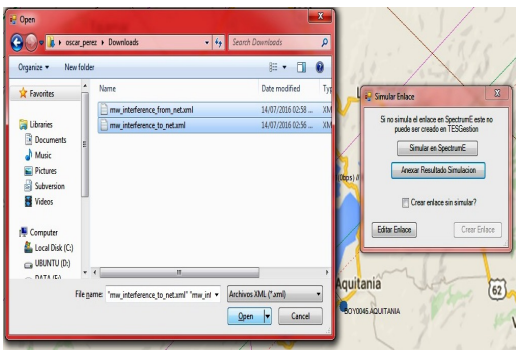
(d) Tablero Scrum al final del Sprint Retrospective

Figura 4.4: Tablero Scrum Sprint 2

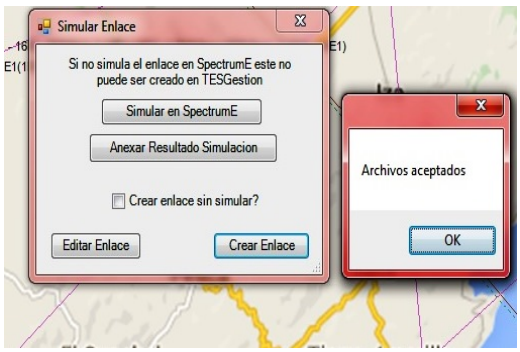
4.2. Análisis y Resultados del Desarrollo

La Figura 4.5(a) muestra el navegador diseñado y el flujo que se sigue luego de la configuración del enlace. La Figura 4.5(a) es la ventana siguiente a los capturadores mostrados en la Figura 3.1, al seleccionar la opción de *Simular en SpectrumE* se abre en el explorador de Internet la página de la ANE y el simulador Spectrum-E donde se carga el enlace propuesto como lo muestra la Figura 4.6(a). Luego de generar la simulación y descargar los archivos se habilita la opción *Anexar Resultados Simulación* tras lo cual se abre la ventana de la Figura 4.5(b) donde se seleccionan los archivos generados por el simulador y luego de ser procesados se muestra un mensaje de confirmación del éxito de la operación como se muestra en la Figura 4.5(c) y se habilita la opción *Crear Enlace* tras lo cual se genera el enlace como se puede apreciar en la Figura 4.5(d)

