PROTOTIPO WEB PARA LA GESTIÓN DE MÉTRICAS COMO SOPORTE A LA METODOLOGÍA DE DESARROLLO TSP (TEAM SOFTWARE PROCESS ®)

Presentado por

JUAN CAMILO CAMACHO BELTRÁN



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2017

PROTOTIPO WEB PARA LA GESTIÓN DE MÉTRICAS COMO SOPORTE A LA METODOLOGÍA DE DESARROLLO TSP (TEAM SOFTWARE PROCESS ®)

Presentado por

JUAN CAMILO CAMACHO BELTRÁN

CÓDIGO 20111020012

PROYECTO DE GRADO

DIRECTORA

MSc. ALBA CONSUELO NIETO LEMUS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2017

AGRADECIMIENTOS

Doy gracias a Dios por guiar mi vida, tanto personal como profesional, permitiéndome cumplir satisfactoriamente mis sueños, brindarme salud y bienestar, y proveerme de las capacidades suficientes para culminar exitosamente la carrera de pregrado actualmente en curso.

Agradezco a la Universidad Distrital Francisco José de Caldas por haberme permitido ser parte integral de la institución y darme la oportunidad de formarme a fin de adquirir el título profesional.

A mi directora MSc. Alba Consuelo Nieto Lemus, le estoy enormemente agradecido por aceptarme para la ejecución del presente proyecto de grado, guiándome incondicionalmente, a través de su conocimiento y experiencia, en este proceso, en el que además, ha puesto su esfuerzo, dedicación y paciencia, a fin de poder culminar mi carrera. Ha sido realmente enriquecedor contar con su guía y ayuda.

Así mismo, agradezco a mi jurado Fernando Martínez y Jhon Herrera Cubides, quiénes mediante su minuciosa labor, me han brindado herramientas para enriquecer el proyecto, logrando así mantener el enfoque. Su conocimiento ha sido de gran ayuda para valorar alternativas de gran significancia en el proyecto.

También quiero agradecer a todas las personas que estuvieron guiándome y apoyándome en distintos aspectos durante el tiempo de desarrollo del proyecto. A mi familia y mis seres amados por su comprensión y a los ingenieros Bauke Scholtz y alias “Kukeltje” por resolver la mayor parte de mis dudas asociadas a la implementación.

TABLA DE CONTENIDO

1. INTRODUCCIÓN 11
2. PLANTEAMIENTO DEL PROBLEMA 13
   2.1 ANÁLISIS DEL PROBLEMA 13
   2.2 DESCRIPCIÓN DEL PROBLEMA 13
   2.3 FORMULACIÓN DE LA PREGUNTA CENTRAL DEL TRABAJO 14
3. OBJETIVOS 15
   3.1 OBJETIVO GENERAL 15
   3.2 OBJETIVOS ESPECÍFICOS 15
4. JUSTIFICACIÓN 16
5. DELIMITACIÓN 18
6. MARCO REFERENCIAL 19
   6.1 CALIDAD EN EL DESARROLLO DE SOFTWARE 19
      6.1.1 La calidad orienta el adecuado desarrollo de software 19
      6.1.2 Logro de la calidad en el desarrollo de software 21
   6.2 GESTIÓN DE MÉTRICAS 22
      6.2.1 Importancia de las métricas en el desarrollo de software 22
      6.2.2 Conceptos de métricas 23
      6.2.3 Taxonomía de métricas 25
      6.2.4 Métricas clásicas de software [36] 26
   6.3 METODOLOGÍAS Y MODELOS DE CALIDAD DE SOFTWARE 29
      6.3.1 Metodologías de proceso de calidad de software 30
      6.3.2 Modelos de proceso de calidad de software 35
      6.3.3 Estado del arte de la metodología TSP 38
      6.3.4 La metodología TSP y la calidad de software 39
      6.3.5 Metodologías híbridas TSP 43
   6.4 CALIDAD DE SOFTWARE EN COLOMBIA 51
      6.4.1 Gestión de la calidad de software en Colombia 51
      6.4.2 Tendencia al clúster de software en Colombia 54
      6.4.3 Impulso a la certificación de calidad en Colombia 55

6.4.4 Impulso TSP en Colombia 56
7. ESTADO DEL ARTE 58
7.1 PROBLEMAS DE TSP FRENTE A LA GESTIÓN DE MÉTRICAS 58
7.2 ACTUALES HERRAMIENTAS DE SOPORTE PARA TSP 59
7.2.1 Team Dashboard 59
7.2.2 tVista 60
7.2.3 Point 60
7.2.4 Libro de trabajo de TSP 60
7.2.5 Enfoque de proceso guiado 61
7.3 SOFTWARE ORIENTADO A MÉTRICAS DE PROCESO 61
7.4 PROPUESTA 62
7.4.1 Aportes significativos 62
7.4.2 Valor agregado 63
8. METODOLOGÍA 64
8.1 ESTRUCTURA DEL EQUIPO SCRUM 64
8.2 ACTIVIDADES Y OBJETIVOS SPRINT 64
9. DESARROLLO DEL PROYECTO 66
9.1 HISTORIAS DE USUARIO 66
9.1.1 Características de agrupación 66
9.1.2 Pila de producto 67
9.1.3 Planeación de Sprints 69
9.1.4 Documentación de historias de usuario 70
9.1.5 Dependencias entre historias de usuario 76
9.2 GESTIÓN DE DATOS TSP 77
9.2.1 Mediciones, indicadores y métricas 77
9.2.2 Número de semanas de proyecto 79
9.3 EVALUACIÓN DEL DISEÑO 80
9.3.1 Prototipos del prototipo 80
9.3.2 Prototipo final 81
9.4 MODELADO DE REQUERIMIENTOS 83
9.4.1 Modelado UML ágil 83

9.4.2 Modelo de formatos 84
9.4.3 Diagrama de paquetes 85
9.4.4 Diagrama de clases 87
9.4.5 Diagramas de secuencia 93
9.4.6 Diagrama de despliegue 95
9.4.7 Diagrama de estado 96
9.5 MODELOS DE PERSISTENCIA 98
9.5.1 Modelo relacional 98
9.5.2 Modelo no relacional 103
9.6 HERRAMIENTAS TECNOLÓGICAS PARA EL DESARROLLO DEL PROYECTO 104
9.6.1 Ambiente de desarrollo 104
9.6.2 Software utilizado en el desarrollo 104
9.7 PRODUCTO RESULTANTE 111
9.7.1 COMPONENTES A DESTACAR 112
9.7.2 HERRAMIENTAS DE VALOR AGREGADO 116
9.7.3 Cartilla TSP 116
9.7.4 Comparador de código 116
9.7.5 Exportador de datos 117
9.8 PRUEBAS DE PROTOTIPO 117
9.8.1 Hardware utilizado 118
9.8.2 Escenarios 119
9.8.3 Cuantificación de las pruebas 127
9.8.4 Resultados 128
10. CONCLUSIONES 131
11. TRABAJOS A FUTURO 133
12. GLOSARIO 134
13. BIBLIOGRAFÍA 135

ÍNDICE DE TABLAS

Tabla 1. Ejemplos de entidades de tipo recurso, proceso o producto, descritos a través de atributos internos y externos. Fuente [32]..... 23
Tabla 2. Fases en las que los formularios de TSP deben ser completados. Fuente [8]... 34
Tabla 3. Modelo Continuo y Modelo Escalonado de CMMI. Fuente [44]..... 36
Tabla 4. Factores de calidad utilizados respectivamente en cuatro enfoques de desarrollo de software: orientado a objetos, orientado a componentes, orientado a aspectos y orientado a metodologías ágiles. Fuente [57]...... 43
Tabla 5. Características de los proyectos de aprendizaje en la metodología TSP. Fuente autor..... 44
Tabla 6. Formatos TSP utilizados en los diferentes proyectos desarrollados por los estudiantes de las respectivas fuentes bibliográficas. Fuente autor..... 45
Tabla 7. Actividades específicas de LTSP. Fuente [16]. 49
Tabla 8. Número de empresas que clasifican cada etapa de la Gestión de Proyectos por nivel de importancia. Fuente [65]. 53
Tabla 9. Comparación de las actuales herramientas de desarrollo de software. Fuente [14]..... 63
Tabla 10. Asignación de roles SCRUM para el proyecto. Fuente autor..... 64
Tabla 11. Actividades genéricas de sprint para el proyecto. Fuente autor..... 64
Tabla 12. Descripción de los sprints estimados del proyecto. Fuente autor..... 65
Tabla 13. Características de agrupación de historias de usuario. Fuente autor..... 67
Tabla 14. Historias de usuario con su respectiva descripción Fuente autor 68
Tabla 15. Historia de usuario de arquitectura de software. Fuente autor..... 70
Tabla 16. Historia de usuario de base de datos. Fuente autor..... 70
Tabla 17. Historia de usuario sobre la refinación de las historias. Fuente autor 71
Tabla 18. Historia de usuario de la literatura TSP. Fuente autor 71
Tabla 19. Historia de usuario de creación de proyecto. Fuente autor..... 71
Tabla 20. Historia de usuario de gestión de lanzamiento. Fuente autor..... 71
Tabla 21. Historia de usuario de gestión de estrategia. Fuente autor 72
Tabla 22. Historia de usuario de gestión de fases. Fuente autor 72
Tabla 23. Historia de usuario de control de riesgos. Fuente autor..... 72
Tabla 24. Historia de usuario de métricas de proyecto. Fuente autor..... 73
Tabla 25. Historia de usuario de métricas personales. Fuente autor 73
Tabla 26. Historia de usuario de registro de tiempo. Fuente autor 73
Tabla 27. Historia de usuario de gestión de tareas. Fuente autor 73
Tabla 28. Historia de usuario de rastreo del progreso. Fuente autor..... 74
Tabla 29. Historia de usuario de progreso en calendario. Fuente autor..... 74
Tabla 30. Historia de usuario de formatos complementarios. Fuente autor..... 74
Tabla 31. Historia de usuario de rastreo de defectos. Fuente autor..... 75
Tabla 32. Historia de usuario de control de cambios. Fuente autor..... 75

Tabla 33. Historia de usuario de resumen del proyecto. Fuente autor 75
Tabla 34. Historia de usuario de documentación del proyecto. Fuente autor 75
Tabla 35. Historia de usuario de pruebas. Fuente autor..... 76
Tabla 36. Historia de usuario de documentación de software. Fuente autor..... 76
*Tabla 37. Matriz de trazabilidad para las historias de usuario funcionales. Fuente autor*77
Tabla 38. Descripción de controladores del prototipo. Fuente autor 87
*Tabla 39. Clasificación de formatos de acuerdo a tres criterios principales. Fuente autor*90
Tabla 40. Descripción de tablas de modelo relacional. Fuente autor 98
Tabla 41. Privilegios de los usuarios sobre las talas del modelo relacional 100
Tabla 42. Atributos de tabla Usuario. Fuente autor..... 100
Tabla 43. Atributos de tabla Rol. Fuente autor 101
Tabla 44. Atributos de tabla Proyecto. Fuente autor 101
Tabla 45. Atributos de tabla Py_Cr. Fuente autor 101
Tabla 46. Atributos de tabla Criterio. Fuente autor..... 101
Tabla 47. Atributos de tabla OpcionTSP. Fuente autor 101
Tabla 48. Atributos de tabla Ciclo. Fuente autor..... 102
Tabla 49. Atributos de tabla Fase. Fuente autor..... 102
Tabla 50. Atributos de tabla Rl_Cl. Fuente autor 102
Tabla 51. Atributos de tabla Parte. Fuente autor..... 102
Tabla 52. Atributos de tabla Meta. Fuente autor 102
Tabla 53. Atributos de tabla Documento. Fuente autor 103
Tabla 54. Ordenador sobre el que se gestionaron los datos de cada rol. Fuente autor 126
Tabla 55. Pruebas funcionales y no funcionales sobre el prototipo. Fuente autor..... 127
Tabla 56. Porcentajes de acierto a las pruebas sobre el prototipo. Fuente autor 129
Tabla 57. Cantidad de artefactos desarrollados 130

ÍNDICE DE FIGURAS

Ilustración 1. Flujo de procesos en PSP. Fuente [38]..... 30
Ilustración 2. Estructura y flujo de TSP. Fuente [7]. 33
Ilustración 3. Entradas Six Sigma. a. Entradas PSP. b. Entradas TSP. Fuente [61]. 50
*Ilustración 4. Planes iniciales de entrenamiento en el proyecto FEDESOFT. Fuente [84]*57
Ilustración 5. Porcentaje de participación de algunos departamentos de Colombia en el proyecto FEDESOFT. Fuente [84]..... 57
Ilustración 6. Interfaz gráfica inicial del software Process Dashboard. Fuente Aplicación original...... 60
Ilustración 7. Vista inicial del libro de trabajo TSP. Fuente archivo SEI [8]..... 61
Ilustración 8. Características del proyecto clasificadas, dentro de la herramienta IceScrum, por color. Imagen extraída de la herramienta IceScrum...... 67
Ilustración 9. Historias de usuario en su respectivo sprint. Imagen capturada de la herramienta IceScrum. 69
Ilustración 10.Algoritmo para calcular el número de semana de TSP. Fuente autor..... 79
Ilustración 11. Interfaz piloto para el proyecto. Fuente autor 81
Ilustración 12. Interfaz definitiva para el proyecto. Fuente autor 82
Ilustración 13. Colores asignados para cada fase TSP en la herramienta. Fuente autor 83
Ilustración 14. Diagrama de contexto. Fuente autor...... 84
Ilustración 15. Grafo de inyección de información. Realizado con Pencil 85
Ilustración 16. Diagrama de paquetes de todo el sistema. Realizado con la herramienta Enterprise Architect 86
Ilustración 17. Diagrama de clases de Beans. Capa Web. Realizado con Enterprise Architect 88
Ilustración 18. Diagrama de clases de gestión de formatos. Realizado con Enterprise Architect 89
Ilustración 19. Diagrama de clases de gestión de atributos. Realizado con Enterprise Architect 90
Ilustración 20. Diagrama de clases para componente de revisión de código. Realizado con Enterprise Architect 91
Ilustración 21. Diagrama de clases para la gestión cualitativa. Realizado con Enterprise Architect 91
Ilustración 22.Diagrama de clases de gestión de persistencia. Realizado con Enterprise Architect 92
Ilustración 23. Diagrama de clases para la gestión de persistencia. Realizado con Enterprise Architect 93
Ilustración 24. Diagrama de secuencia para la carga de métricas, indicadores y mediciones de un proyecto de desarrollo de software. Creado con la herramienta Enterprise Architect 94

Ilustración 25. Diagrama de despliegue. Realizado con Enterprise Architect..... 95
Ilustración 26. Diagrama de estado. Realizado con Enterprise Architect 97
*Ilustración 27. Modelo de base de datos relacional. Realizado con Enterprise Architect*99
Ilustración 28. Arquitectura JSF. Fuente [94] 105
Ilustración 29. Logo de la herramienta. Fuente propia 111
Ilustración 30. Formato TASK en herramienta prototipo. Fuente propia..... 113
Ilustración 31. Registro de tiempo en TSPSupport. Fuente propia 113
Ilustración 32. Finalización de tarea en TSPSupport. Fuente propia..... 114
Ilustración 33. Formato SUMP en TSPSupport. Fuente propia..... 115
Ilustración 34. Directorio del proyecto en TSPSupport. Fuente propia...... 116
Ilustración 35. Interfaz de comparación de código. Fuente propia 117
Ilustración 36. Servicio de exportación de datos para el formato LOGT. 117
Ilustración 37. Servidor de pruebas y desarrollo. 118
Ilustración 38. Ordenador portátil cliente..... 118
Ilustración 39. Ordenador de escritorio de cliente...... 119
Ilustración 40. Router para conexión LAN de pruebas. 119
Ilustración 41. Comparación de metas documentadas e ingresadas al software. Fuente autor..... 120
Ilustración 42. Comparación de formato STRAT documentado e ingresado al software. Fuente autor..... 121
Ilustración 43. Comparación de formato ITL documentado e ingresado al software. Fuente autor..... 122
Ilustración 44. Comparación de formato TASK documentado e ingresado al software. Fuente autor..... 122
Ilustración 45. Comparación de formato SUMP documentado e ingresado al software. Fuente autor..... 123
Ilustración 46. Comparación de formato SUMQ documentado e ingresado al software. Fuente autor..... 124
Ilustración 47. Comparación de formato LOGT documentado e ingresado al software. Fuente autor..... 125
Ilustración 48. Diagrama representativo de conexión LAN. Fuente autor. 125
Ilustración 49.Montaje real de prueba en LAN. Fuente autor. 126
Ilustración 50. Flujo de trabajo del prototipo resultante. Fuente autor 130

1. INTRODUCCIÓN

El software es de los productos de ingeniería que más rápido ha evolucionado, pasando del empirismo hasta el desarrollo basado en principios de calidad. Esto implica que debe ser construido aplicando estándares mundiales, modelos arquitectónicos, estructurales y funcionales, métricas, capacitación del recurso humano y otros principios y técnicas de la ingeniería de software [1]. Dichos cambios se deben a que los clientes son cada vez más exigentes con el cumplimiento de los requisitos y los estándares de calidad en los productos de software [2] [3].

En la actualidad, la industria del software necesita ingenieros que sepan producir productos de calidad en el tiempo establecido [4], y dado que la competencia en el desarrollo de software es cada día más fuerte [3], es fundamental que las empresas se preocupen por ofrecer un mejor producto. Sin embargo, la competencia no debe ser la única motivación por la que se deba buscar calidad en el software: es fundamental desarrollar conciencia, disciplina y responsabilidad acerca de las consecuencias que puede ocasionar un defecto en el producto resultante. Son muchos los casos de proyectos de desarrollo de software no exitosos que acarrean, no solamente pérdidas económicas, sino también costos de oportunidad e incluso problemas legales. Por ejemplo, recientemente en Colombia, el Sistema Integrado de Movilidad, que tiene por propósito integrar todos los trámites relacionados con los vehículos automotores, evidenció falta de información hacia los usuarios (digi-turnos), irregularidades en los tiempos de los trámites y sobre todo, la poca confiabilidad por el acceso total a los datos de los usuarios [5], ocasionando serios inconvenientes en la prestación del servicio.

Dadas estas dificultades en la gestión de proyectos de software, que no sólo atañen a Colombia sino en general a la industria del software, y teniendo en claro que, como afirma Humphrey [6], para obtener un producto de calidad se requiere un proceso de calidad que debe ser planificado y continuamente gestionado, surge la imperiosa necesidad de adoptar una metodología que permita formar disciplina tanto a nivel personal como de equipo, definir métricas de proceso de desarrollo, y además, que sea apropiada por los ingenieros, a tal grado que no se considere como un esfuerzo adicional o un consumo innecesario de tiempo. Si bien, dichas metodologías existen [6] [7]; lo que no se tiene son herramientas de soporte adecuadas que faciliten el proceso de recolección y gestión de métricas.

En este proyecto se propone el desarrollo de una herramienta de software que apoye el proceso de gestión de métricas (definición, planeación, recolección y consolidación) soportado en la metodología TSP de Humphrey, ya que estudios realizados por el Instituto de Ingeniería de Software SEI [8], muestran que adoptar dicha metodología permite mejorar notablemente, tanto la calidad del proceso como del producto del software, logrando incluso alcanzar niveles de calidad 17 veces superiores a compañías que cuentan con un nivel 5 de madurez en CMMI [9]. Esto último es un claro indicador de las bondades de la metodología TSP, pero para que las empresas la puedan adoptar con éxito, deben contar con una herramienta que facilite su implementación.