(a) Ventana para ejecutar simulación



(b) Carga de archivos de resultados

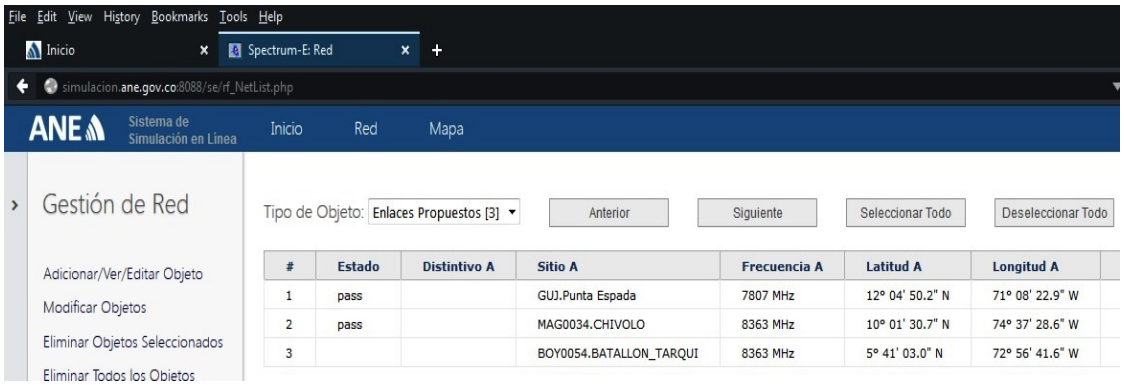


(c) Archivos aceptados

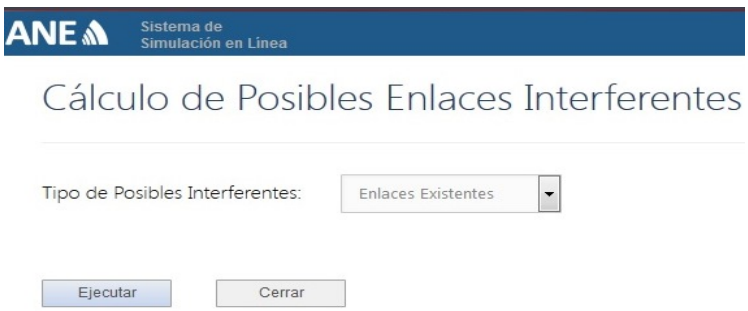
(d) Enlace creado en el mapa

Figura 4.5: Navegador de Final Creación

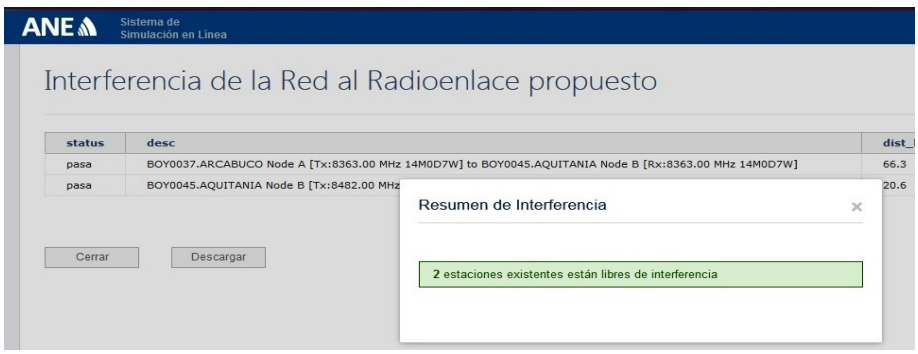
La Figura 4.6 muestra el simulador de la ANE con los enlaces importados a la red del usuario (4.6(a)) debidamente asignados a la lista de enlaces propuestos, en la imagen 4.6(b) se presenta la interfaz para la selección de enlaces a simular en ambos sentidos (desde la red y hacia la red) que se puede realizar tanto con enlaces existentes como con enlaces propuestos, sin embargo a solicitud del cliente para la validación solo se usan los resultados generados al simular contra los enlaces existentes. Finalmente en la imagen 4.6(c) se observa la interfaz que presenta los resultados de la simulación en la cual se informa al usuario los parámetros empleados y los resultados obtenidos; así mismo se presenta la opción de descarga de los archivos que luego serán importados y analizados por TesGestion para validarlos y proceder con la creación del enlace.



(a) Enlace Exportado A Spectrum-E



(b) Cálculo Posibles Interferentes



(c) Resultados Posibles Interferentes

Figura 4.6: Simulación en Spectrum-E

Luego de la liberación del producto, se llevó a cabo un proceso de seguimiento a la evolución del proyecto con el fin de analizar el impacto que la nueva funcionalidad generó en los procesos llevados a cabo por el cliente en la planeación e instalación de enlaces nuevos. Para ello, se hicieron mediciones periódicas a la cantidad de enlaces (nuevos) simulados que se registraron en TesGestion y se obtuvo como resultado la Tabla 4.1.

Tabla 4.1: Enlaces Simulados

Tiempo (Meses)	Enlaces Simulados
1	15
2	42
3	137
4	151
5	229
6	401
7	420
8	552
9	595
10	751

De los datos obtenidos, se observa que durante los primeros meses seguidos a la fecha de la liberación del producto con las nuevas funcionalidades la cantidad de enlaces simulados fue baja; este resultado puede atribuirse al hecho de que en principio solo un usuario fue designado para el desarrollo de las simulaciones y en los meses siguientes se incrementó el número de usuarios destinados a este propósito.

La Figura 4.7 presenta la cantidad de enlaces simulados que se registraron en función del tiempo. De esta gráfica se observa al inicio el fenómeno descrito anteriormente, a partir del segundo mes se observa el crecimiento en el número de enlaces simulados. Por otra parte se puede evidenciar la existencia de periodos en los cuales la cantidad de nuevos enlaces simulados tiene un mayor incremento en comparación con otros periodos. Al hacer la comparación con las convocatorias de procesos de selección objetiva para *Otorgar permisos para el uso del espectro radioeléctrico para los servicios fijos y móviles terrestres* que se adelantaron por parte del MinTIC, se encontró que los periodos coinciden con los meses en los que se ha dado la apertura a dichas convocatorias.