2. PLANTEAMIENTO DEL PROBLEMA

2.1 ANÁLISIS DEL PROBLEMA

El producto software, al igual que otros productos del mercado, tiene un costo asociado a los recursos necesarios para su construcción, sin embargo, la estimación de dicho costo es una tarea bastante compleja. Debido a que muchas veces el software se desarrolla a la medida del problema que trata de resolver, y el grado de dificultad recae sobre el nivel de dominio, tanto de tecnología como del contexto del problema, es imposible hacer comparaciones precisas [10] [11], y mucho menos, establecer un costo promedio global por cada producto software a construir; el software depende de qué tan capacitado es el recurso humano para ofrecer una solución tecnológica a las necesidades de sus clientes.

Las correctas estimaciones de tiempo, esfuerzo, costo, entre otros, para la construcción de un producto software de calidad, se apoyan en modelos y metodologías de calidad; sin embargo, modelos como ISO 9001 [39] son altamente criticados por su estructura genérica, por olvidarse del mejoramiento de procesos y limitarse a adjuntar documentos para validar listas de chequeo [12]. Por el contrario, las metodologías PSP [38] y TSP [7], teniendo por propósito adquirir la disciplina para mejorar el proceso de desarrollo de software, y por lo tanto la calidad, promueven el registro de métricas y la retroalimentación a partir de su análisis. Sin embargo el trabajo de registrar (manualmente) toda actividad realizada, producto construido o cambio requerido resulta dispendioso.

2.2 DESCRIPCIÓN DEL PROBLEMA

Aunque TSP fomenta una cultura disciplinaria en el desarrollo de software, los procesos directamente asociados a la metodología requieren demasiado esfuerzo que se percibe como si se tratara de un proyecto paralelo [13]. Ninguno de los proyectos desarrollados con la metodología TSP de Humphrey, descritos a lo largo del presente documento, afirma haber completado satisfactoriamente todos los formatos y tareas requeridas [57]. Puesto que los formatos deben ser diligenciados manualmente, aumenta en forma considerable la probabilidad de fallo en las mediciones por datos no ingresados o ingresados incorrectamente debido a la fuerte dependencia de los datos diligenciados en los distintos formatos [58].

TSP depende en mayor medida del registro de métricas de proceso y, éstas a su vez dependen principalmente del tiempo y esfuerzo requerido sobre las tareas de cada integrante del equipo, la velocidad de detección de riesgos, entre otros. Al ser el tiempo un factor tan importante, los integrantes del equipo deben estar constantemente detectando la hora de inicio y hora de finalización de cada una de sus tareas e interrupciones dentro de éstas, lo que provoca, además de estrés, registros cuantificables inexactos y retrasos al llevar a cabo cálculos matemáticos y su diligenciamiento.

Dada la alta densidad de la metodología TSP [13], las empresas de desarrollo de software de Colombia, en su mayoría PyMES, prefieren abstenerse de aplicarla ya que involucra en sus procesos mayores esfuerzos y costos, lo que desaprovecha la posibilidad de mejorar la calidad del desarrollo de software y por ende de ser reconocidas nacional e internacionalmente. Frente a esta problemática que no sólo atañe a Colombia, surgen herramientas de soporte para la metodología TSP, con el fin de facilitar su incorporación en los procesos de desarrollo de software.

Existen herramientas de soporte a TSP, sin embargo, éstas se enfocan en el registro de tareas y tiempos nativos de la metodología PSP[14], sin hacer un aporte significativo al registro, recolección y análisis de métricas asociadas con el trabajo en equipo. Estas aplicaciones (a excepción de tVista) se despliegan en escritorio, lo que impide trabajar en múltiples plataformas o dar inmediatez en el acceso, dejando de lado la orientación web, que ha tomado auge durante los últimos años. Cabe destacar que la herramienta de soporte para la metodología TSP del SEI [8], desplegada en hojas de cálculo, no brinda una adecuada visualización de la información recolectada [14] ni mucho menos la experiencia del trabajo de equipo, ya que cada integrante cuenta hojas de cálculo en forma independiente.

2.3 FORMULACIÓN DE LA PREGUNTA CENTRAL DEL TRABAJO

¿Cómo desarrollar una herramienta de apoyo para la metodología TSP que se pueda integrar fácilmente con el proceso de desarrollo de software para la gestión de métricas de manera que sea apropiada por los integrantes de un equipo de desarrollo, sin considerarla una carga adicional en sus tareas?

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar una herramienta prototipo orientada a la web como soporte a la metodología TSP propuesta por Humphrey [7], que permita apoyar la gestión y análisis de métricas asociadas al proceso de desarrollo de software, mediante el registro, validación y control automático de los datos.

3.2 OBJETIVOS ESPECÍFICOS

- Proporcionar mecanismos para diligenciar y completar automáticamente formatos de la metodología TSP, teniendo en cuenta principalmente la usabilidad y amigabilidad con los usuarios.
- Proveer acceso remoto y centralizado a la información correspondiente a los proyectos de desarrollo de software, separando contenidos por proyecto y suministrando los directorios base de la metodología TSP (ciclos, fases y roles).
- Construir un modelo de datos que soporte la relación de todos los elementos de un proyecto de desarrollo de software en torno a la metodología TSP, facilitando el acceso, registro y consolidación de la información para su respectiva retroalimentación.
- Determinar la tecnología a utilizar para cada una de las capas del modelo de arquitectura (MVC), teniendo en cuenta características como presupuesto, dificultad de despliegue, soporte, desempeño en entorno web, entre otros.
- Ejecutar pruebas funcionales y no funcionales sobre la herramienta, hacer los ajustes necesarios con base a los resultados y desplegar el producto en un servidor de uso libre para actividades de formación y producción de la comunidad.

4. JUSTIFICACIÓN

Puesto que la correcta implantación de un modelo de calidad ayuda a que la empresas puedan disminuir sus costos de desarrollo, orientar las ganancias y administrar mejor los recursos, las empresas del sector TI tienen como finalidad certificarse en calidad por razones asociadas a la exportación de software [10] y resultar, según Peláez [15], más competitivas, generar mayores utilidades y ser más atractivas para los clientes.

Para el logro de estos objetivos, las empresas de desarrollo de software incorporan una metodología de calidad, entre las cuales se destaca TSP. Esta metodología se caracteriza por formar ingenieros mejor calificados para gestionar proyectos exitosamente mediante el trabajo en equipo [13]; reduce el costo y tiempo de producción, incrementa la productividad [16] y se articula perfectamente a la gestión de métricas, como consecuencia del constante y detallado registro de actividades, que tanto la identifica, a lo largo de proyectos de desarrollo de software.

Pese a todas las ventajas de la metodología, la implementación de TSP, en definitiva provoca que los costos aumenten, haciendo que los ingenieros dediquen más esfuerzo y tiempo en las labores de un proyecto de desarrollo de software. Como consecuencia de ello, países como Colombia, principalmente constituidos por empresas PyMES (en el sector TI), pierden interés por adoptarla, fomentando así, el desarrollo de herramientas de soporte para la metodología, con la intención de hacerla más fácil de implementar.

Las actuales herramientas de soporte para la metodología TSP no satisfacen las necesidades de los usuarios y no proporcionan mecanismos de retroalimentación; haciendo inútiles las mediciones tomadas a lo largo de los proyectos que se desarrollan. Los integrantes de un equipo de desarrollo, buscan una herramienta que los guíe a lo largo del proceso, que sea amigable con el usuario, registre información automáticamente y brinde la documentación necesaria en todo momento. Team Dashboard [14], por ejemplo, es un software de escritorio que depende de la versión Java instalada en el equipo local, lo que ocasiona que pueda o no ejecutar muchas de sus funcionalidades.

Desarrollar una herramienta de soporte para TSP orientada a la web, además de resolver problemas de compatibilidad de plataforma, facilidad de acceso y control de versiones, hace más

flexible a la metodología porque la documentación de cada proyecto se almacena en forma centralizada y todos los integrantes del equipo pueden acceder a ella. Con la presente propuesta, el diligenciamiento de formatos se hará en forma manual, automática y semiautomática, indicando al usuario si la información ingresada se ajusta a las especificaciones de la metodología propuesta por Humphrey y, por ende, guiándolo en la ejecución de las tareas individuales y de equipo.

Con la propuesta flexible de TSP, la herramienta no sólo facilitará el desarrollo de la metodología, sino que será de fundamental ayuda para recolectar métricas a lo largo de la ejecución de diferentes proyectos, generando costos mínimos adicionales y permitiendo obtener mayores beneficios mediante la retroalimentación personal y de equipo. Con esta herramienta software de tipo colaborativo, nuevas empresas de desarrollo podrán apropiar rápidamente procesos de calidad, de la metodología TSP, sin mayor esfuerzo, adquiriendo una disciplina suficiente para obtener rápidamente mejoras en sus procesos, y mayor competitividad en el mercado.

5. DELIMITACIÓN

5.1 ALCANCE

En este proyecto se pretende desarrollar una herramienta prototipo orientada a la web, con el fin de apoyar la gestión de las métricas asociadas a los roles en un equipo de desarrollo, de acuerdo con la metodología TSP. La aplicación debe contar con la capacidad de gestionar métricas, planear tiempos, controlar defectos y gestionar la información para todos los miembros, facilitando el proceso de registro y consolidación sin representar una carga adicional a las tareas propias de un proyecto de software.

5.2 LIMITACIONES

- Se pretende desarrollar un prototipo de software más que un producto de software terminado, dado el tiempo y los recursos con los que se cuentan en un proyecto de pregrado. Sin embargo, el diseño que se proponga, deberá permitir un fácil escalamiento.
- El prototipo a desarrollar será una herramienta de apoyo a la metodología en la tarea de registro y consolidación de métricas para que el gestor del proyecto ejecute los análisis que considere pertinentes.
- El desarrollo del prototipo se hará con herramientas de software libre.
- Se desarrollará una aplicación web, no una herramienta móvil.
- La seguridad de los datos dependerá de la tecnología sobre la que se despliegue la herramienta. No se implementarán protocolos adicionales de protección de datos.
- Las pruebas se llevarán a cabo en máximo dos computadores portátiles, con uno de ellos como servidor. La conexión se llevará a cabo en forma local vía WiFi.
- El control de congestión se llevará cabo de acuerdo a la configuración por defecto de la tecnología sobre la cual se despliegue la aplicación.

6. MARCO REFERENCIAL

6.1 CALIDAD EN EL DESARROLLO DE SOFTWARE

La adecuada gestión de métricas y la correcta implementación de la metodología TSP, se fundamentan en la comprensión de la calidad, puesto que ésta, determina las cualidades que debe satisfacer el software, y su proceso de desarrollo. A continuación se describe la importancia de la calidad y su propósito de mejorar la de vida de quienes hacen uso de los productos software, que tratan de resolver problemas cada vez más complejos y catastróficos.

6.1.1 La calidad orienta el adecuado desarrollo de software

No es fácil encontrar una definición única acerca de lo que es Calidad de Software. De acuerdo con Estayno Marcelo [17] y Lidia García [18], la calidad de software es una compleja combinación de factores que varían entre diferentes aplicaciones, sin mencionar que su definitivo y útil significado es difícil de precisar.

Para ISO 25000 [19], la calidad es la aptitud de un producto o servicio para satisfacer la necesidad del usuario; es sinónimo de eficiencia, corrección, mantenibilidad, usabilidad, flexibilidad, confiabilidad, portabilidad, seguridad e integridad [18]. Según UNE-EN ISO 8402 [17], la calidad es el conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer necesidades explícitas o implícitas. Por su parte en [10] y [20], se refieren a la calidad como aquello que no tiene defectos, con acciones correctas desde el principio y con la capacidad de incorporar la satisfacción de los usuarios.

Concretando la definición de calidad UNE-EN ISO 8402 al contexto del software, la IEEE Std 610 [17] [15] señala que la calidad de software es el grado con el que un sistema comparte, procesa o cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario, a lo que [21] alerta de la importancia de satisfacer también otros aspectos como seguridad, rendimiento, accesibilidad, mantenibilidad (requisitos no funcionales), entre otros, que no siempre están definidos formalmente y que contribuyen propiedades inherentes del producto, que de una otra manera, son probados y muy valorados por los clientes y usuarios finales. Adicional a todo esto, es importante reconocer que la calidad de software es siempre mejor obtenida si se controla durante todas las etapas del ciclo de vida del software (o en todos sus estados

de evolución [10]) y no al final del producto software, esto con el propósito de disminuir al máximo los costos generados en un proyecto de desarrollo de software [18].

Scalone [10] y Rojas [1] citan una definición puntual de calidad de software: "Concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente". Cuando se habla de calidad de software se habla igualmente de eficiencia, rendimiento y funcionamiento del equipo, de cumplir y de ser posible, superar las expectativas de los usuarios, siguiendo los procedimientos acordados, resolviendo problemas y efectuando los estándares solicitados [22]. A su vez, debe existir consistencia con el diseño independiente de la plataforma y optimización de recursos [1] [17] (Sommerville, 2005).

Existen dos visiones profesionales acerca de la calidad de desarrollo de software propuestas por Pressman y por Vega, Rivera & García [2]. De acuerdo a Pressman, la calidad del desarrollo de software es la concordancia del software producido con los requerimientos explícitamente establecidos con los estándares de desarrollo preestablecidos y con los requerimientos implícitos no establecidos formalmente [17]. En cuanto a Vega, Rivera & García, se trata de la totalidad de características de un producto de software que tienen como habilidad satisfacer las necesidades explícitas o implícitas [1].

En síntesis, con base en lo ya mencionado y otras características tratadas en [15], la calidad es un conjunto de propiedades y características de un producto o servicio, creado mediante procedimientos claramente definidos, reglamentados y bien documentados, que no presenta defectos, satisface los requisitos explícitos e implícitos para los que fue creado y, por ende, cubre las necesidades y expectativas de los clientes, todo esto dentro del factor económico mínimo requerido para su construcción. En cuanto a la calidad del software en específico, se puede definir como el cumplimiento de estándares y de procedimientos continuamente controlados, que permiten dar cumplimiento, tanto de las expectativas los clientes, como de las propiedades inherentes del producto, garantizando con ello, la concordancia entre los requerimientos explícitos e implícitos del software producido.

6.1.2 Logro de la calidad en el desarrollo de software

Considerando la importancia de la calidad en el desarrollo de software, las empresas dentro de este nicho de mercado, no pueden dejar de lado su adquisición. De acuerdo a [23], el logro de la calidad sólo es posible si existe una cultura de mejora continua, la cual consiste en buscar mejores métodos de trabajo y revisar continuamente los procesos organizativos.

Para que las empresas puedan incorporar calidad en sus procesos, deben cumplir con los principios básicos de la calidad, que igualmente aplican para la calidad en el desarrollo de software; los principales son [23]:

- Enfoque al cliente: La prioridad es dejar al cliente satisfecho, dando a conocer sus expectativas y necesidades.
- Liderazgo: Todas las organizaciones necesitan líderes que los guíen y mantengan un ambiente interno para el logro de objetivos.
- Participación del personal: Necesaria participación de todo el personal para adquirir mejores ideas.
- Enfoque basado en procesos: Actividades y recursos que deben ser gestionados con base en procesos estratégicos, operativos y de soporte.
- Enfoque basado en hechos para la toma de decisiones: Al momento de tomar una decisión, ésta se debe basar en hechos, datos e información que se posea y garantice una baja posibilidad de errores o la no existencia de ellos.

No es suficiente tener en claro los principios de la calidad para adquirirla; la calidad de software está directamente vinculada al ciclo de vida del desarrollo y al conjunto de rígidos patrones usados a lo largo de éste. La calidad de software depende de la calidad con la cual se lleve a cabo todo el proceso (análisis, diseño, implementación, pruebas e implantación) y cada subproceso, fase o etapa del proyecto, siendo indispensable contar con planeamiento, estándares, entrenamiento, experiencia, documentación y soporte [24]. Por esta razón, los expertos en calidad sugieren adoptar los siguientes pasos para lograr la calidad en el software [25]:

1. Establecer un sistema de evaluación y medición para determinar en qué medida se está actuando correctamente. Elegir proyectos terminados para evaluar con ellos, tiempos, esfuerzo, errores y correcciones a lo largo de sus desarrollos.

2. Documentar el proceso de desarrollo actual por muy caótico que sea.
3. Calcular el costo de corregir errores de software.
4. Hacer lo necesario para eliminar errores en requerimientos y diseño.
5. Entrevistar cuidadosamente a usuarios y directores, y verificar que se comprende lo que necesita.
6. Realizar pruebas con frecuencia. Comprobar módulos conforme son desarrollados.

Aunque se sepa qué pasos seguir para obtener calidad de software, no es sinónimo de comprender cómo deben llevarse a cabo. Para resolver esta necesidad, surgen las metodologías y modelos de gestión de la calidad de software. Éstas tienen el fin de adaptar las actividades empresariales a procesos de calidad y, mediante el uso de métricas, dar estimaciones objetivas de los atributos asociados al software a construir [26]. Estas metodologías y modelos se clasifican por proceso y por producto, sin embargo, es de especial interés la orientación de proceso porque se centra en los procedimientos llevados a cabo para obtener calidad en el software; entre estas metodologías destacan Six Sigma [60], Bootstrap [108], PSP [38] y TSP [7], y modelos como CMMI [43] e ITMark [44].

El logro de la calidad finalmente recae en los procesos de gestión a través de la planificación, lo que se soporta en el uso de métricas que permitan recolectar y analizar información cuantificable, para con ello, retroalimentar y garantizar la mejora continua [24]. Por esta razón, la gestión de métricas es fundamental en el desarrollo de software con calidad; ésta provee de herramientas, a las empresas y equipos de desarrollo de software, para focalizar los aspectos más relevantes que faciliten la toma de decisiones orientadas al mejoramiento.

6.2 GESTIÓN DE MÉTRICAS

6.2.1 Importancia de las métricas en el desarrollo de software

La medición es fundamental para cualquier disciplina de ingeniería, y específicamente, en la ingeniería de software, adquiere especial importancia en el intento de inculcar una mejora continua. Esto justifica que, el proceso de desarrollo de software, se soporte principalmente en métricas. Las métricas de software pretenden mejorar los procesos de desarrollo, y por ende, todos los aspectos de la gestión de los mismos, haciéndose útiles (junto a los modelos) para estimar y predecir costos, y medir la productividad y calidad del producto [27].

Mediante un registro constante y sistemático, una empresa de desarrollo de software podrá transformarse a través del tiempo, gracias a la información recolectada y continuamente refinada. Por este motivo, los equipos de desarrollo de software deben determinar qué medir y controlar dentro del proceso, seleccionando las métricas a usar y evitando en lo posible, hacer malas mediciones que conlleven a conclusiones erróneas y consecuentemente, a malas decisiones y malas acciones correctivas [28].

La métrica en el software, ayuda a mejorar tanto el proceso como el producto [29], así como al entendimiento del proceso tecnológico utilizado para el desarrollo [30]. Comprende un amplio rango de actividades entre las cuales se destacan el aseguramiento y control de calidad, modelos de evaluación, ejecución, y modelos de medida de productividad [29], dando una idea sobre qué acciones emprender [31].

6.2.2 Conceptos de métricas

6.2.2.1 Conceptos básicos

Las métricas ayudan a dar un mejor entendimiento del proceso de desarrollo de software al permitir comparar los diferentes productos y, ambientes de desarrollo [32]. Esto requiere entender los siguientes conceptos.

- Entidad. Objeto caracterizado mediante una medición de sus atributos. Puede ser física (tangible) o abstracta (intangible) [33]. Una entidad se diferencia de otra a partir de sus características [32].
- Atributo. Propiedad mensurable, física o abstracta, de una entidad, que se puede medir por medio de una métrica directa o indirecta. Puede ser interno (como el tamaño del código fuente) o externo (como el precio) [33]. Cuando se describen entidades mediante el uso de atributos, suelen utilizarse números o símbolos para dar juicio a partir de éstos; un ejemplo puede ser el tamaño de un programa o el tiempo utilizado en la fase de prueba [32].

Tabla 1. Ejemplos de entidades de tipo recurso, proceso o producto, descritos a través de atributos internos y externos. Fuente [32].

Entidades		***Atributos***	
		Internos	***Externos***
Productos	*Especificaciones*	*Tamaño, reutilización, modularidad, redundancia, funcionalidad y exactitud*	*Comprensibilidad y mantenibilidad*
	Diseños	*Tamaño, reutilización, modularidad, acoplamiento, cohesión y funcionalidad*	*Calidad, complejidad y mantenibilidad*

	Código	*Tamaño, reutilización, modularidad, acoplamiento, funcionalidad, complejidad algorítmica y flujo de control estructurado*	*Confiabilidad, utilización y mantenibilidad*
	Datos de prueba	*Tamaño y nivel de cobertura*	*Calidad*
Procesos	*Especificaciones de construcción*	*Tiempo, esfuerzo y número de requerimientos cambiados*	*Calidad, costo y estabilidad*
	Diseño detallado	*Tiempo, esfuerzo y número de errores encontrados.*	*Costo y efectividad*
	Pruebas	*Tiempo y esfuerzo*	*Costo y estabilidad*
Recursos	*Personal*	*Años y costo*	*Productividad, experiencia e inteligencia*
	Equipos	*Tamaño y nivel de comunicación*	*Productividad y calidad*
	Software	*Precio y tamaño*	*Utilización y confiabilidad*
	Hardware	*Precio, velocidad y tamaño de memoria*	*Confiabilidad*
	Oficinas	*Tamaño, temperatura y luz*	*Comodidad y calidad*

Para efectuar la medición, las entidades y sus respectivos atributos deben especificarse, a fin de precisar qué se quiere medir y bajo qué términos. En la tabla 1 se observan algunos ejemplos de entidades junto a sus atributos internos y externos.

6.2.2.2 Medida-medición, métrica e indicador

- Medida. Es el resultado de una cuantificación de datos, único y específico [34].
- Medición. Consiste en asignar números o símbolos a entidades reales, que en el contexto de la ingeniería de software, propone una indicación cuantitativa sobre la cantidad, densidad, capacidad o tamaño de algún atributo de un producto o proceso [35] [29], capturando datos que permiten obtener información para facilitar la toma de decisiones con un criterio fundado [31].
- Métrica. Combinación de múltiples medidas que a menudo establece un contexto para comprender las tendencias de los datos en el tiempo. La métrica representa una extrapolación al cálculo matemático sobre las mediciones resultantes de un atributo correspondiente a un sistema, componente o proceso [34]. Específicamente en el desarrollo de software, las métricas son recopiladas por los ingenieros para obtener indicadores [35]. La métrica debe validarse empíricamente en una amplia variedad de contextos antes de publicarse o aplicarse para la toma de decisiones, siendo fundamental, que ésta posea propiedades matemáticas deseables [35].
- Indicador. Es un predictor. Consiste en la representación de una medición o métrica de una manera sencilla o intuitiva para facilitar su interpretación contra una referencia o meta [34]. Con los indicadores, los ingenieros adquieren conocimiento acerca del proceso de desarrollo de software, y a partir de éste, ajustan el proceso, proyecto o producto, con el propósito de mejorar en todos los aspectos necesarios [35].

Para comprender los términos medida, métrica e indicador, se propone el siguiente ejemplo. Al evaluar la exactitud de la estimación del cronograma de un proyecto, dos medidas importantes podrían ser la duración del proyecto real y el estimado, siendo la métrica la precisión del horario de estimación (duración real del proyecto divido entre la duración estimada del proyecto) y el indicador podría ser una representación de la estimación, dada en porcentaje [34].

6.2.3 Taxonomía de métricas

6.2.3.1 Clasificación general de métricas

- Métricas de producto. Medidas de producto durante cualquier fase del desarrollo de software, desde los requisitos hasta la instalación. Pueden medir la complejidad del diseño, tamaño del producto final o el número de páginas del documento producido [27].
- Métricas de proceso. Medidas del proceso de desarrollo de software, que pueden utilizarse para mejorar un producto o servicio [34]. Como ejemplo son el tiempo total de desarrollo o el esfuerzo en horas/hombre [27].

  Las métricas de proceso pueden ser usadas para mejorar el desarrollo de software y su mantenimiento. Incluye medidas como la duración del proceso (o alguna de sus actividades), esfuerzo asociado con el proceso y el número de incidentes que ocurren en el proceso [32].

  Se pueden definir tres clases de métricas de proceso que permiten descubrir si los cambios en el proceso mejoran o no, éstas son [32]:

  1. El tiempo requerido para completar un proceso en particular. Es el tiempo total dedicado a al proceso.
  2. Los recursos requeridos para un proceso en particular. Puede ser el esfuerzo total en personas al día.
  3. El número de ocurrencias de un evento en particular. Un ejemplo puede ser el número de defectos descubiertos durante la fase de codificación.
- Métricas de proyecto. Permiten evaluar el estado del proyecto y dar seguimiento a los riesgos [30].

- Métricas de recurso. Describen las características del proyecto y su ejecución, ayudando a entender y controlar el proceso. Un ejemplo es el número de desarrolladores de software durante el ciclo de vida de éste [32].

6.2.3.2 Clasificación general de métricas en el desarrollo de software

- Directo. Hace alusión al costo y esfuerzo aplicados, velocidad de ejecución, líneas de código y espacio de memoria [30].
- Indirecto. Se refiere a la funcionalidad, calidad, eficiencia, fiabilidad, facilidad de mantenimiento, etc. [30].

El siguiente ejemplo ayuda a comprender fácilmente la diferencia entre las métricas directas e indirectas. Para la métrica directa puede considerarse la cantidad de enlaces web rotos internos medidos por la presencia de errores de tipo 404. La métrica indirecta, por el contrario, para el mismo caso, correspondería al porcentaje de enlaces rotos, calculado a partir de la división entre los enlaces rotos internos y externos sobre la cantidad total de enlaces web del sitio [33].

6.2.3.3 Clasificación de métricas según el criterio [29]

- De complejidad. Métricas que definen la medición de la complejidad: volumen, tamaño, anidaciones, y configuración.
- De calidad. Métricas que definen la calidad del software: exactitud, estructuración o modularidad, pruebas y mantenimiento.
- De competencia. Métricas que intentan valorar o medir las actividades de productividad de los programadores con respecto a su certeza, rapidez, eficiencia y competencia.
- De desempeño. Métricas que miden la conducta de módulos y sistemas de un software bajo la supervisión del sistema operativo o hardware.
- Estilizadas. Métricas de experimentación y de preferencia: estilo de código, convenciones, limitaciones, etc.

6.2.4 Métricas clásicas de software [36]

6.2.4.1 Métricas de tamaño

Todos los programas comparten un atributo independiente del lenguaje de programación o área de aplicación, sin embargo, todos tienen un tamaño. El tamaño es importante porque la

productividad normalmente se basa en éste, que además, es fácil de calcular una vez que el programa se ha puesto en ejecución.

Las mediciones de tamaño más importantes son:

- Líneas de código (LOC). El tamaño se determina a partir de las líneas de código del programa. Una línea de código, es toda línea de un programa que no es un comentario, o línea en blanco, independientemente del número de instrucciones o fragmento de instrucciones en ella, incluyendo líneas de encabezamiento, declaraciones, e instrucciones ejecutables y no ejecutables.

  El problema de esta medición, es que no todas las líneas son equivalentes en dificultad de codificación.
- Número de Tokens. Consiste en describir un programa como un conjunto de tokens (unidades clasificables como operandos u operadores). Los operadores son representados con cualquier símbolo o palabra reservada que especifica una acción, y los operandos son símbolos para representar datos.

  La mayoría de los signos de puntuación son consideradores operadores y, las variables, constantes y *labels* se consideran operandos. Símbolos matemáticos (*+,-,/, etc.*), nombres de comandos (*while, for, read)* y símbolos especiales (*(),{},[],etc.*) son igualmente operadores. El problema con esta métrica, es que las reglas de contabilización dependen del lenguaje y por ende, las ambigüedades ocurren con frecuencia.
- Puntos de función. Métrica basada en la idea de utilizar unidades más grandes para medir el software a partir de funciones. Una función se define como una colección de sentencias o instrucciones ejecutables que realizan cierta tarea, y que se desarrollan en conjunto con declaraciones de parámetros formales y variables locales manipuladas por dichas sentencias.

6.2.4.2 Métricas de estructura de datos

Se basa en el concepto de caja negra para así recolectar métricas a partir de los datos de entrada, los datos que se procesan y los datos que salen del sistema.

- Cantidad de datos. Se caracteriza por la identificación de VARS (variables) a partir de una lista de referencias cruzadas, excluyendo variables definidas pero nunca usadas. El

inconveniente con esta métrica, es que sólo refleja el número de variables únicas, sin indicar su grado de uso.

- Uso de datos dentro de un módulo.
  - ✓ Variables vivas. Se fundamenta en la hipótesis de que, a mayor número de ítems de datos que un programador debe mantener en su cabeza mientras programa, más difícil será la construcción de éste. Esta métrica mide la utilización de datos dentro de un módulo, considerando que una variable está viva desde la primera hasta la última referencia dentro de un procedimiento.
  - ✓ Spans (Separaciones o amplitudes) de variables. Consiste en la captura de la frecuencia en que una variable es usada dentro de un programa o procedimiento (una variable referenciada *n* veces, tiene *n-1* spans).

6.2.4.3 Métricas de estructura lógica

Se argumenta en los algoritmos que ejecutan los ordenadores. Además de las grandes cantidades de datos que almacenan los computadores, también estos dispositivos tienen la habilidad de probar datos y tomar distintos caminos dependiendo los resultados de las pruebas. Bajo este concepto surgen las siguientes métricas:

- Número de decisiones. Consiste en contar el número de instrucciones del tipo *if*, *while, case,* entre otros condicionales y controladores de ciclo. A mayor número de decisiones se considera que tiene mayor complejidad el programa.

  Esta métrica se divide en tres tipos de decisiones:
  - ✓ Ramificación hacia delante. Un test condicional seguido de una elección entre por lo menos dos posibilidades de acción.
  - ✓ Ramificación hacia atrás. *Loops* condicionales o incondicionales.
  - ✓ Ramificación horizontal. Típicamente una transferencia de control a una subrutina.
- Complejidad ciclomática. Métrica diseñada originalmente para determinar el número de caminos literalmente independientes a través de un programa.

6.2.4.4 Métricas de ciencia del software

Se basa en la contabilización de las cuatro métricas mencionadas anteriormente (cantidad de datos, uso de datos dentro de un módulo, número de decisiones y complejidad ciclomática). Consiste en un modelo de razonamiento con operadores y operandos que permiten resolver

preguntas relacionadas a la programación, esfuerzo de implementación y nivel de lenguaje utilizado. Esta métrica cuenta con pruebas de campo y, a pesar de la controversia detrás de sus fundamentos teóricos, remarca la necesidad de métricas y sus puntos potenciales.

6.2.4.5 Otras métricas

- Métricas de esfuerzo de desarrollo. Consiste en medir el esfuerzo a partir de las horas efectivas dedicadas a las tareas de desarrollo y número de personas por unidad de tiempo (mes o año) dedicadas al desarrollo del proyecto.
- Métricas de defectos. Considerando un defecto como la evidencia de la existencia de un error en el software que se produce, la métrica contabiliza el número de cambios tanto en diseño como en código asociados a la identificación y corrección de defectos.

6.3 METODOLOGÍAS Y MODELOS DE CALIDAD DE SOFTWARE

Con el proceso de globalización, las organizaciones se ven obligadas a competir a nivel mundial, y por ende, la calidad se convierte en un impactante punto diferenciador, que ayuda a aumentar la satisfacción de los clientes, a disminuir costos y optimizar recursos [24]. Como se mencionó en el apartado anterior (5.2), las métricas cumplen un rol fundamental en el logro de la calidad, gracias a que permiten efectuar correctas mediciones a lo largo de los procesos de desarrollo de software, sin embargo, para implementar adecuadamente la gestión de métricas, los ingenieros necesitan de metodologías y/o modelos de calidad, que de igual manera, proporcionan herramientas para el buen cumplimiento de los procesos necesarios a lo largo del ciclo de vida del software.

Las metodologías y modelos de calidad de software describen las actividades que deben llevar a cabo los ingenieros para construir un producto que cumpla con las expectativas de los clientes, y aporten información tanto empírica como objetiva, acerca de las facultades y falencias del equipo de desarrollo, y con ello, construir planes de mejora. Además, las empresas de desarrollo de software, adquieren reconocimiento al certificarse en alguna metodología o modelo de calidad, logrando así, transmitir seguridad y confianza, y tener la posibilidad de extender sus estrategias de comercialización [24].

Dado que la calidad de los productos de software se ve determinada por la calidad del proceso con el que se desarrolla [25], a continuación se describen las metodologías y modelos de proceso más representativos en el contexto del software.

6.3.1 Metodologías de proceso de calidad de software

6.3.1.1 Personal Software Process (PSP)

PSP fue propuesto por Watts Humphrey [20] y dirigido a estudiantes. El libro de Humphrey, An Introduction to Personal Software Process [6], afirma que PSP es un proceso de auto-mejora diseñado para ayudar a controlar, gestionar y mejorar la forma de trabajar de los ingenieros en un marco estructurado de formas, directrices y procedimientos para el desarrollo de software. En la ilustración 1 se puede observar el flujo de procesos general de PSP.

PSP se concentra en las prácticas de trabajo de los ingenieros de una forma individual [10] con la meta de producir software de calidad, diseñado para enseñar a los profesionales cómo planear y darle seguimiento a su trabajo, y a utilizar un proceso bien definido y medido [20]. PSP demuestra a los ingenieros cómo manejar la calidad desde el principio del trabajo [6], a analizar los resultados de cada trabajo y a cómo utilizar los resultados para mejorar el proceso del proyecto siguiente.

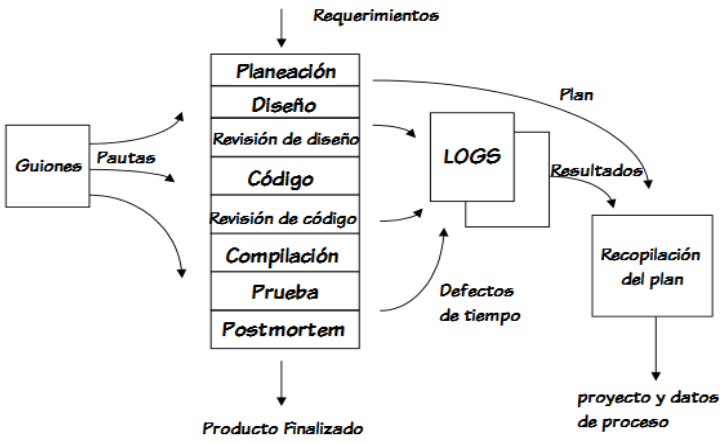


Ilustración 1. Flujo de procesos en PSP. Fuente [38].

La estructura del proceso PSP comienza con los requerimientos. Hay un script de planificación que sirve de guía para este trabajo y un resumen de planificación para registrar los datos de planificación [20] [38]. Los ingenieros registran el tiempo y los datos de los defectos. Al final del trabajo, durante la última etapa, los ingenieros suman todos los datos de los defectos con

los tiempos, miden el tamaño del programa e ingresan los datos del resumen del plan. Luego se entrega el producto terminado en el resumen de la planificación [20].

Un punto importante a mencionar es la recolección de datos en PSP. Esta recolección de datos se lleva a cabo mediante diferentes medidas expuestas en [38]: medidas de tiempo, medidas de tamaño y medidas de calidad.

Las medidas de tiempo hacen alusión al tiempo empleado en cada fase, teniendo registro del momento en el que se comenzó a trabajar en una tarea, el monitoreo de cuándo se dejó de hacer y cualquier intervención [38].

Las medidas de tamaño deben correlacionarse con el tiempo de desarrollo para el producto, siendo las líneas de código, según Humphrey, la principal medida de tamaño (a pesar de la existencia de otras métricas más confiables). Las líneas de código se refieren a la construcción de la lógica del lenguaje de programación con una gestión que no tome mucho tiempo y sea lo más exacto posible, sin embargo, esta métrica de tamaño ha sido sustituida especialmente por la métrica de puntos funcionales por la poca equivalencia que ofrece. En cuanto a las categorías de tamaño se consideran [38]: Base, código añadido, código modificado, código eliminado, código nuevo y modificado, código reutilizado, nueva reutilización y el total.

Para la eliminación temprana de defectos, PSP incluye el diseño y revisión de diseño en el que los ingenieros personalmente revisan sus productos antes de las pruebas. Los ingenieros entrenados en PSP pueden encontrar un promedio de 6,52 defectos por hora en las revisiones de código y 2,96 defectos por hora en las revisiones de diseño personales [6].

6.3.1.2 Team Software Process (TSP)

Desarrollado por Watts Humphrey [10] y descrito en su libro Introduction to the Team Software Process [7] como un marco de trabajo definido para un curso de nivel superior mediante la ingeniería de desarrollo en equipo. El objetivo de TSP es suministrar un proceso operacional que ayude a los ingenieros a hacer trabajos de calidad, teniendo como principal motivación la convicción sobre el trabajo extraordinario que pueden hacer los ingenieros en equipo, pero sólo si están formados y entrenados [10]. En la ilustración 2 se puede observar la estructura y flujo, compuesta por tres ciclos.

TSP construye y guía a los equipos que son requeridos en la mayoría de proyectos de ingeniería a través de los pasos de un proyecto de software [7]. Formalmente, sus objetivos son ayudar a los equipos de ingeniería de software a elaborar productos de calidad dentro de los costos y tiempos establecidos, tener equipos rápidos y confiables y optimizar el performance del equipo durante todo el proyecto [10].

El equipo es un grupo de personas que comparten un objetivo en común. De acuerdo a Humphrey [7], un equipo es un conjunto de dos o más personas que suele ser muy efectivo cuando sus miembros conllevan una comunicación y trato muy abierto. Los miembros del equipo deben tener roles, los cuales proveen un sentido de liderazgo y pertenencia. Los roles ayudan a los miembros del equipo a realizar su trabajo, prevenir conflictos y establecer un grado de control respecto a su ambiente. Además, TSP provee un proceso operacional definido que guía a los ingenieros y directivos en los pasos para la construcción de un equipo, estableciendo un ambiente de trabajo efectivo y que guía a los equipos a realizar su trabajo [10].