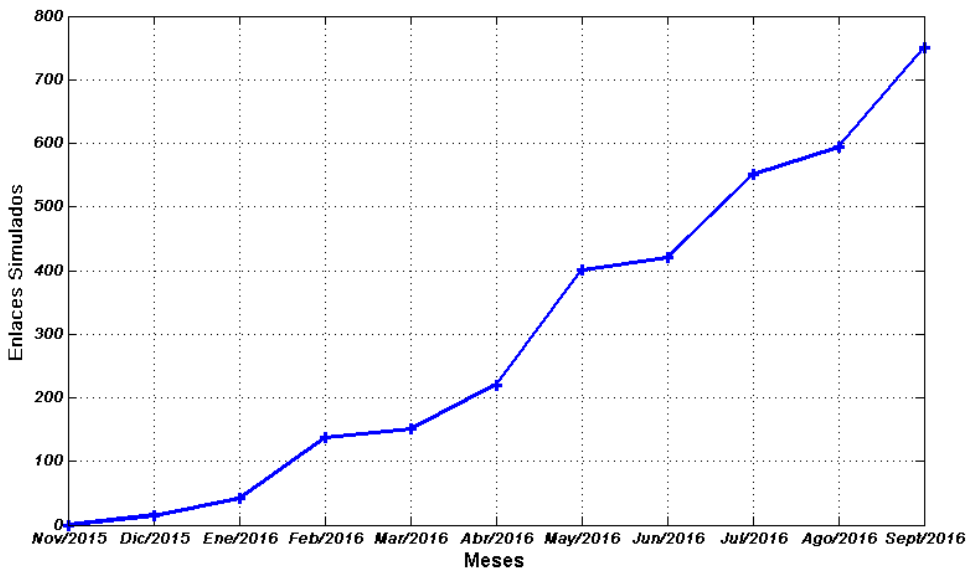


Figura 4.7: Enlaces simulados en el tiempo

Como se mencionó anteriormente, en el primer mes solo se contó con un usuario para desarrollar las simulaciones, inicialmente se le facilitó un manual básico que indicaba los cambios en el flujo del proceso normal para la creación de enlaces y manifestó que aunque se enfrentó a una funcionalidad nueva y a una herramienta desconocida, pudo desarrollar sus labores sin mayores inconvenientes,

sin embargo sugirió detallar mejor el manual para poder integrarlo al manual propio del aplicativo.

En consecuencia, con el objetivo de facilitar e incentivar el uso de esta extensión se tomó la decisión de generar un manual corto y mejor detallado en cuanto al proceso que permita al usuario tener una mayor claridad de este, luego de tener el manual desarrollado se optó por desarrollar unas sesiones de capacitación para el manejo de la nueva funcionalidad y del simulador de la ANE como medida de refuerzo.

Además de las cifras obtenidas en cuanto a los enlaces nuevos que poseen simulación asociada, en los registros de acceso al simulador se encuentra que en promedio se realizan 4 simulaciones diarias, también se tienen diversas versiones de redes en las que el usuario importa los enlaces desde TesGestion las cuales cuentan hasta con 4000 enlaces. Estos indicadores muestran que se obtiene una buena usabilidad de la nueva funcionalidad.

En cuanto al desempeño de la funcionalidad no se han registrado mayores inconvenientes, luego de la liberación se han reportado 6 incidentes en el proceso de simulación de enlaces. El primer incidente registrado fue causado por el uso de una versión desactualizada del software, luego de este primer reporte se registraron 2 incidentes que fueron causados por fallas en la conexión a la red, específicamente de usuarios que trabajan por VPN y finalmente los últimos incidentes fueron causados por una actualización en el simulador que requirió ciertos ajustes en la versión personalizada para la ANE. Teniendo en cuenta que el estimado de líneas de código no comentadas que componen esta funcionalidad es cercano 1500 líneas y que hasta el momento no se han reportado fallas que se puedan atribuir como tal a la herramienta se puede inferir que la calidad en cuanto a funcionamiento es alta.