El diseño de TSP consta de siete decisiones principales de diseño expuestas en el libro de Humphrey [7]:

1. Proveer un marco de trabajo simple que se constituye sobre la base de PSP. El propósito del proceso es ayudar a desarrollar una tarea con las asignaciones de rol, por lo que TSP ayuda a tener un enfoque de lo que realmente necesita aprender el ingeniero.
   Los roles de TSP especifican quién va a hacer cada una de las tareas: planeación, registro, calidad, soporte y liderazgo, lo que ayuda a desarrollar experiencia en el desarrollo de software y a apreciar los intereses personales.
2. Desarrollo de productos en distintos ciclos. En un proceso de implementación completa de TSP, cada uno de los ciclos incluye un proceso completo de requerimientos, diseño, implementación y pruebas. En el primer ciclo se construye un subconjunto de función mínima del producto final. El segundo ciclo consta de los resultados construidos en el ciclo anterior a partir de los cuales se puede cambiar roles de equipo, ajustar el proceso y utilizar métodos de calidad más disciplinados. El último y tercer ciclo, consiste en la información de 2 proyectos completos al final después de 2 o más ciclos dejando la sensación de experticia y confianza de seguir utilizando los métodos.

3. Establecer medidas para calidad y desempeño. Las medidas son una parte esencial a llevar a cabo para lograr un trabajo consistente de alta calidad.
4. Proveer medidas precisas para equipos y estudiantes. Las medidas hacen que el desempeño personal sea invisible, en cambio, si se hace un esfuerzo razonable se puede estar a la espera de buenos comentarios de los compañeros de equipo.
5. Usar evaluaciones de rol y equipo. Para estar mejor informados del desempeño del equipo y de cada uno, se hace una evaluación de roles, no de las personas, con el fin de evitar dificultades y lograr una mayor objetividad.
6. Requiere disciplina de proceso. No hay una constancia del trabajo personal: No hay tradición en la ingeniería de software, el proceso de software no impone una disciplina natural y se requieren altos estándares y apoyo competitivo.
7. Provee guía para problemas de trabajo en equipo. Inclusive en los proyectos más formales enfrentan problemas de proceso y se ha descubierto que la mejor forma de resolver los problemas de equipo es la presión.

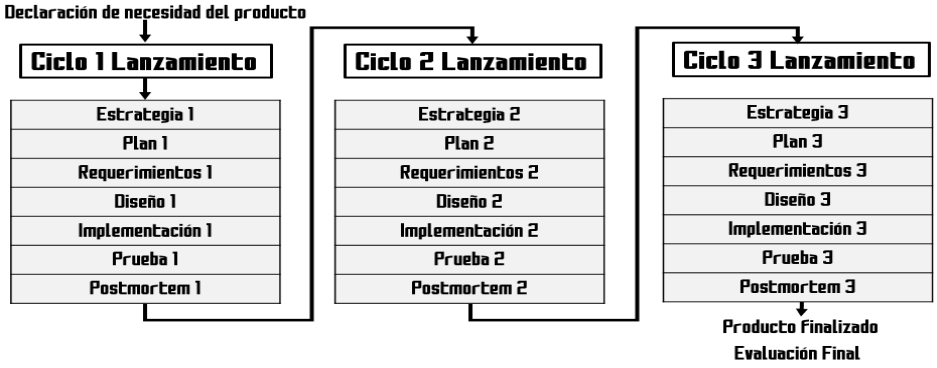


Ilustración 2. Estructura y flujo de TSP. Fuente [7].

Lo interesante de la estrategia por ciclos es poder enfrentar productos grandes de software, y en TSP se manejan las siguientes restricciones [7]: Cada ciclo debe producir una versión que pueda probarse en un subconjunto o propiedades del producto final, cada ciclo debe ser lo suficientemente pequeño para entregar nuevos desarrollos y permitir una constante disponibilidad, y la combinación de los productos deben conllevar al producto deseado [20].

Una vez el equipo TSP es lanzado, se necesita asegurar que los miembros del equipo sigan el plan. El equipo examina el plan de calidad para ver si los parámetros de calidad son razonables y si corresponden con los objetivos de calidad. En caso de no ser así, los ingenieros deben estimar y generar un nuevo plan de calidad [20].

El principal énfasis de la calidad está en el manejo de defectos, por lo que los equipos deben establecer medidas de calidad, determinar objetivos de calidad, establecer planes para alcanzar los objetivos, medir el progreso de los planes y efectuar una acción de recuperación cuando no se alcanzan los objetivos. Una vez el equipo tiene identificado los módulos o componentes que tienen problemas de calidad, se sugiere monitorear mediante TSP el módulo durante la prueba para determinar los problemas y sus acciones correctivas respectivas, valorar e inspeccionar el módulo antes de la prueba del sistema, revisar el módulo para determinar los problemas encontrados y volver a desarrollar el módulo [20].

En la tabla 2 se pueden observar los formularios de TSP y las fases en las que deben llenarse para el correcto cumplimiento de la metodología. La letra *X* significa que estos formularios deben ser completados en su respectiva fase, el símbolo *?* significa que estos formularios pueden o no ser completados en su respectiva fase y las letras *E* y *T* representan al ingeniero en forma individual y al equipo respectivamente; *E/T* es la unión de los dos conjuntos anteriores [8].

El objetivo del presente proyecto es entonces, implementar una herramienta de soporte a TSP para la gestión de los formularios de soporte de TSP y las métricas asociadas.

Tabla 2. Fases en las que los formularios de TSP deben ser completados. Fuente [8].

Formulario	***Abreviatura***		***Fases del Proceso***							
		Ingeniero O Equipo	***LAU***	***STRAT***	***PLAN***	***REQ***	***DES***	***IMP***	***TEST***	***PM***
Solicitud de Cambio de Configuración	*CCR*	*E*				*X*	*X*	*X*	*?*	
Reporte de Estado de Configuración	*CSR*	*T*				*X*	*X*	*X*		
Hoja de Información de Estudiante	*INFO*	*E*	*X*							
Registro de Rastreo de Problemas	*ITL*	*T*		*X*	*X*	*X*	*X*	*X*	*X*	
Registro de Defectos	*LOGD*	*E/T*				*X*	*X*	*X*	*X*	
Registro de Tiempo	*LOGT*	*E*		*X*	*X*	*X*	*X*	*X*	*X*	*X*
Registro de Prueba	*LOGTEST*	*E*						*X*	*X*	
Evaluación de Equipo y Pares	*PEER*	*E*								*X*
Propuesta de Mejora de Proceso	*PIP*	*E*		*?*	*?*	*?*	*?*	*?*	*?*	*?*
Planeación de Calendario	*SCHEDULE*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*
Registro de Estrategia	*STRAT*	*T*		*X*			*X*			
Resumen de Defectos Inyectados	*SUMDI*	*T*								
Resumen de Defectos Removidos	*SUMDR*	*T*								
Resumen del Plan de Programa	*SUMP*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*
Plan de Calidad	*SUMQ*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*
Resumen de Tamaño	*SUMS*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*
Resumen de Tiempo	*SUMT*	*T*								
Resumen de Tarea	*SUMTASK*	*E*								
Resumen de Planeación	*TASK*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*
Estado Semanal	*WEEK*	*E/T*			*X*	*X*	*X*	*X*	*X*	*X*

6.3.2 Modelos de proceso de calidad de software

6.3.2.1 ISO/IEC 15504 (SPICE) [20]

Es un estándar dirigido a la evaluación y determinación de la capacidad y mejora continua de procesos de ingeniería de software. Su filosofía es desarrollar un conjunto de medidas de capacidad, estructuradas para todos los procesos del ciclo de vida y para todos los participantes.

La estructura estándar de ISO/IEC consiste en la valorización de una organización de desarrollo de software en dos dimensiones:

- Dimensión del proceso. Define un conjunto estándar de procesos para el ciclo de vida completo del software. Parte de tres clases básicas de procesos (primaria, soporte y organizativas).
- Dimensión de la capacidad del proceso. Se sustenta en un conjunto de atributos que determinan el nivel con el objetivo de definir la escala de medida para la capacidad del proceso.

El resultado de la evaluación ISO/IEC consiste en la definición de criterios de conformidad para permitir la comparación de modelos externos de procesos y encontrar requisitos comunes. Las problemáticas que enfrenta este modelo se asocian a la significativa complejidad de las evaluaciones y la dificultad de que todos los atributos de proceso sean universales y por ende, aplicables a todos los procesos y prácticas base.

6.3.2.2 ISO 9001:2015

La primera edición de la norma ISO 9001 se publicó en 1987, y teniendo en cuenta el tiempo transcurrido, se ha transformado en la más popular de todas las familias de normas publicadas por la Organización Internacional de Normalización ISO. La más actualizada norma de esta serie es la ISO 9001:2015 que favorece con un enfoque externo a la gestión de la calidad como parte de su perspectiva basada en riesgos [39] [40], e involucra los siguientes componentes: Ámbito de aplicación, Normativas de referencia, Términos y definiciones, Contexto de la organización, Liderazgo, Planificación, Soporte, Operación, Evaluación del rendimiento y Mejora.

En cuanto a los cambios de contenido con respecto a versiones anteriores, la norma ISO 9001:2015 toma el enfoque de riesgos sobre los requisitos en varias cláusulas de la norma, como lo es en el enfoque a procesos, en el liderazgo y especialmente en la planificación [41]. Se deja

además en claro la importancia de la documentación del sistema de calidad en torno a los riesgos que ponen en peligro la conformidad de los productos o servicios, la satisfacción del cliente y el buen rendimiento del sistema de gestión de calidad [40] [41].

El documento ISO 9001:2015 ha sido escrito tomando en cuenta los desafíos actuales de los negocios y para organizaciones de cualquier tamaño y cualquier industria [42]. A diferencia de sus versiones anteriores, ahora se exige a la dirección de máximo nivel que demuestre una implicación más directa en el sistema de gestión de calidad de la organización, con especial atención en el liderazgo [40] a la vez que es lo suficientemente flexible para ofrecer múltiples servicios a las organizaciones [42].

Se le critica a la norma ISO:9001 por su estructura genérica y el no enfocarse al mejoramiento real de los procesos, limitándose a adjuntar gran cantidad de documentos para validar una lista de chequeo [12].

6.3.2.3 CMMI [43]

De acuerdo a la página oficial y actualizada del CMMI [43], *Integración de Modelos de Madurez de Capacidades*, éste es un modelo de mejora de la capacidad que se puede adaptar para resolver cualquier problema de rendimiento en cualquier nivel de la organización en cualquier industria. El modelo proporciona directrices y recomendaciones para ayudar a una organización a diagnosticar problemas y mejorar el rendimiento. Es utilizado por más de 5000 organizaciones de más de 70 países de todo el mundo, ayudando a identificar y alcanzar los objetivos de negocio medibles. CMMI está diseñado para mejorar los procesos de una organización, adoptando las mejores prácticas desarrolladas y probadas por los miembros de la industria, el gobierno y la academia. Es así como CMMI ayuda a las empresas a mejorar su rendimiento operativo, a reducir el costo de desarrollo, la producción y la entrega [43].

Tabla 3. Modelo Continuo y Modelo Escalonado de CMMI. Fuente [44].

Modelo Continuo (Capacidad)	***Modelo Escalable (Madurez)***
0. Incompleto: Proceso que se ejecuta parcialmente o no se ejecuta.	
1. Realizado: Es un proceso que satisface las metas específicas del área de proceso. Soporta y permite el trabajo necesario para producir los productos del trabajo.	*1. Inicial. Los procesos son generalmente ad-hoc y caóticos. La organización generalmente no proporciona un entorno estable para dar soporte a los procesos.*
2. Gestionado: Es un proceso realizado que tiene la infraestructura básica dispuesta para soportar el proceso. Se planifica y ejecuta de acuerdo a políticas; emplea personal con habilidades; tiene los recursos adecuados para producir resultados controlados.	*2. Gestionado. Los proyectos de la organización han asegurado que los procesos se planifican y realizan de acuerdo a políticas; los proyectos emplean personal con habilidad que dispone de recursos adecuados para producir resultados controlados.*

3. Definido: Es un proceso gestionado que se adapta a partir del conjunto de procesos estándar de la organización, de acuerdo a las guías de adaptación de la organización, y contribuye a los activos de proceso de la organización con productos del trabajo, medidas e información adicional de mejora de procesos.	*3. Definido. Los procesos son bien caracterizados y comprendidos, y se describen en estándares, procedimientos, herramientas y métodos. El conjunto de procesos estándar de la organización, que es la base del nivel de madurez 3, se establece y mejora a lo largo del tiempo.*
4. Gestionado cuantitativamente: Es un proceso definido que se controla utilizando técnicas estadísticas y otras técnicas cuantitativas. Se establecen los objetivos cuantitativos de calidad y de ejecución del proceso, y se utilizan como criterios para gestionar el proceso.	*4. Gestionado cuantitativamente. La organización y los proyectos establecen objetivos cuantitativos en cuanto al rendimiento de calidad y del proceso, y los utilizan como criterios en la gestión de los procesos.*
5. En optimización: Es un proceso gestionado cuantitativamente (nivel de capacidad 4) que se mejora en base a una comprensión de las causas comunes de variación inherentes al proceso.	*5. En optimización. Una organización mejora continuamente sus procesos basándose en una comprensión cuantitativa de las causas comunes de variación inherentes a los procesos.*

CMMI tiene dos modelos de representación: El continuo y el escalonado. El Modelo Continuo clasifica los niveles de capacidad de 0 a 5 en diferentes áreas de proceso relacionadas con el desarrollo de software; el Modelo Escalonado clasifica los niveles de madurez de los procesos de desarrollo en una organización entre 1 y 5. (Ver tabla 3).

El uso disciplinado de TSP le permite a una organización abordar con más probabilidad de éxito un proceso de certificación en CMMI; tanto así, que los estándares de TSP se pueden homologar con las prácticas de nivel 3 o superior de CMMI [13].

6.3.2.4 ITMark

Diseñado por ESI (European Software Institute), fundación privada y sin ánimo de lucro creada en 1993 por la comisión europea en colaboración con el gobierno vasco e integrada en la corporación tecnológica TECNALIA. ITMark evalúa y acredita la calidad de las empresas en tres grandes áreas [15] [45]: Una relación con la gestión general de la empresa (estratégica, comercial, financiera, marketing, entre otros), el área de seguridad de la información y la tercera área en la madurez de los procesos de software [20].

El modelo ITMark está pensado específicamente para pequeñas y medianas empresas de TI con base en modelos internacionalmente reconocidos y de gran peso en el mercado [46] [47]. Consiste en un modelo escalable [10] diseñado para PyMES [2] que engloba modelos de mejora en un único esquema de certificación mediante procesos técnicos y de negocio que se adecua fácilmente en dicho sector TI [47].

6.3.2.5 Light MECPDS [15]

El modelo Light MEPDS está basado en el modelo SPICE en busca de brindar una metodología fácil y accesible a PyMES latinoamericanas, principalmente colombianas. De

acuerdo a Pino et al (2006), es un modelo ágil de procesos de mejora de software basado principalmente en metodologías y principios ágiles, requerimientos livianos y adaptaciones de modelos internacionales enfocados en PyMES y medianas empresas, cuya finalidad es motivar la aplicación de proyectos de mejora de procesos en empresas de desarrollo de software iberoamericanas con el objetivo de crear, aplicar y probar un sistema de mejora de procesos con base en otros modelos de calidad reconocidos internacionalmente.

6.3.2.6 COMPETISOFT [49]

Modelo de procesos dirigido a las empresas o áreas internas dedicadas al desarrollo y/o mantenimiento de software. Este modelo permite a empresas sin procesos establecidos ajustar algunos, de acuerdo a las necesidades y experiencia adquirida, y a empresas con procesos establecidos, usarlos como referencia para identificar elementos que les haga falta recubrir.

El modelo está conformado por tres categorías: alta dirección, gerencia y operación. La alta dirección establece la razón de las organización a partir de la definición de alcances y estrategias, la gerencia incorpora planes de acción para instrumentar estrategias necesarias en cuanto a proyectos, procesos y recursos, y la operación consiste en la realización de proyectos o mantenimiento de software en el tiempo y costos esperados, y reportando los resultados obtenidos.

6.3.3 Estado del arte de la metodología TSP

Con base en las metodologías y modelos de proceso descritos en el apartado anterior (5.3), este apartado pretende mostrar las características fundamentales, que hacen a la metodología TSP, la que mejor fomenta la cultura disciplinaria de calidad mediante el uso de métricas a través de mediciones detalladas y suficientes. Se describen los modelos propuestos en artículos y proyectos para la mejora de la metodología, los cuales serán base en la construcción de la propuesta prototipo del presente documento.

Pese a que la metodología TSP ha proporcionado satisfactorios resultados, no ha sufrido grandes cambios respecto a sus procedimientos, actividades y formatos. El estado del arte de esta metodología se divide principalmente en dos aspectos: informales y formales. Las metodologías informales se refieren a los modelos propuestos implícitamente por organizaciones o cursos que pretenden llevar a cabo un proceso evolutivo de la metodología con base en la experiencia. Las metodologías formales, por el contrario, documentan explícitamente las características del modelo

que pretenden simplificar, facilitar y mejorar, de la metodología propuesta originalmente por Humphrey.

A continuación se dará a conocer por qué la metodología TSP sobresale con respecto a la calidad y el uso adecuado de métricas, y su estado actual frente al intento de los ingenieros por perfeccionarla.

6.3.4 La metodología TSP y la calidad de software

El éxito y reconocimiento que reciben las empresas de desarrollo de software se debe en gran medida a la calidad de los productos que ofrece. La implementación de la metodología TSP es la que ha permitido a grandes empresas adoptar procesos de calidad realmente beneficiosos para sus clientes y para sí mismas.

6.3.4.1 Casos de aplicación de TSP

La metodología TSP ha demostrado ser realmente útil para el desarrollo de proyectos de ingeniería de software, aspecto que se evidencia en el trabajo del equipo mexicano SPEG [50], que con la ayuda de TSP, pudo llevar a cabo la planeación y rastreo del proyecto. La estrategia implementó CMMI en múltiples ciclos y reuniones semanales de control y gestión del plan; al final de cada ciclo se hacía un post mortem y un relanzamiento al inicio de cada nuevo ciclo.

El impacto más destacable de TSP, en el proyecto del equipo SPEG, fue la capacidad de detectar errores de estimación con base en información histórica y posteriormente aplicar acciones correctivas. La técnica implementada en este trabajo omite la fase de aprendizaje y se centra únicamente en la recolección y análisis de métricas en forma incremental durante los post mortem [50].

SPEG reconoce que TSP hace más fácil la implementación de CMMI [50] y más aún, un estudio en torno a la aplicación de CMMI mediante la planeación con TSP, afirma que CMMI está reforzado por TSP debido a que se provee información clara acerca de cómo llevar a cabo prácticas específicas (liderazgo, formatos e instrucciones) y provee roles bien definidos que soportan el establecimiento de responsabilidades [51].

6.3.4.2 Algunos casos de éxito de TSP [52]

- BBVA Bancomer. Una de las empresas que ha llevado a cabo en distintos procesos las metodologías, tanto de PSP como de TSP, con éxito en países como Cuba, Colombia, Venezuela y España; todo esto con la ayuda de dos instructores de PSP y un coach de TSP, todos autorizados y reconocidos por SEI.
- Csoftmty. Empresa de desarrollo de software de México, ubicada en Monterrey, la cual tiene implementado la metodología TSP y PSP bajo el modelo CMMI.
- Seonti. Empresa que utiliza con éxito la metodología TSP y que ha ayudado a otras empresas a lograr reconocimiento por parte de SEI (como a Sigma Tap Software) a través de capacitación en TSP y PSP.

Un ejemplo destacable de la implementación de la metodología TSP es Microsoft. La empresa utiliza PSP/TSP para mejorar significativamente sus procesos, obteniendo así, en uno de sus proyectos, tan solo cinco defectos en la prueba integración del sistema, 1.36 defectos por KLOC en prueba de sistema y un 96% de exactitud en los tiempos (con dos semanas de tardanza) [53].

6.3.4.3 Ventajas de la metodología TSP

La metodología TSP ha probado ser efectiva en un gran rango de organizaciones a nivel mundial [54]; facilita la implementación de CMMI, brinda un proceso general de equipo como vehículo de entrenamiento [50] y permite cubrir la demanda actual de ingenieros efectivos para desarrollar software en equipo [51]. En años recientes su crecimiento ha aumentado especialmente en Estados Unidos, México y Asia, lo que ha hecho que se considere uno de los procesos de software con mejores resultados de desempeño, al contribuir gradualmente a la mejora personal, en la calidad de los productos de desarrollo y conducir a una alta reducción de costos en las organizaciones, haciéndolas más efectivas y eficientes [54].

A continuación se listan las más representativas ventajas de la metodología TSP, de las que cabe aclarar, se aceptan como verídicas frente a la investigación realizada.

- Orientado a grupos/equipos, proporciona un ambiente de trabajo favorable. Brinda un plan de trabajo bien estructurado y una buena comunicación entre los integrantes del equipo [55].
- Mejora la productividad de las personas [55] [56].

- Se establece la detección temprana de defectos y riesgos en la producción y desarrollo de software; mejora la calidad del producto terminado [55] [56].
- Mejora de hábitos de programación [56].
- Identificación clara de los objetivos del proyecto [55].
- Enlaza cada fase del proyecto para propiciar el desarrollo esperado. [55].

6.3.4.4 Desventajas de la metodología TSP

Con base en las ventajas mencionadas, la metodología TSP en definitiva, cuenta con un buen proceso de administración de calidad; sin embargo, ejecutarlo es un proceso costoso, representando un alto esfuerzo en el aprendizaje de la misma. Adicional a ello, TSP es principalmente utilizado por organizaciones que desean alcanzar niveles más altos de CMMI (niveles 4 y 5) [54], por lo que se considera una metodología pesada y por ende difícil de apropiar.

A continuación se listan las desventajas más representativas de la metodología TSP de las que no se tiene total aceptación con base en la investigación realizada.

- Cada uno de los miembros debe estar completamente comprometido con la causa del proyecto [55] y seguir una disciplina de calidad [56].
- El proceso de llenar la documentación es largo y tedioso [56].
- Se debe contar con un proceso de conjunto de métricas y parámetros de calidad [55] [56].

El motivo por el que no se da completa aceptación con las desventajas mencionadas, es porque todo proyecto de desarrollo de software requiere un real compromiso de sus integrantes sin importar el lugar, organización o metodología que se aplique. Además, la metodología TSP cuenta con un proceso específico para los ingenieros que la implementan por primera vez, brindando información cada vez más refinada y útil en la ejecución de un proyecto [10].

De manera complementaria, es importante mencionar que, aunque la metodología TSP ha representado grandes costos y esfuerzos de parte de las organizaciones, ésta otorga una recompensa significativa a medida que los proyectos avanzan [50]. Las empresas que implementen TSP, deben reconocer que deben pasar por una curva de aprendizaje que se normalizará posteriormente y que, al pasar del tiempo, aumentará la productividad de la empresa, reducirá costos y evitará el re-trabajo [56].

6.3.4.5 TSP frente a los modelos y metodologías de proceso

Cuando se habla de calidad de software, se tiende a hablar necesariamente del modelo CMMI. Esto ocurre porque dicho modelo retoma un conjunto de buenas prácticas de calidad de software, que son capaces de conducir a las organizaciones hacia el éxito, tanto interno como externo, destacando el aumento de productividad y el reconocimiento internacional. Es por esta razón que la metodología TSP recibe mayor reconocimiento; ya que esta metodología es capaz de alinearse perfectamente con el modelo CMMI debido a sus prácticas orientadas a la disciplina y sobre todo, a la generación de un excelente producto software.

Al igual que CMMI, la metodología TSP conlleva a las empresas de desarrollo de software a un mejor posicionamiento y facilita la apertura de nuevos mercados; asimismo, se considera que conlleva costos muy elevados y resulta difícil de implementar, especialmente tratándose de PyMES, por su organización y tamaño [49].

Se puede afirmar que la metodología TSP sobrepasa las expectativas de los modelos SPICE e ISO 9001:2015 porque está orientada hacia la mejora específica del equipo que la implementa, logrando que éste mejore de acuerdo a sus propias capacidades para resolver las necesidades de sus clientes, situación que el modelo SPICE olvida por completo por causa de su estructura genérica. TSP se enfoca realmente en el mejoramiento mediante el registro detallado de actividades, procesos, cambios, entre otros datos, realmente útiles por su nivel de cuantificación; en este aspecto, supera enormemente al modelo ISO 9001:2015, que se centra en la documentación insignificante, tal como se dio a conocer en el apartado 5.3.3.2 [20].

Frente a metodologías y modelos ágiles de desarrollo de software, la metodología TSP afronta varias dificultades relacionadas a la facilidad de implementación. Los modelos ITMark, Light MECPDS y COMPETISOFT están orientados hacia las PyMES, y por ende, hacia la rápida ejecución de procesos dinámicos y de calidad. Lo que se critica hacia los modelos y metodologías ágiles, es su orientación aceptable de calidad, reconociendo que ITMark se basa en CMMI, Light MEPDS en SPICE y COMPETISOFT pretende que quienes la implementen, obtengan la certificación ISO 9001, por lo que estos estándares se consideran aún en etapa de maduración [57]. De acuerdo al estudio realizado por la Universidad Nacional de Patagonia Austral [57], el enfoque de desarrollo ágil de software, cubre menos factores de calidad que los factores orientados a objetos, componentes y aspectos, tal como se muestra en la tabla 4.

Tabla 4. Factores de calidad utilizados respectivamente en cuatro enfoques de desarrollo de software: orientado a objetos, orientado a componentes, orientado a aspectos y orientado a metodologías ágiles. Fuente [57].

Factores de Calidad	Enfoques			
	DSOO	DSBC	DSOA	MA
Funcionalidad		X		X
Usabilidad		X		
Mantenibilidad	X	X	X	X
Eficiencia		X		
Fiabilidad	X	X		
Adaptabilidad	X			
Reusabilidad	X		X	
Comprensibilidad	X		X	
Flexibilidad			X	

Las metodologías ágiles cuentan con elementos positivos que se asemejan a la metodología TSP. Al igual que en la metodología TSP, las metodologías ágiles se llevan a cabo mediante ciclos de trabajo, que al finalizar, deben proporcionar un producto funcional, de acuerdo a las expectativas del ciclo, y que, con cada ciclo, los integrantes (que dependen de una comunicación constante), se reúnen para discutir acerca de lo hecho hasta el momento y de lo que falta por hacer [57].

Reconociendo que TSP es la metodología que mejor saca provecho a la documentación y al crecimiento disciplinario de los equipos de desarrollo de software, con un enfoque de calidad basado en mediciones precisas, se pretende facilitar su implementación a través de una herramienta prototipo para incentivar el desarrollo de software de calidad de forma eficiente, y en especial, a bajo costo.

6.3.5 Metodologías híbridas TSP

6.3.5.1 Metodologías TSP informales

El Estado del Arte en TSP se ve especialmente impactado por proyectos, reales e hipotéticos, desarrollados por estudiantes a lo largo de sus últimos años de estudio. Esto se evidencia en los posteriores trabajos a mencionar, en los que se tiene cierto grado de flexibilidad en el uso de los formatos y reglas de TSP. Uno de éstos es el proyecto *Capstone* [14], de Australia; consiste en el uso de una versión híbrida de TSP implementada por todos los equipos del proyecto, en el que, adicionalmente cada estudiante debe desarrollar sus procesos individuales, recolectar métricas y demostrar que ha logrado los objetivos de la gestión de proyectos en concordancia al rol particularmente asignado [14].

En el proyecto *Capstone* [14] los estudiantes identifican qué métricas utilizar mediante el proceso Objetivo-Preguntar-Métrica (GQM), lo que adicionalmente requiere que los estudiantes formulen metas, submetas y declaren éstas en forma de preguntas. Durante el curso se lleva a cabo un seminario de una hora durante doce semanas de clase en ambos semestres, en el que se introduce TSP, tópicos básicos de relevancia de la gestión de proyectos (incluyen valor ganado, establecimiento y medición de metas cumplidas) y la presentación formal de trabajo en equipo, donde los estudiantes dan a conocer sus revisiones literarias respecto al caso y dan a conocer en forma verbal el estatus de su rol en torno a las métricas recolectadas [14].

En la Escuela de Ciencias de la Computación, Universidad Politécnica de Madrid, se aplica el entrenamiento en TSP a los equipos de estudiantes de cuarto año, para lo cual han realizado varias muestras escritas de las experiencias adquiridas en distintos cursos. A continuación, se dan a conocer cuatro de sus entrenamientos de un tiempo total de un año, con dos ejercicios compuestos de factores humanos y desarrollo específico del proyecto con TSP, respectivamente [17] [58] [59] [4].

Los estudiantes de la Escuela de Ciencias de la Computación llevan a cabo dos ejercicios globales. En el primer ejercicio, los estudiantes identifican su tipo de personalidad (interventor, patrocinador, emprendedor, gerente, integrador o experto) y con base en ello se establecen los roles necesarios para desarrollar el proyecto de software, identificando las características principales: liderazgo efectivo, comunicación intra-equipo y cohesión de equipo. En el segundo ejercicio, cada equipo desarrolla el mismo proyecto con los mismos requerimientos funcionales, con mismas fechas de entrega, dos ciclos de desarrollo y el apoyo constante de sus instructores [17] [59].

Tabla 5. Características de los proyectos de aprendizaje en la metodología TSP. Fuente autor

Número de equipos	***Número de estudiantes***	***Medidas***	***Referencia***
Desconocido	*3 a 5*	*Desconocido*	[14]
Desconocido	*5 a 6*	*Tamaño*	[17]
		Esfuerzo	
		Productividad	
		Densidad de defectos	
		Costo	
31	*5 a 6*	*Tamaño*	[4]
		Esfuerzo	
		Productividad	
		Densidad de defectos	
44 (2 equipos de 22)	*5 a 6*	*Desconocido*	[59]
20	*5*	*Desconocido*	[54]

49	*Desconocido*	*Densidad de defectos* *Tamaño*	[58]

En la tabla 5 se presenta un cuadro comparativo de las características en cada uno de los proyectos de aprendizaje de acuerdo a la información proporcionada. En la primera columna se da a conocer el número de equipos formados en cada proyecto, la segunda columna contiene el número de estudiantes, la tercera columna da a conocer las mediciones seleccionadas en cada proyecto y la cuarta columna indica la referencia a la que pertenece la información registrada en su respectiva fila. Cabe resaltar que para el caso [58] se tuvieron en cuenta las medidas de tamaño, esfuerzo, productividad y densidad de defectos, expuestas por Humphrey y Hilburn y en [4], además de las medidas anteriores, se tuvieron en cuenta las métricas propuestas por David H. Withers: costo, esfuerzo, calendarización y defectos.

Los roles asignados para los estudiantes no están especificados más que para tres de las implementaciones de aprendizaje, las cuales comparten los mismos: Líder de Proyecto, Gerente de Desarrollo Gerente de Planeación, Gerente de Calidad/Proceso y Gerente de Soporte [14] [4] [54]. En el primer caso [14], al ser posible la existencia de equipos de menos de cinco personas, en dichas situaciones, los miembros se distribuyen las responsabilidades de los roles no asignados. Otro aspecto importante que se aplica para el mismo proyecto es que los estudiantes deben cambiar de rol a mitad del año para adquirir una mayor experiencia, hecho particular que no desarrolla ningún otro proyecto semejante.

La tabla 6 muestra los formatos TSP utilizados en los distintos proyectos llevados a cabo por los estudiantes. En la primera columna se enuncia el nombre del formato, la segunda columna hace una breve descripción y la tercera columna muestra la referencia del proyecto sobre el cual se utilizó. A partir de estos datos se determina que los formatos TSP de mayor uso son: SUMP y SUMQ, seguidos de SUMS, LOGD, LOGT, PEER, STRAT, TASK, SCHEDULE e ITL.

Tabla 6. Formatos TSP utilizados en los diferentes proyectos desarrollados por los estudiantes de las respectivas fuentes bibliográficas. Fuente autor

Formato	***Objetivo***	***Referencia***
STRAT	*Guiar al equipo a través de la producción de una estrategia de desarrollo TSP y tamaño preliminar y estimados de tiempo*	[51] [59]
SUMP	*Mantener un plan e información real para ensambles de programa. Un ensamble puede ser un sistema con múltiples productos, un producto con múltiples componentes, o un componente con módulos múltiples*	[17] [51] [59] [4]
SUMS	*Para resumir la información sobre tamaño de producto*	[51] [4]

SUMQ	*Mantener un plan e información de calidad real para partes o ensambles. Una parte puede ser un objeto, módulo, componente o producto*	[17] [51] [59] [4]
TASK	*Para producir el plan para cada tarea y proveer una base para rastreo del progreso del calendario aun cuando la tarea no se haya completado en el orden planeado*	[51] [4]
SCHEDULE	*Para planear el calendario y rastrear el valor ganado contra el valor conforme se completan las tareas*	[51] [59]
ITL	*Registrar y rastrear el riesgo del Proyecto y problemas*	[51] [59]
LOGD	*Tiene información sobre los defectos encontrados y corregidos*	[59] [4]
LOGT	*Registra el tiempo invertido en cada tarea del proyecto*	[59] [4]
LOGTEST	*Prueba*	[59]
PEER	*Evaluación en pares y de equipo*	[17] [59]
SUMTASK	*Formulario de resumen de tareas*	[59]
WEEK	*Informa de situación semanal*	[59]

En el primer ciclo del proyecto [4], dirigido por G. Cuenvas, de la Universidad Politécnica de Madrid, se llevó a cabo, además de la formación de los equipos, el primer desarrollo del proyecto de acuerdo a las etapas y formatos establecidos por la metodología TSP. Para el segundo ciclo, los equipos repitieron los mismos pasos del ciclo anterior, obteniendo una versión ampliada del producto con énfasis en las pruebas de integración y validación. Cada ciclo finaliza con la exposición (por equipo) de los objetivos, enfoque del trabajo, rendimiento, dificultades encontradas, puntos débiles, puntos fuertes y lecciones aprendidas, todo esto sumado a la entrega de una evaluación individual de desempeño, con respecto a la contribución al desarrollo del producto [58].

Otra implementación en torno a TSP a cargo de estudiantes, consiste en el desarrollo de una aplicación para efectuar reservaciones [54], en la que los estudiantes debían registrar toda iteración haciendo énfasis en la generación y documentación de preguntas específicas, con sus respectivas respuestas. Al igual que en el primer caso de aprendizaje [14], los estudiantes debían formular sus propias metas individuales y de equipo [54].

Gracias a los constantes registros y la alta disciplina por parte de la Escuela de Ciencias de la Computación y el proyecto software para la realización de reservas [54], es posible afirmar que los estudiantes que desarrollan proyectos con la metodología TSP, evidencian un rápido y creciente aprendizaje empírico que mejora significativamente la forma en que llevan a cabo los proyectos. Por ejemplo, en el proyecto de reservas [54], durante el post mortem del ciclo I, los estudiantes analizaron la información e identificaron formas de mejorar el proceso del ciclo II, especialmente en la planeación. Este hecho hizo que la prueba con el cliente fuera exitosa, ya que todos los equipos tenían casi todas las funciones implementadas con algunos defectos mínimos.

En los proyectos desarrollados por la Escuela de Ciencias de la Computación, los estudiantes adquieren alta experiencia entre el ciclo I y ciclo II [59], lo que se ve reflejado en la disminución de la densidad de defectos inyectados [59] y eliminados [58]. En [58], los estudiantes presentan una mejora significativa del 96.75% de efectividad entre los dos ciclos, con desviaciones cada vez menores.

En el proyecto [17] se afirma que los beneficios del segundo ciclo se resumen en la motivación aumentada, mejor compromiso, niveles más altos de desempeño y tiempo de desarrollo disminuido. Para el caso [4], durante este ciclo, los estudiantes produjeron una versión más amplia de productos, enfocándose en el uso de inspecciones, sistemas y pruebas de integración, lo que evidencia mayor competencia y una visión más centrada al logro de metas. Esta mejora, según Cuevas [4], puede estar basada en un mejor entendimiento de TSP, sin embargo, deben tomarse medidas adicionales para evitar algunos problemas ocurridos, como abandono de grupo, fallo de lectura de medio de almacenamiento y formularios con datos incompletos e inconsistentes [58].

Pese a la falta de control en el proyecto Capstone, a diferencia de los demás proyectos de aprendizaje, los resultados se centran en la realización de preguntas a los estudiantes que hicieron parte del proyecto [14]. 90% reconoció que el proyecto les ayudó a desarrollar y mostrar habilidades de liderazgo, y 100% afirmó que el proyecto Capstone les ayudó a desarrollar mejores habilidades para trabajar en equipo. Al analizar los datos se evidenció el uso de TSP en un 42.8 % con una desviación estándar de 20.8. Se determina que este enfoque híbrido es suficiente para mostrar la significancia de los aspectos de la gestión de ingeniería de software; se espera que las estrategias mejoren con base en los datos históricos obtenidos y, en consecuencia, se logre mayor utilidad a futuro con esta metodología [14].

En resumen, los resultados demuestran que el entendimiento y experiencia en TSP mejora la exactitud de las estimaciones de tamaño, esfuerzo, la productividad y densidad de defectos. Se considera que el entrenamiento en TSP será positivo para todos los equipos de desarrollo de software [4].

6.3.5.2 Metodologías TSP formales

Con el fin de extraer elementos importantes para el desarrollo de este proyecto, además de las informales medidas híbridas ya mencionadas sobre TSP, algunos autores realizan propuestas formales para hacer de TSP una herramienta más simple y en la medida de lo posible, obtener

mejores resultados. A continuación se presentan tres principales propuestas correspondientes a cuatro documentos que dan a conocer una mejora personalizada de TSP en dos esquemas [16]: el uso del análisis Gap en conjunto con TSP [54] y la unificación de componentes adaptables entre Six Sigma, PSP y TSP [60] [61].

6.3.5.2.1 TTSP y LTSP

La primera propuesta de mejora de TSP parte de una visión de esta metodología desde dos dimensiones: Organizaciones TSP y Operación TSP. Las "Organizaciones TSP" cuentan con tres partes generales: Formación de equipo, Lanzamiento de equipo y Trabajo en equipo. La "Operación TSP" se constituye por la Estructura y flujo de TSP, Proceso de TSP y Elementos del proceso TSP [16].