Capítulo 5

CONCLUSIONES Y TRABAJO FUTURO

El nuevo módulo desarrollado representa una ventaja importante para el usuario ya que puede advertir y prevenir posibles factores de riesgo como las interferencias para la creación del enlace planeado. Al anticipar esto, reduce la posibilidad de que la solicitud de la asignación de frecuencias sea rechazada lo cual en el negocio de las telecomunicaciones es crítico pues los recursos que se invierten son elevados.

Si bien es cierto que este nuevo módulo permite anticipar posibles interferencias y que SpectrumE es una herramienta de simulación con un porcentaje de fiabilidad bastante elevado, cerca del 90 %, ésta no garantiza que no se puedan presentar interferencias en la realidad.

El desarrollo de este módulo es el principio de una actualización de la suite TESGestion que busca darle un mayor alcance e impacto al permitirle la conexión con una serie de múltiples herramientas para el uso y la gestión del espectro que emplean las entidades encargadas de esta labor en el marco de la Ley TIC (*Ley 1341 de 2009*) que estimula y promueve el uso de dichas herramienta entre otras disposiciones.

Como se evidencia en los resultados de usabilidad (Figura 4.7), el impacto que ha generado esta extensión en las funcionalidades de la plataforma en los procesos del cliente ha sido muy importante y da generado un alto valor agregado al producto. Por esta razón se debe procurar enfocar los esfuerzos en la integración de TesGestion con otras plataformas que generen un valor agregado similar al que se generó con la integración a Spectrum-E.

Para seguir adelante con esta actualización y avanzar en la misión de facilitar a los usuarios y clientes de la aplicación el desarrollo de sus procesos, se propone extender los servicios de TesGestion al proceso de radicación de solicitudes ante el MinTIC, de manera que además de registrarse la documentación de este proceso en la plataforma, se permita generar los documentos con la información técnica correspondiente a las solicitudes de adiciones, modificaciones y cancelaciones de redes en el Sistema de Gestión del Espectro (SGE) soportado por el MinTIC.

De la metodología se resalta el aporte que genera en la construcción y consolidación del equipo de desarrollo y la adaptación al cambio de requerimientos en el transcurso del proyecto. Sin embargo, se debe tener en cuenta que el fortalecimiento del equipo no se da únicamente por el empleo de la metodología y que no es sencillo lograrlo, se requiere que de un gran compromiso de los individuos y de una buena comunicación además de otros elementos que se adquieren con el transcurso del tiempo.

Índice Alfabético

Bases de Datos, 22
- *Elementos*
  - *Procedimientos Almacenados*, 24
  - *Tablas*, 23
  - *Triggers*, 24
  - *Vistas*, 23

CMMI, 6
- *CMM*, 6
  - *Áreas Claves de Proceso*, 7
  - *Capacidades del Proceso*, 7
  - *Funciones Comunes*, 7
  - *Niveles de madurez*, 7
  - *Objetivos*, 8
  - *Prácticas Claves*, 7

Metodología Scrum, 12
- *Artefactos*
  - *Product Backlog*, 16
  - *Sprint Backlog*, 17
  - *Sprint Burndown Chart*, 18
  - *Sprint Tasks*, 18
  - *User Stories*, 17
- *Eventos*
  - *Daily Scrum Meeting*, 19
  - *Product Backlog Refinement Meeting*, 18
  - *Sprint Planning Meeting*, 18
  - *Sprint Retrospective Meeting*, 19
  - *Sprint Review Meeting*, 19
- *Roles*
  - *Product Owner*, 15
  - *Scrum Master*, 15
  - *Scrum Team*, 14
  - *Stakeholders*, 16

Programación Orientada a Objetos, 20
- *Abstracción*, 21
- *Clase*, 21
- *Encapsulamiento*, 21
- *Herencia*, 22
- *Objetos*, 21
- *Polimorfismo*, 22