En esta propuesta se plantean dos esquemas informáticos: TSP Personalizado y TSP de Nivel [16]. Con esta propuesta se pretende refinar la idea central de TSP y aplicarla a organizaciones de desarrollo de software de diferente capacidad, procesar la división jerárquica de TSP, guiar a las organizaciones de desarrollo para que gradualmente perfeccionen su trabajo en equipo y proveer un proceso que sea simplificado y pueda seguirse [16].

El esquema personalizado, llamado TTSP, tiene el objetivo de simplificar el TSP debido a que tiene un tamaño muy grande, haciéndolo más ligero y mejorando su operabilidad. Esto se logra gracias a que mantiene los elementos clave de la gestión de tiempo de TSP, remueve elementos sobre-refinados, minimiza los formatos y documentos innecesarios y fusiona el contenido similar [16].

Para personalizar los elementos TSP se realiza lo siguiente [16]:

- Proceso personalizado: El proceso TSP cuenta con cuatro fases (requisitos, diseño, implementación e implantación). En la fase inicial de implantación se puede personalizar el proceso de software.
- Formato personalizado: TSP consta de 21 formatos, incluyendo el KPA del CMMI. Por ejemplo, el formato de equipo y evaluación de pares pueden personalizarse y los formatos de resumen de defectos y resumen de defectos removidos, pueden fusionarse.

- Rol de equipo personalizado: En los equipos de desarrollo de software hay 8 roles, en equipos de menor tamaño, algunos pueden fusionarse, en caso contrario, de ser más, pueden repartirse las responsabilidades de los roles.

El segundo esquema es el LTSP, o TSP de Nivel. Éste busca optimizar el actual proceso de software, con poco incremento en la carga de trabajo cuando las organizaciones deben corresponder primero con TSP y analizar el proceso de software existente [16]. Para lograr tal fin, primero deben considerarse las características jerárquicas de CMMI y PSP, implementar paso a paso, gradualmente mejorar, reforzar la capacidad operacional y finalmente implementar el LTSP correspondiente con el CMMI nivel 2, nivel 3 y PSP [16]. En la tabla 7 se describen a detalle las actividades a desarrollar en LTSP.

Tabla 7. Actividades específicas de LTSP. Fuente [16].

LTSP	KPA	Actividad	Libretos/Guiones	Formato	Mapeo PSP	Mapeo CCM
TSPi	Gestión de tareas y calendarios; Gestión de requerimientos	Lanzamiento; reunión mensual; línea de base de requerimiento de formato; traer documento de requerimientos,; función para determinar prioridad. Construcción de prototipo de interfaz de usuario	LAU; REQ; SEMANA	INFOR; TAREA; SEMANA; CALENDARIO; ESTRATO	PSP0 Proceso de medición individual	Gestión de requerimientos (2) Planeación de proyecto de software (2) Rastreo y vista de proyecto de software
TSP2	Manejo de riesgos/problemas; Gestión de configuración; Gestión de calidad	Desarrollar plan SCM; diseño general; asignación de tareas	DEV; DES; SCM	SUMP; SUMT; SUMTASK; SUMQ; ITL; LOGT; CSR; CCR	PSP1 Proceso de planeación individual PSP2 Proceso de gestión de calidad individual	SCM (2) SQA (2) Revisión de pares (3)
TSP3	Gestión de tamaño; Gestión de prueba	Implementar marco de trabajo; integración de sistema; prueba de unidad; prueba de diseño; prueba de implementación	IMP; PRUEBA	SUMS; LOGD; LOGTEST; SUMDI; SUMDR	PSP1 Proceso de planeación individual PSP2 Proceso de gestión de calidad individual	Ingeniería de producto de software (3)
TSP4	Manejo de re-revisión	Prueba de integración; prueba de sistema; prueba de evaluación	INS; PM	INS; PIP; PEER	PSP2 Proceso de gestión de calidad individual PSP3 Proceso de ciclo individual	Coordinación intergrupal (3) Definición de proceso de organización (3)

6.3.5.2.2 Metodología de Análisis Gap

La segunda propuesta consiste en una metodología de análisis Gap para TSP para ayudar a la evaluación de las prácticas actuales de una organización que esté considerando implementar TSP. Este método desarrollado recolecta información de artefactos, entrevistas con gente clave de la organización y conversaciones informales [54].

El análisis Gap depende del equilibrio de la organización, por tanto requiere alto nivel de planeación, preparación y suficientes fuentes de información sin que éstas excedan su tamaño. Este análisis depende de una serie de pasos que en síntesis constan de: preparación de la investigación de mercado, recolección de información, consolidación de información reunida y preparada en un

documento, producción y presentación del reporte, identificación de factores de éxito y perspectivas de análisis (desempeño, trabajo en equipo, gestión de calidad, gestión de proyecto, verificación y validación y gestión de proceso) [54].

Esta segunda propuesta fue revisada por ingenieros senior en conocimiento de TSP para validar su contenido y al ser los resultados muy positivos, se lleva a la implementación práctica en una organización de rápido crecimiento. Los resultados generales fueron buenos, motivo por el que la organización aceptó los gaps identificados en su reporte [54].

6.3.5.2.3 Incorporación Six Sigma

En cuanto a la tercera propuesta, a través del análisis de elementos TSP/PSP [61], se evalúa cuáles herramientas de Six Sigma pueden incorporarse a TSP/PSP. El análisis estadístico y las herramientas de toma de decisiones de Six Sigma pueden emplearse para la mejora continua y control de proceso en TSP/PSP [60], teniendo una especial atención en las métricas de PSP para proveer una mejor infraestructura y obtener un buen enfoque para Six Sigma [60].

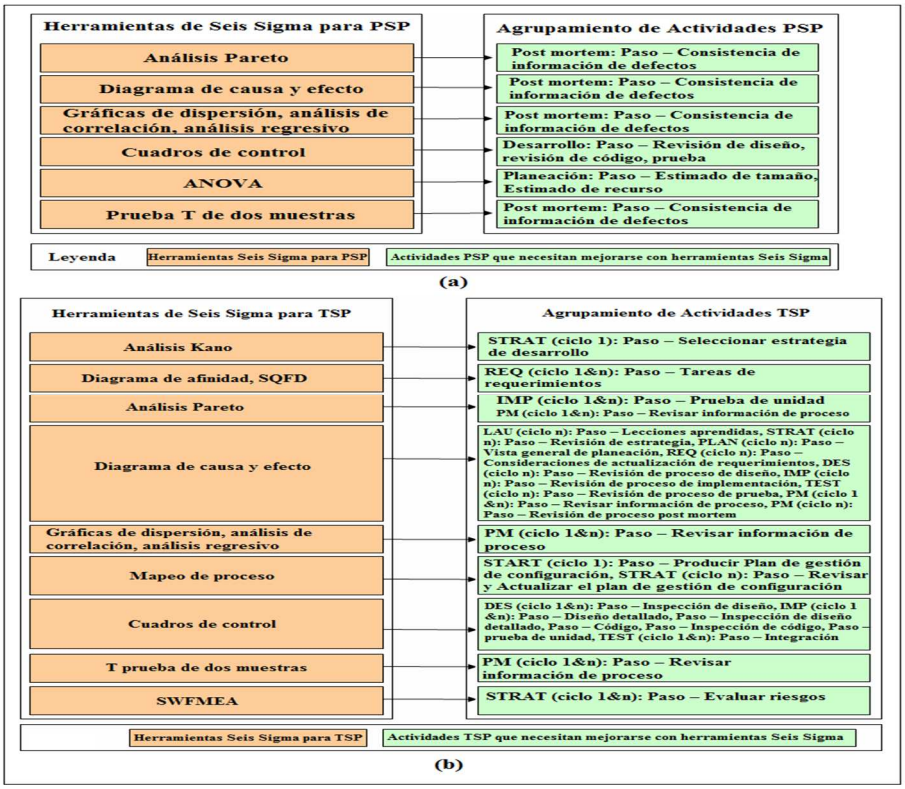


Ilustración 3. Entradas Six Sigma. a. Entradas PSP. b. Entradas TSP. Fuente [61].

El despliegue de TSP/PSP en conjunción con Six Sigma, puede producir capacidades de análisis cuantitativo para identificar actividades de alto apalancamiento, evaluar la efectividad de

los cambios de proceso, cuantificar costos y beneficios, y controlar el desempeño del proceso. De acuerdo a este modelo híbrido, hay sinergia definitiva y significativa entre ellos [60].

La metodología de incorporación de Six Sigma a TSP/PSP especifica las entradas para mejorar la metodología descrita por Humphrey, clasificándolas en: Diagrama de causa y efecto, ANOVA, Gráficas de dispersión, SWFMEA y Mapeo de proceso. La ilustración 3 muestra las entradas Six Sigma para PSP (ilustración 3a) y TSP (ilustración 3b) para cada una de las actividades en las que es posible adaptarse [61].

Lo que básicamente hace la metodología es capturar los LOC y efectuar el PSP normal, tabular la información y verificar el desvío frente a una distribución normal. Se hacen pruebas de varianza; si la información es continua y normalmente distribuida, puede usarse la "Prueba F", y si es continua pero no necesariamente distribuida, se usa la "Prueba Lenove". Se crea un diagrama de caja para visualizar la información, se prueba la hipótesis usando los resultados del resumen y del diagrama, y a partir de un valor P, se define si las estrategias han sido correctas ($P < 0.05$) o incorrectas ($P \geq 0.05$) [60].

6.4 CALIDAD DE SOFTWARE EN COLOMBIA

Colombia es un país, como se explica a detalle en este apartado, que a nivel tecnológico no se ve claramente competitivo internacionalmente. Esto se traduce como falta de calidad y procesos ineficientes en el desarrollo de software, especialmente porque la nación está principalmente conformada de empresas PyMES que no están dispuestas a adoptar metodologías rigurosas. A continuación se describe el problema de calidad que enfrenta el país y las iniciativas que pretenden aumentar su reconocimiento a nivel global; la propuesta de soporte de métricas a través de la metodología TSP puede ayudar al nacimiento de grandes empresas de desarrollo de software, así como apoyar la efectiva ejecución de clusters, todo gracias a la información clara, específica y medible que brindará esta herramienta.

6.4.1 Gestión de la calidad de software en Colombia

Dentro del contexto de calidad en tecnologías de la información, Colombia es descrita con muchos aspectos positivos e igualmente negativos, todo dependiendo del punto de vista desde el que se le juzgue. Documentos como [17] y [10] se enfocan más en presentar sus problemáticas a

nivel de software, resaltando la piratería (por saturación en el mercado) y el desconocimiento de normas [62] y estándares de calidad, lo que es totalmente opuesto a lo que afirman documentos como [17] o [63] al resaltar una rápida maduración en la industria del software en el país.

Scalone [10] y Franco [64] afirman que las soluciones propuestas en Colombia en el contexto de la industria del software, no cumplen con los requerimientos mínimos que satisfacen las necesidades puntuales de sus clientes, ni les permiten competir a nivel de calidad de software en mercados internacionales, lo que lógicamente conduce a problemas de costo, mantenimiento, administración, entre otros. Unido a ello, [10] expone que la realidad de Colombia es la demostración de falencias en el desarrollo de software por causa de métodos artesanales y poco estructurados, lo que provoca un alto nivel de fracaso, dejando en claro que sus causas se deben a la falta de conocimiento sobre los métodos a emplear [10]. Además de todo esto, [17] señala acerca del país, la falta de capacidad técnica del personal de bajo nivel empresarial y las alarmantes experiencias en el desarrollo de software por motivos de sobrecostos, grandes retrasos y consecuentemente la falta de calidad. Según el Estado, "aún falta mucho camino por recorrer" [3].

Otro problema al que Colombia se enfrenta es su alto número de empresas pequeñas y medianas desarrolladoras de software, es decir, las PyMES y microempresas que aparentemente no cuentan con los recursos suficientes para obtener una muy reconocida certificación de calidad [10]. Las PyMES son una pieza muy importante en el avance económico nacional y mundial ya que, no sólo Colombia está mayormente compuesta por este tipo de empresas, sino también gran parte de los países del mundo. Por tanto se pretende fortalecer este tipo de organizaciones, que en el caso colombiano, se lleva a cabo mediante la promoción de otras certificaciones tales como ITMark o Light MECPDS [15]. En el mundo se han certificado 174 empresas en ITMark, de las cuales Colombia lidera el listado de empresas con un total de 80, muchas de ellas apoyadas por el MinTIC [3].

Según Durán [62], la industria colombiana necesita profesionales con una visión total del proceso de desarrollo de software y con la disciplina para enfrentarlo, siendo relevante mencionar que además de las 346 empresas del departamento de Cundinamarca, se destacan empresas de los departamentos de Antioquia (82 empresas) y Valle del Cauca (56 empresas). De todas estas empresas, 48,4% (casi la mitad) de las empresas entrevistadas se dedican al desarrollo de software y el 46% tanto del desarrollo de aplicaciones como la venta de hardware y software [62].

De acuerdo al estudio realizado por [65] a una muestra correspondiente a 103 empresas colombianas destacadas y adscritas a la base de datos de FEDESOFT, las organizaciones colombianas de tecnologías de la información llevan a cabo las siguientes actividades:

- Hacer un plan de sistemas de información cada año, donde se priorizan las actividades de mayor importancia a desarrollar por la empresa en el transcurso del año. Esto es ejecutado por el 50% de las empresas encuestadas.
- Utilizar un modelo o metodología de desarrollo. El 85% de las empresas encuestadas utilizan un modelo o metodología.
- Contar con profesionales altamente calificados. El 64% de las empresas encuestadas tienen entre sus empleados profesionales posgraduados en ingeniería de desarrollo de software.
- Desarrollar comercialmente. 76% de las empresas encuestadas lo hace.
- Desarrollar software a la medida. 84% de las empresas encuestadas lo hace.

En la mayoría de empresas colombianas se trabaja con una mezcla de una serie de metodologías que se apoyan en RUP [65]; tienden a buscar normas parcialmente ordenadas para hacer uso eficiente de sus recursos en el menor tiempo posible.

Para evaluar el proceso de desarrollo en Colombia, [65] divide la información obtenida de las encuestas en tres fases: inicial, desarrollo y final. De acuerdo a los resultados obtenidos, la fase inicial es considerada de alta y muy alta importancia por el 80% de las empresas encuestadas, en la fase de desarrollo se indica un nivel de importancia bajo y muy bajo, sobre todo en las últimas cuatro etapas de esta fase. En cuanto a la fase final, no se observa una tendencia clara, dado que alrededor del 65% de las empresas encuestadas consideran esta fase de alta y muy alta importancia, sin embargo, en sus dos primeras etapas, el 60% de las empresas indicó una baja y muy baja importancia.

Tabla 8. Número de empresas que clasifican cada etapa de la Gestión de Proyectos por nivel de importancia. Fuente [65].

Tareas	***Importancia***			
	Baja	***Media***	***Alta***	***NR***
Planificación	*3*	*23*	*75*	*2*
Gestión de la planificación	*12*	*34*	*55*	*2*
Gestión de incidencias	*20*	*37*	*43*	*3*
Gestión del costo	*10*	*30*	*61*	*2*
Gestión del recurso humano	*12*	*34*	*55*	*2*
Gestión del tiempo	*5*	*34*	*62*	*2*
Gestión del riesgo	*30*	*43*	*27*	*3*
Gestión de la documentación	*19*	*48*	*34*	*2*

Gestión de la calidad	*10*	*28*	*63*	*2*
Gestión de la configuración	*19*	*35*	*47*	*2*
Gestión de los requisitos	*2*	*24*	*75*	*2*

En la tabla 8 se da a conocer el número de empresas que clasificaron cada una de las etapas de la gestión de proyectos de software. A partir de este estudio se determina que en el ámbito colombiano, la etapa de desarrollo de software muestra una tendencia a disminuir la importancia de etapas a medida que transcurren en el tiempo, además de otorgarle una calificación muy alta a la gestión de los procesos, exceptuando la gestión de riesgos y gestión de la documentación.

6.4.2 Tendencia al clúster de software en Colombia

Como alternativa al crecimiento de empresas de desarrollo de software de Colombia, han surgido proyectos que pretenden, mediante clusters, mejorar el desempeño, la competitividad y rentabilidad [66]. Con esta medida, se han podido crear vínculos de confianza y colaboración para ofrecer soluciones integrales [67], que ayudan a la nación a entrar en un territorio más próspero, con mejores salarios y oportunidades de negocio más rentables [68].

Un claro ejemplo de los beneficios que brinda la implementación de clusters en Colombia es CaribeTIC. Consta del agrupamiento de 17 compañías de Barranquilla y Cartagena, que tiene por objetivo incrementar la competitividad de las empresas dedicadas, en estas regiones del país, al desarrollo de herramientas informáticas. Esta estrategia ha generado ingresos superiores a los 100 mil millones de pesos (colombianos) y más de 100 empleos para el año 2012 [69].

De igual manera, en el país se evidencian otros ejemplos como la Iniciativa Clúster de Software y TI de Bogotá y el CLINSA. El primero pretende orientar el modelo digital a la estructura de negocio dentro de contextos como la seguridad digital o el análisis de productividad [70]; y el segundo constituye la relación entre siete empresas colombianas (de regiones como el Valle del Cauca, la Costa Atlántica y Bogotá) [71], para vincular la salud y la tecnología en el desarrollo de dispositivos de uso terminal para pacientes, a fin de mejorar su calidad de vida [72].

Con las grandes ventajas mencionadas acerca de la implementación de clusters en Colombia asociadas al crecimiento de rentabilidad, productividad y competitividad, se puede afirmar que esta estrategia incorpora calidad en sus procesos [69], y por ende, ser un pilar en el fortalecimiento de organizaciones nacionales [72]. Sin embargo, el país necesita de un fuerte

impacto tecnológico, pues, de acuerdo a [73], aún con la implementación de clusters, no se logra un posicionamiento claro ni diferencia a nivel internacional. Esto ocurre porque, cerca del 81% de los ingresos generados por la nación en desarrollo de software, provienen de empresas de la ciudad de Bogotá, entre las cuales sólo el 6% son microempresas, es decir, que el mayor impacto en calidad a nivel internacional recae sobre empresas no interesadas en la implementación de clusters.

Si la mayoría de empresas de desarrollo de software del país se enfocan en seguir procesos bien definidos, la estrategia de clusters será más eficaz en el sentido de que cada parte tendrá mejor definidas sus capacidades, logrando aumentar rápidamente el número de proyectos de esta índole y así, además de los beneficios laborales y económicos, conseguir una clara posición competitiva en el mercado internacional que exige el actual modelo globalizado.

6.4.3 Impulso a la certificación de calidad en Colombia

Con la certificación en modelos de calidad como ITMark y CMMI, 100 empresas de software y servicios presentes en 12 regiones del país han fortalecido sus procesos de planificación, dirección, control y almacenamiento de datos, lo que les permitirá aumentar su productividad y generar mayores capacidades. Colombia es líder en Latinoamérica ubicándose en tercer lugar detrás de México y Brasil [63] [74] [75].

Como recientes impulsos de parte de MinTIC, se han realizado dos acciones importantes para el sector de TI descritos en [76] y [77]. El artículo [76] informa acerca de lo que la viceministra de tecnología y sistemas de información, presentó a las iniciativas del MinTIC en el ciclo de conferencias “Modelos de calidad para la industria del software en Colombia” llevado a cabo en Febrero de 2016, siempre con la intención de promover el fortalecimiento de la industria TI y la implementación de modelos de calidad en el sector. En cuanto a la Cámara de Comercio de Bogotá [77], se da a conocer la propuesta de certificación de calidad gratuita a partir de Enero de 2016 en unión con Colciencias, a la que pueden aplicar empresas construidas como personas jurídicas y establecidas como sociedades mercantiles pertenecientes a la industria de tecnologías de la información, recalcando el cumplimiento de ciertos requisitos orientados al buen orden y gestión interna de la empresa que lo desee.

6.4.4 Impulso TSP en Colombia

MinTIC ha sido un promotor del aprendizaje y certificación en PSP y TSP mediante diferentes convocatorias como las mencionadas en [78] [79] y [80] correspondientes a Septiembre de 2013, Enero de 2014 y Noviembre de 2014 respectivamente. Con estas convocatorias [79], MinTIC busca incentivar la certificación de los estudiantes de ingeniería de sistemas, profesionales y profesionales en el sector TI, lo que proporciona un alto valor agregado al desempeño profesional y laboral. La diferencia fundamental que menciona MinTIC en su cuarta convocatoria [80] es pertenecer "al top de los desarrolladores a nivel mundial", lo que es más definido y relacionado a las necesidades actuales en la industria del software.

El impulso por hacer de Colombia un país altamente reconocido a nivel mundial en la industria del software, es tal, que se resalta en [81] que sólo con el cumplimiento de la formación, las personas garantizan el 90% de la condonación del crédito, por lo que el examen de certificación representa tan solo el 10% restante. Adicional a ello, en [82] se expone el incentivo para TSP por parte directa del ICETEX (igualmente vinculada para los proyectos MinTIC promotores de acreditación en TSP ya descritos), que tiene por objetivo incentivar la formación de capital humano especializado en el uso de tecnologías de la información, fortalecimiento de la industria TI, entre otros.

FEDESOFT es otra organización promotora del uso y certificación de TSP en Colombia. De acuerdo a García [83], FEDESOFT en alianza con el SENA y algunas organizaciones interesadas en implementar calidad de software (incluidas las empresas INTERSOFTWARE y SEONTI [9]), están generando una estrategia que busca implementar PSP y TSP como modelo de desarrollo de software para PyMES. Con el objetivo de hacer que Colombia alcance y rebase a los líderes en desarrollo de software, se obtuvo un proyecto de formación en prácticas internacionales TSP/PSP [9].

Con el objetivo de hacer que Colombia alcance y rebase a los líderes en desarrollo de software, FEDESOFT obtuvo un proyecto de formación en prácticas internacionales TSP/PSP, dando como resultado un total de 1036 personas capacitadas, más de 60 empresas beneficiarias, presencia en 5 regiones del país y más de 10 ciudades participantes [9]. En la ilustración 4 y 5 se

pueden observar los planes iniciales de entrenamiento y el porcentaje de participación en el proyecto por departamento, respectivamente.

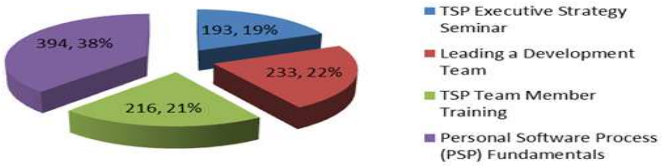


Ilustración 4. Planes iniciales de entrenamiento en el proyecto FEDESOFT. Fuente [84].

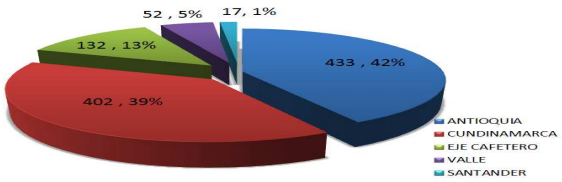


Ilustración 5. Porcentaje de participación de algunos departamentos de Colombia en el proyecto FEDESOFT. Fuente [84].

En un contexto privado y con base en el auge del interés por la metodología TSP/PSP, empresas como TI Institute Colombia y SEONTI ofrecen sus servicios para adquirir aprendizaje y certificación respectivamente. Institute Colombia [85] ofrece un curso de tres días de TSP dirigido a gerentes de software, gerentes de proyecto, líderes de equipo de software y supervisores de equipo de desarrollo, con el fin de mostrar las habilidades de los ingenieros que ya se encuentran capacitados en PSP. Por su lado, SEONTI [86], una empresa que cuenta con oficinas en Estado Unidos, México y Colombia, provee soluciones integrales en la mejora de procesos y calidad de software, siendo TSP/PSP uno de sus enfoques, recalcando que uno de sus casos de éxitos ha sido la certificación de la empresa colombiana PersonalSoft al CMMI nivel 3, gracias a un conjunto de capacitaciones y prácticas en TSP/PSP.

Es un hecho que en Colombia existen empresas y profesionales orientados a CMMI, PSP y TSP en términos de industria del software [84], con más de 500 personas formadas y certificadas en TSP/PSP [87], lo que es suficiente para que sean apetecidas por cualquier empresa TI del mundo [88], razón por la que actualmente es común encontrar requisitos laborales asociados a TSP/PSP en trabajos asociados a las tecnologías de la información. Por ejemplo, las empresas PERSONALSOFT [89] y Aranda Software [90] buscan personal capacitado (o en el caso de la empresa Linexperts Consultoría Empresarial SAS, con ciertos conocimientos [91]) en metodologías de desarrollo de software como RUP, SMF, TSP/PSP, UML y/o modelo CMMI, o hasta el punto de incluir en los perfiles solicitados, conocimientos básicos en TSP/PSP en los requisitos para el cargo de profesor de cátedra en la facultad de ingeniería de Cali [83].

7. ESTADO DEL ARTE

7.1 PROBLEMAS DE TSP FRENTE A LA GESTIÓN DE MÉTRICAS

Como se ha mencionado, TSP es la metodología por excelencia, que brinda a los ingenieros elementos suficientes para lograr alta calidad en los productos software, al tiempo que permite capturar datos precisos y cuantificables para determinar qué procesos se deben mejorar en la ejecución de futuros proyectos [9]. Con la captura de datos de la metodología TSP, es realmente necesaria la gestión de métricas a fin de lograr mediciones reales, objetivas y de significativa importancia en la toma de decisiones por parte del equipo de desarrollo de software [28].

El mayor problema al que se enfrentan los ingenieros en el desarrollo de software es el control de la documentación [24]. Ellos necesitan recolectar y analizar las métricas de una forma eficiente y con el menor esfuerzo posible, logrando así, centrar sus actividades en las necesidades directas del proyecto [12]. Sin embargo, los métodos de recolección son manuales, inexactos y/o conllevan demasiado tiempo, sumado a la difícil manipulación y control de la información, haciendo que ésta se registre incompleta o inadecuadamente [56].

Implementar la metodología TSP propuesta por Humphrey, requiere efectuar demasiada documentación, que sumado a su diligenciamiento manual, ha fomentado el mal e incompleto diligenciamiento de formatos [58]. Los integrantes del equipo de desarrollo deben, además de evaluar el buen desempeño de los demás roles, verificar que la información está correctamente diligenciada y sin ambigüedades. Si la documentación presenta fallas, tanto la identificación manual como sus acciones correctivas, conllevan demasiado tiempo y por ende al re-trabajo [56].

Dada la complejidad de la metodología TSP y la importancia de las métricas en el desarrollo de software, los ingenieros han simplificado la propuesta de Humphrey, para hacerla más sencilla de implementar al tiempo que la orientan a la gestión de métricas. Se han minimizado los formatos, se ha mejorado la operabilidad [16] y se ha centrado en la detección de errores [50], así como del manejo de medidas de tamaño, esfuerzo y productividad. El problema es que aún se siguen cometiendo errores de diligenciamiento e inexactitud a falta de una herramienta capaz de completar automáticamente información exacta y confiable, lo que ha incentivado el surgimiento de software como soporte a dicha metodología.

7.2 ACTUALES HERRAMIENTAS DE SOPORTE PARA TSP

Uno de los mayores problemas para el gerente del proyecto y de los desarrolladores, es cómo medir y mejorar el tiempo de desarrollo; y cómo mejorar el proceso en sí mismo. Con este propósito han surgido algunas herramientas de soporte para TSP, que inclusive se han utilizado por miembros participantes y líderes de equipo de diferentes organizaciones [14].

Todos los proyectos mencionados en el apartado 6.4.2.1 (a excepción de [4] y [13]) reconocen la necesidad de una herramienta TSP, por tal motivo, proponen, utilizan o desarrollan software para la gestión de sus procesos, como parte de sus planes. [92] por ejemplo, recomienda establecer un sistema para gestionar y almacenar archivos y formularios de manera centralizada, logrando así su fácil localización y reutilización, criterio que comparte [54] frente a su necesidad de recolectar información y afirmando que podría ayudar a acelerar el desarrollo de proyectos.

El proyecto [59] de la Universidad Politécnica de Madrid, al requerir de una herramienta, utilizó Excel para el llenado de formularios empleando muchas hojas para la planeación. En los proyectos [16] y [58], de la misma universidad, en cambio, desarrollaron su propia herramienta web orientada al entrenamiento, permitiendo la interacción entre estudiantes e instructores, la publicación de noticias y el control de información a través de un repositorio para todos los integrantes del proyecto.

7.2.1 Team Dashboard

Es la herramienta de soporte de TSP más familiar. Es de código abierto y provee funcionalidades como análisis de información, planeación jerárquica, planeación simultánea, coordinación mutiproyectos en tiempo real, rastreo de dependencia, formatos y reportes personalizables y flujos de trabajo de proceso personalizables [14]. En la ilustración 6 se puede observar la interfaz gráfica inicial de la aplicación.

Aunque la herramienta es popular, su interfaz gráfica no es amigable con el usuario, lo que hace difícil utilizarlo a menos que el usuario tenga experiencia con TSP/PSP. El resultado de diferentes medidas se muestra en números que únicamente pueden ser interpretados por un experto [14]. A nivel empírico, las pruebas sobre el software demostraron que posee varios módulos que tienen la intención de satisfacer gran parte de los requerimientos de un proyecto, sin embargo,

presenta falencias en su interfaz gráfica nada amigable y en el control de información ingresada por el usuario, lo que sumado a la poca experiencia con TSP/PSP de quiénes normalmente lo usarían, no ofrece la confianza suficiente para percibir que los datos estén correctamente almacenados.



Ilustración 6. Interfaz gráfica inicial del software Process Dashboard. Fuente Aplicación original.

7.2.2 tVista

Es una aplicación web para soporte TSP de código abierto, desarrollada por estudiantes de últimos semestres. Cuenta con reporte de inspección, registro de rastreo de problemas, registro de defectos y reportes de resumen (que incluyen defectos inyectados, resumen del plan de calidad, entre otros) [14].

Poco se sabe acerca del software tVista. A partir de la documentación brindada públicamente en su página web, se puede afirmar que el software cuenta con elementos básicos de TSP, tales como cuentas de usuario, roles y registros de defectos. Al observar detalladamente el modelo de datos del proyecto tVista, se deduce que su principal enfoque es el registro de defectos y tiempos de desarrollo, lo cual se lleva a cabo en forma manual y sin el suficiente control que garantice la correcta adición de datos.

7.2.3 Point

Herramienta desarrollada por un equipo de estudiantes de CMU que tenían por objetivo facilitar que múltiples equipos utilizaran su información almacenada periódicamente. Posee una base de datos relacional que provee características de gestión central multiproyectos, gestión de equipo terminado e información personal y consolidada en tiempo real [14]. Su nombre no fue de ayuda para localizar el software, motivo por el que no se tiene una descripción empírica.

7.2.4 Libro de trabajo de TSP

El TSP Workbook es la herramienta de TSP más famosa provista por el SEI. Es un archivo de Excel que cubre todos los formatos del TSP, lo que la hace comparativamente muy fácil de usar si se tiene un extenso conocimiento en TSP [14]. En la ilustración 7 se puede observar la primera vista del archivo Excel de la herramienta.

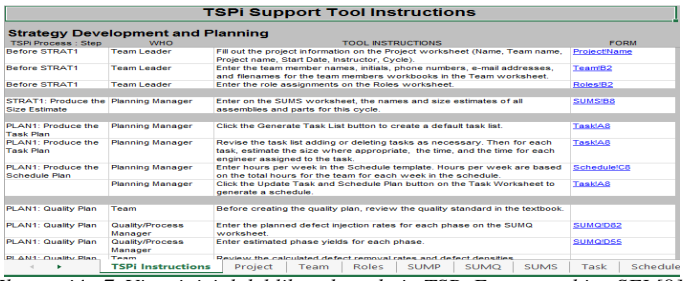


Ilustración 7. Vista inicial del libro de trabajo TSP. Fuente archivo SEI [8].

Como básicamente el programa es un archivo de Excel, la interfaz de usuario no es atractiva y le falta visualización apropiada de la información [14], características que señalan específicamente los resultados empíricos.

7.2.5 Enfoque de proceso guiado

Consiste en una propuesta diferente; es una guía electrónica de proceso para TSP que asiste a los principiantes para ejecutar un proceso junto con la guía típica de TSP y otra con la información importante del proceso. Consta de dos guías: la primera es un documento y la segunda es el Repositorio Electrónico (ER) para compartir experiencias [14].

La interfaz gráfica de esta herramienta cuenta con tres secciones principales: barra de navegación, flujo de proceso y barra de descripción. La aplicación provee un login con base en el soporte del usuario. La barra de flujo de proceso lleva al usuario a buscar desigualdades del TSP en forma gráfica. El usuario puede seleccionar y ver las secciones correspondientes en las subsecciones de descripción [14].

La siguiente parte consta de integrar las diferentes funcionalidades de TSP, aunque aún está en investigación. Esta sección pretende contemplar cómo añadir las funcionalidades y hacerlas más fáciles de utilizar para las organizaciones que no tienen un conocimiento amplio en TSP [14].

7.3 SOFTWARE ORIENTADO A MÉTRICAS DE PROCESO

Aunque la metodología TSP, como se ha mencionado, conlleva una adecuada gestión de métricas de software, no significa que sean suficientes las referencias tecnológicas orientadas exclusivamente a dicha metodología. Con el objetivo de gestionar adecuadamente las métricas en el desarrollo de software, las herramientas tecnológicas enfocadas en este contexto, sirven como guía en la construcción de la propuesta prototipo del presente documento, a fin de desarrollar una herramienta que continúe afinando el estado del arte en torno a la calidad de software.

Una herramienta a citar, proveniente de un proyecto de pregrado de la facultad de Ingeniería de Sistemas, de la Universidad Distrital Francisco José de Caldas, y titulado *Aplicación basada en componentes de software para el apoyo a la captura de métricas de proceso en proyectos de desarrollo de software* [32], tiene por resultado una herramienta colaborativa en la recolección de métricas y control de edición, orientado a equipos de desarrollo de software.

El proyecto de grado permite capturar métricas de proceso de proyectos de desarrollo de software a través de la identificación y recolección de información clasificable como entidades y atributos. Empíricamente, la herramienta es muy amigable con el usuario, permitiendo agregar, editar y eliminar atributos y defectos. La aplicación también permite el registro general de un proyecto de software y sus integrantes, además de controlar cambios, especificando quién los ha llevado a cabo [32].

7.4 PROPUESTA

7.4.1 Aportes significativos

De acuerdo al apartado anterior (apartado 6), el desarrollo de software debe llevarse a cabo mediante procedimientos de significativa calidad, dada la complejidad de los problemas a los que se enfrenta en la actualidad. Con la propuesta de la herramienta prototipo, se pretende, además de continuar con la mejora mediante la recolección de métricas (descrito en el proyecto de grado del apartado 7.3), automatizar la captura de información y la generación de resultados, con relación a los procesos asociados a un proyecto (o conjunto de proyectos) de desarrollo de software. Con esta herramienta los equipos podrán retroalimentar fácil, eficiente, y rápidamente sus procesos de desarrollo, de modo que puedan enfocarse en los problemas que tratan de resolver sin que la documentación y control interno genere costos adicionales, sino por el contrario, que la herramienta sea vital para hacer más eficiente el ciclo de vida de los software a desarrollar.

Según lo expuesto en el apartado 6.4.1, la metodología TSP es la que brinda las mejores herramientas para identificar claramente las habilidades y dificultades de los ingenieros, tanto a nivel personal como de equipo, siendo así, fundamental en la toma de decisiones, el logro de la calidad y el crecimiento tanto interno como externo de las organizaciones dedicadas al desarrollo de software. Por ello, la propuesta, además de automatizar gran parte del proceso de gestión de métricas, pretende satisfacer todas las necesidades procedimentales de los equipos de desarrollo,

a través de elementos tecnológicos que faciliten la implementación de la metodología TSP. Con la herramienta prototipo se mitigarán las desventajas nativas de la metodología como la dificultad de implementación y de los software actualmente existentes (con base en la tabla 9) como la recolección automática de información, repositorios centralizados de documentación y la amigabilidad con el usuario.

Tabla 9. Comparación de las actuales herramientas de desarrollo de software. Fuente [14].

Herramientas / ***Criterio***	***Team Dashboard***	***Point***	***Libro de trabajo de TSP***	***tVista***
Recolección de información	*Parcialmente automático*	*Manual*	*Manual*	*Automático*
Validación de información	*No*	*No*	*No*	*Información no disponible*
Herramienta de análisis estadístico	*Algo*	*No*	*No*	
Comunicación/Discusión entre miembros del equipo	*No*			
Acceso a materiales potenciales (formatos, plantillas)	*No*	*Parcialmente*	*Parcialmente*	*Sí*
Visualización de desempeño e indicadores	*Principalmente numérico con algunas Gráficas y Cuadros*	*Únicamente numérico*	*Únicamente numérico*	*Información no disponible*
Repositorio de experiencia	*Sólo respaldo de base de datos*	*No*	*No*	*No*
Manejo de defectos	*Sí*	*Sí*	*Sí*	*Sí*
Configuración de manejo	*Sí*	*Sí*	*No*	*Sí*
Guía de proceso/Manual de usuario	*Manual de usuario provisto*	*No*	*1 página del manual provista*	*Información no disponible*
Amigabilidad con usuario	*Mínimo*	*Nominal*	*Nominal*	*Información no disponible*
Sigue los formatos y plantillas del SEI	*Parcialmente*	*Totalmente*	*Totalmente*	*Información no disponible*

7.4.2 Valor agregado

Puesto que Colombia está altamente compuesta por empresas PyMES en el contexto de las TI, la implementación de la herramienta software se propone como colaboración a la ejecución de sus procesos para adquirir una cultura disciplinaria y de calidad a modo interno de la empresa. Se recalca que la certificación no debe ser el único incentivo de las empresas de desarrollo de software, sino la producción con calidad. Si las PyMES hacen uso de la herramienta, notarán una fácil implementación de la metodología TSP con resultados realmente significativos en la mejora continua, la preferencia de sus clientes y aumento en el número de proyectos por año.

En cuanto a los clusters en Colombia, la herramienta juega un papel importante en la medida en que su implementación proveerá a cada empresa con procesos bien definidos con los cuales pueden dejar en claro, en forma argumental y cuantificable, los servicios que pueden prestar. Gracias a la implementación de procesos con la herramienta será fácil identificar facultades y limitaciones, y con ello evaluar el grado de aportación de cada integrante en el cluster al que se pertenezca.