Relaciones Entre Clases, 22
- *Agregación*, 22
- *Asociación*, 22
- *Composición*, 22
- *Especialización*, 22, 23
- *Generalización*, 22

Lista de Acrónimos

ANE Agencia Nacional del Espectro

ANTV Autoridad Nacional de Televisión

CNTV Comisión Nacional de Televisión

DSM Daily Scrum Meeting

ERE Espectro Radio-Eléctrico

ITU International Telecommunication Union

MinTIC Ministerio de las Tecnologías de la Información y las Comunicaciones

OOP Object Oriented Programming

PB Product Backlog

PO Product Owner

SB Sprint Backlog

SBC Sprint Burndown Chart

SH Stakeholders

SM Scrum Master

SPM Sprint Planning Meeting

SR Sprint Review

SRM Sprint Retrospective Meeting

ST Scrum Team

TS Tareas del Sprint

URL Uniform Resource Locator

US User Stories

Bibliografía

Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 244–254. Ieee.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Boehm, B. W. et al. (1981). *Software engineering economics*, volume 197. Prentice-hall Englewood Cliffs (NJ).

Cadavid, A. N., Martínez, J. D. F., and Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, 11(2):30–39.

Chamorro, L. and Barbosa, A. (2011). Espectro abierto para el desarrollo estudio de caso: Colombia. *Colombia: Asociación para el Progreso de las Comunicaciones (APC)*.

Chapple, M. (2009). *Microsoft SQL Server 2008 for dummies*. John Wiley & Sons.

Chrissis, M. B., Konrad, M., and Shrum, S. (2011). *CMMI for development: guidelines for process integration and product improvement*. Pearson Education.

Clark, D. and Sanders, J. (2011). *Beginning C# object-oriented programming*. Springer.

Davidson, L. and Moss, J. M. (2012). *Pro SQL server 2012 relational database design and implementation*. Springer.

Deemer, P., Benefield, G., Larman, C., and Vodde, B. (2010). The scrum primer. *Scrum Primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, available at: http://assets. scrumtraininginstitute.com/downloads/1/scrumprimer121. pdf*, 1285931497:15.

Elmasri, R., Navathe, S. B., Castillo, V. C., Pérez, G. Z., and Espiga, B. G. (2002). *Fundamentos de sistemas de bases de datos*. Addison-Wesley.

James, M. (2010). Scrum reference card. *CollabNet Inc*.

Jeffries, R. (2011). What is extreme programming? *Disponible en: http://ronjeffries.com/xprog/what-is-extreme-programming/*.

Keith, C. (2010). *Agile Game Development with Scrum*. Pearson Education.

Kendal, S. (2014). *Object oriented programming using C Sharp*. Bookboon.

Kniberg, H. (2007). Scrum y xp desde las trincheras. *C4Media Inc. InfoQ*.

Mary Beth Chrissis, Mike Konrad, S. S. J. A. C.-M. V. (2012). *CMMI® para Desarrollo. Guía para la integración de procesos y la mejora de productos.* Pearson Education.

Muñoz, E. C., Blanco, H. J. E., and Calderón, J. A. F. (2015). Gestión del espectro radioeléctrico en colombia. *Tecnura*, 19(45):159–173.

Nakov, S. . C. (2013). *Fundamentals Of Computer Programming With C*. Free book. Disponible en http://www.introprogramming.info.

Palacio, J. (2014). Gestión de proyectos scrum manager. *Scrum Manager*.

Paulk, M. (1993). *Capability maturity model for software*. Wiley Online Library.

Schwaber, K., Sutherland, J., and Beedle, M. (2013). The definitive guide to scrum: The rules of the game. *Recuperado de: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf*.

Takeuchi, H. and Nonaka, I. (1986). The new new product development game. *Harvard business review*, 64(1):137–146.

Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological bulletin*, 63(6):384.

Uribe, E. H. and Ayala, L. E. V. (2007). Del manifiesto ágil sus valores y principios. *Scientia et Technica*, 1(34).

Velarde, O., Murillo, M., Gómez, L., and Felícita, C. (2006). *Introducción a la programación orientada a objetos*. Pearson Education, 1 edition.