8. METODOLOGÍA

Este proyecto se desarrolló utilizando la metodología SCRUM. Con este marco de trabajo es posible desarrollar productos complejos a corto plazo y con la máxima productividad posible, obteniendo funcionalidades en forma incremental e iterativa [93]. A continuación se describe, para el proyecto, la estructura de equipo SCRUM y las actividades y objetivos sprint.

8.1 ESTRUCTURA DEL EQUIPO SCRUM

El equipo de desarrollo contó con dos integrantes: la directora del proyecto, experta en la metodología TSP, y el estudiante responsable de la tesis de grado, con quienes se hizo la siguiente asignación de roles de acuerdo con la metodología SCRUM (ver tabla 10) .

Tabla 10. Asignación de roles SCRUM para el proyecto. Fuente autor

ROL SCRUM	***PERSONAL***
Dueño de producto	*Alba Consuelo Nieto Lemus*
Equipo de desarrollo	*Juan Camilo Camacho Beltrán*
SCRUM Master	*Juan Camilo Camacho Beltrán*

8.2 ACTIVIDADES Y OBJETIVOS SPRINT

De acuerdo a [94], se deben ejecutar, secuencialmente, los siguientes procesos (eventos) para el cumplimiento de cada sprint: planeación del sprint, sprint diario, revisión del sprint y retrospectiva del sprint. En la tabla 11 se describen las actividades genéricas correspondientes a todos los sprint del proyecto; en este caso seis sprint para seis meses, es decir, un tiempo total de un mes por cada sprint.

Tabla 11. Actividades genéricas de sprint para el proyecto. Fuente autor

EVENTO	***TIEMPO DE REUNIÓN***	***DESCRIPCIÓN***
Planeación del sprint	*8 horas*	*Se fijan las metas del sprint*
Sprint diario	*15 minutos*	*Determinar qué se ha hecho, qué se va a hacer y qué impedimentos se han presentado*
Revisión del sprint	*4 horas*	*Se revisa el incremento a la fecha, productos completos y se discuten temas relacionados a problemas y soluciones afrontadas, y de calendario.*
Retrospectiva del sprint	*3 horas*	*Se identifican elementos importantes, tanto positivos como negativos, concernientes al actual sprint*

En la tabla 12 se enuncian los objetivos de cada sprint para la propuesta prototipo. Cabe mencionar que cada sprint se desarrolla bajo las mismas reglas establecidas por la metodología general SCRUM y con un específico propósito, que en conjunto, se orientan al logro del producto final.

Tabla 12. Descripción de los sprints estimados del proyecto. Fuente autor

NOMBRE DEL SPRINT	***DESCRIPCIÓN***
Conformación del proyecto	*Construcción completa de los elementos necesarios para dar inicio al proceso de desarrollo de software. Entre estos elementos se encuentran, la definición de usuarios, roles, contenido del proyecto, conformación del equipo, definición de metas personales, de rol y de equipo, y gestión de reuniones.*
Formatos de planeación, ejecución y seguimiento	*Formatos de planeación, ejecución y seguimiento. Generación de formatos TSP transversales al proyecto. Entre ellos se encuentran LOGT, TASK, SCHEDULE, WEEK, PIP, CCR, CSR, PEER e INFO.*
Formatos de propósito específico	*Formatos de propósito específico. Generación de formatos TSP vinculados a un estado específico del proyecto. Entre ellos se encuentran SUMP, SUMQ, LOGD, STRAT, ITL.*
Seguimiento y control	*Seguimiento y control. Gestión de ciclos y fases de un proceso de desarrollo de software, controlando el paso de una fase a otra de acuerdo a los criterios de entrada y salida de la metodología TSP.*
Gestión de métricas	*Gestión de métricas. Publicación de resultados cuantitativos y cualitativos personales, de proyecto y de equipo, de acuerdo al historial del trabajo realizado.*
Ajustes y validación de la herramienta	*Ajustes y validación de la herramienta. Se refina el prototipo construido y se efectúan pruebas con base en proyectos de desarrollo de software finalizados, evaluando la efectividad de la herramienta.*

9. DESARROLLO DEL PROYECTO

El producto software obtenido requirió un proceso completo de investigación, análisis, diseño, implementación y pruebas; un desarrollo de software de calidad soportado en la metodología SCRUM. A continuación se darán a conocer los detalles más significativos del presente proyecto de grado: Historias de usuario, modelos de datos, diagramas de estructura y comportamiento, utilización de patrones de diseño (y sus beneficios asociados) y sus respectivas pruebas funcionales y no funcionales.

9.1 HISTORIAS DE USUARIO

Se organizaron y planificaron las diferentes actividades para cada sprint utilizando la herramienta *IceScrum* [76]. Esta herramienta permite definir y ajustar las historias de usuario partiendo del *Área de ensayo* donde se propusieron 23 historias agrupadas en 6 *características*. A lo largo de la ejecución del proyecto se aceptaron, rechazaron y añadieron historias de usuario al área denominada *Pila de producto,* agrupando así un total de 22 historias de usuario de tipo funcional y técnico.

Cada una de las historias de usuario se obtuvo a partir de la investigación teórica-empírica sobre las necesidades de los ingenieros que utilizan la metodología TSP. Inicialmente se desarrolló comprensión teórica de la metodología con base en la literatura (de Humphrey y observaciones de otros autores) para posteriormente retroalimentar el aprendizaje con pruebas empíricas tanto manuales como automatizadas (gracias a los software colaborativos disponibles). Una vez se adquirió dominio de las métricas recolectadas y el proceso cíclico, se concluyeron las necesidades fundamentales en TSP a la fecha, y con base en ello se distribuyeron las historias en cada sprint del proyecto.

9.1.1 Características de agrupación

Para identificar los servicios que debe satisfacer la herramienta software, se definieron las características descritas en la tabla 13, en la que el valor asociado corresponde al nivel de dificultad estimado en un rango de 0 a 10. En la ilustración 8 se pueden observar dichas características clasificadas dentro de la herramienta *IceScrum,* la cual asigna un color a cada una de ellas con el propósito de reconocer rápidamente la pertenencia de cada historia de usuario.

Tabla 13. Características de agrupación de historias de usuario. Fuente autor

Característica	***Descripción***	***Tipo***	***Valor***
Gestión del proyecto	*Gestión de la ejecución temporal del proyecto desde su creación hasta su finalización. Incluye la gestión de fase a fase y de ciclo a ciclo así como elementos varios asociados específicamente a la fase de Lanzamiento de TSP.*	*Funcional*	*8*
Revisiones de progreso	*Gestión de la información de proyecto relacionada directamente con el progreso del mismo.*	*Funcional*	*10*
Control de información	*Recolección de la información del proyecto para registrar nuevos elementos a realizar o cambios sobre alguna parte específica.*	*Funcional*	*10*
Planeación del proyecto	*Captura de la información estimada sobre el proyecto en cuanto a la distribución de tareas y funciones.*	*Funcional*	*10*
Evaluación cualitativa	*Información cualitativa del proyecto a fin de representar los resultados obtenidos en el proceso del proyecto a partir de las métricas recolectadas.*	*Funcional*	*9*
Proceso de ingeniería	*Actividades y procesos asociados al logro del proyecto TSP.*	*Técnico*	*10*



Ilustración 8. Características del proyecto clasificadas, dentro de la herramienta IceScrum, por color. Imagen extraída de la herramienta IceScrum.

9.1.2 Pila de producto

En este apartado se definieron las historias de usuario filtradas con los atributos de nombre, tipo, descripción, característica y valor, tal que el valor, al igual que en las características, corresponde a la estimación del grado de dificultad. En la tabla 14 se puede observar cada una de

las historias de usuario finalmente aceptadas con su respectiva descripción; en el apartado 9.1.4 y el anexo 6 se darán a conocer más detalles de las mismas.

Tabla 14. Historias de usuario con su respectiva descripción Fuente autor

Nombre	***Descripción***
Literatura TSP	*El equipo de desarrollo realiza una lectura del libro de TSP de Humphrey a fin de entender la metodología a profundida*
Historias de usuario	*El equipo de desarrollo refina las historias de usuario para asegurar el buen desarrollo del proyecto.*
Arquitectura de software	*El equipo de desarrollo define la arquitectura de software y la refina constantemente para proveer información de la distriibución de componentes y relaciones del sistema.*
Base de datos	*El equipo de desarrollo diseña la base de datos para garantizar la correcta distribución de la información de proyectos de desarrollo.*
Creación del proyecto	*El usuario crea un proyecto de desarrollo de software para satisfacer, en equipo, una necesidad tecnológica.*
Gestión de fases	*El usuario necesita gestionar la fase actual y pasar a la siguiente fase a fin de continuar con la ejecución del proyecto.*
Gestión de lanzamiento	*El usuario necesita registrar la fase TSP de lanzamiento para designar los roles a los miembros del proyecto,definir metas de rol y de ciclo, y adjuntar el acta del proyecto.*
Gestión de estrategia	*El usuario, como gerente de desarrollo, con ayuda de su equipo, construye la estrategia del proyecto para definir las partes de éste y sus respectivas funciones con estimados de tiempo y tamaño.*
Gestión de tareas	*El usuario necesita registrar las tareas a llevar a cabo en cada parte del proyecto para definir claramente las actividades a realizar y evaluar el diseño a partir de los valores reales y estimados.*
Control de cambios	*El usuario necesita gestionar el control de cambios para asegurar que las partes a unir del proyecto satisfacen las expectativas tanto del cliente como de los integrantes del equipo.*
Formatos complementarios	*El usuario necesita registrar información complementaria al proyecto de desarrollo de software para promover la mejora del equipo en el desarrollo de proyectos.*
Registro de tiempo	*El usuario necesita registrar el tiempo investido en cada una de sus actividades para supervisar el tiempo investido en las tareas que le corresponden.*
Rastreo de defectos	*El usuario necesita dar evidencia de los defectos localizados así como de las medidas para su solución y tiempo requerido.*
Resumen del proyecto	*El usuario necesita gestionar la documentación de resumen para asegurar que el proyecto cuenta con información clara y precisa del proceso llevado a cabo.*
Progreso en calendario	*El usuario necesita registrar el progreso semanal para evaluar el progreso del proyecto y dar muestra detallada y objetiva a los clientes al respecto.*
Documentación del proyecto	*El usuario necesita almacenar la información del proyecto en el cuaderno del proyecto para centralizar la documentación necesaria.*
Control de riesgos	*El usuario necesita registrar los resultados de control de riesgos para asegurar que el módulo implementado o el diseño se alinean al plan de proyecto.*
Rastreo del progreso	*El usuario necesita rastrear el progreso de las tareas para identificar el valor ganado del proyecto.*
Métricas personales	*El usuario necesita visualizar los resultados cualitativos personales en los proyectos en los que ha participado para obtener conclusiones acerca de los logros y fracasos a lo largo del tiempo.*
Métricas de proyecto	*El usuario necesita visualizar los resultados cualitativos del proyecto en el actual ciclo para concluir acerca de los logros y fracasos presentados en la ejecución de éste.*
Pruebas	*El equipo de desarrollo necesita hacer pruebas sobre el producto software resultante a fin de asegurar la calidad del mismo.*

Documentación de software	*El equipo de desarrollo necesita documentar el producto software construído para proveer a los usuarios capacidades sobre el uso de éste y a futuros desarrolladores y analistas de información para la mejora y agregación de componentes.*

9.1.3 Planeación de Sprints

Como se puede observar en el apartado 8.2, el proyecto tenía la finalidad de ejecutar 6 sprints durante los 6 meses de duración estimada del proyecto; un mes por sprint. Sin embargo, la curva de aprendizaje de la gestión de métricas de la metodología TSP, conllevó a un reajuste de calendario que permitiera enfocar la atención en la literatura y así identificar dependencias, disponibilidad de cambios, dominio sobre los formatos y validaciones de datos. Por tanto, cada sprint se redujo en 5 días aproximadamente para dar apertura a otro sprint, en el que además de obtener dominio sobre la metodología, también se tuvo dominio sobre la tecnología.

En la ilustración 9 se puede observar la distribución de las historias de usuario a lo largo de los 7 sprints. Cabe mencionar que todas las historias de usuario del primer sprint se ejecutaron en menor grado en los demás sprints como elementos en paralelo para ajustar los planes con el desarrollo.

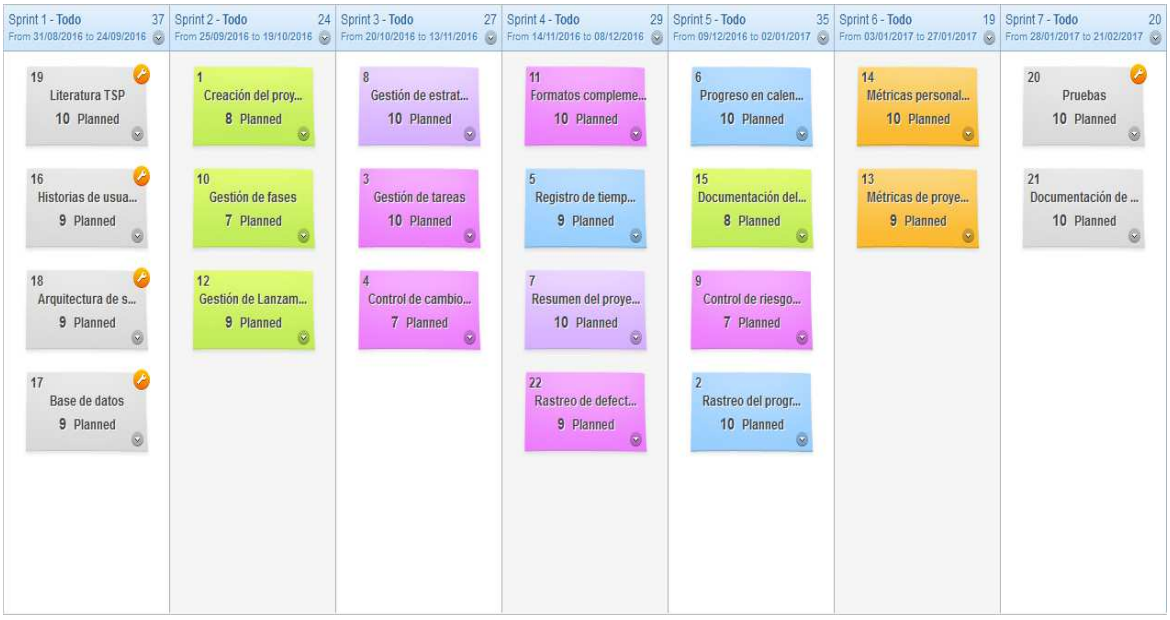


Ilustración 9. Historias de usuario en su respectivo sprint. Imagen capturada de la herramienta IceScrum

Durante la ejecución de los sprints se llevaron a cabo reuniones semanales entre los integrantes del equipo. De igual forma se hicieron las entregas pertinentes de los avances del

producto software dentro de los tiempos estimados donde se consideró que el resultado era el esperado, sin embargo todas las funcionalidades fueron completadas con validaciones posteriores donde se demostraron elementos carentes o fallidos debido a la dependencia de valores capturados entre formatos de TSP, donde ciertas características irregulares no eran tan evidentes.

9.1.4 Documentación de historias de usuario

Con base en la documentación de historias de usuario presentada en [107], se consolida en tarjetas la información del apartado 9.1.2 junto a los siguientes atributos.

- Código. Identificador de tarjeta.
- HU_Relacionadas. Dependencias y relaciones entre historias de usuarios.
- Actor. Rol encargado de llevar a cabo la funcionalidad de la historia de usuario.
- Criterios de aceptación. Requisitos mínimos que debe cumplir para que la historia de usuario sea aprobada por el cliente.

Tabla 15. Historia de usuario de arquitectura de software. Fuente autor

Historia de usuario			
Código	HU_01	**Nombre**	Arquitectura de software
Tipo	Técnico	**Complejidad**	9
Actor	Equipo de desarrollo	**HU_Relacionadas**	Todas las historias de usuario
Característica	Proceso de ingeniería		
Descripción	El equipo de desarrollo define la arquitectura de software y la refina constantemente para proveer información de la distribución de los componentes y relaciones del sistema.		
Criterios de aceptación			
Condición		**Resultado**	
El equipo de desarrollo evalúa y guarda la arquitectura con base en todo el proceso analizado a la fecha.		Arquitectura de software con todos los componentes del sistema relacionados entre sí.	

Tabla 16. Historia de usuario de base de datos. Fuente autor

Historia de usuario			
Código	HU_02	**Nombre**	Base de datos
Tipo	Técnico	**Complejidad**	9
Actor	Equipo de desarrollo	**HU_Relacionadas**	Todas las historias de usuario
Característica	Proceso de ingeniería		
Descripción	El equipo de desarrollo diseña la base de datos para garantizar la correcta distribución de la información de proyectos de desarrollo.		
Criterios de aceptación			
Condición		**Resultado**	
El equipo de desarrollo evalúa y guarda el diagrama de base de datos con todos los parámetros estimados a la fecha.		Modelo de base de datos construido y transformado para incorporarse a un motor de base de datos.	

Tabla 17. Historia de usuario sobre la refinación de las historias. Fuente autor

Historia de usuario			
Código	HU_04	**Nombre**	Literatura TSP
Tipo	Técnico	**Complejidad**	10
Actor	Equipo de desarrollo	**HU_Relacionadas**	Ninguna
Característica	Proceso de ingeniería		
Descripción	El equipo de desarrollo realiza una lectura del libro de TSP de Humphrey a fin de entender la metodología a profundida.		
Criterios de aceptación			
Condición		**Resultado**	
Disponer del libro de la metodología TSP propuesta por Humphrey.		Comprensión de la metodología, sus características e implementación.	

Tabla 18. Historia de usuario de la literatura TSP. Fuente autor

Historia de usuario			
Código	HU_03	**Nombre**	Historias de usuario
Tipo	Técnico	**Complejidad**	9
Actor	Equipo de desarrollo	**HU_Relacionadas**	Todas las historias de usuario
Característica	Proceso de ingeniería		
Descripción	El equipo de desarrollo refina las historias de usuario para asegurar el buen desarrollo del proyecto.		
Criterios de aceptación			
Condición		**Resultado**	
El equipo de desarrollo evalúa y guarda la información de los sprints del proyecto de acuerdo con la evaluación realizada a la fecha.		Historias de usuario actualizadas. Historias de usuario nuevas, eliminadas y modificadas.	

Tabla 19. Historia de usuario de creación de proyecto. Fuente autor

Historia de usuario			
Código	HU_05	**Nombre**	Creación del proyecto
Tipo	Funcional	**Complejidad**	8
Actor	Usuario	**HU_Relacionadas**	Ninguna
Característica	Gestión del proyecto		
Descripción	El usuario crea un proyecto de desarrollo de software para satisfacer, en equipo, una necesidad tecnológica.		
Criterios de aceptación			
Condición		**Resultado**	
El usuario debe asociarse en un equipo de desarrollo, indicar el número de ciclos estimados del proyecto, fecha estimada de finalización del proyecto, especificar la métrica de tamaño en implementación, especificar el día de referencia a la semana para evaluaciones de progreso y asociar un instructor.		Usuarios asociados al proyecto.	

Tabla 20. Historia de usuario de gestión de lanzamiento. Fuente autor

Historia de usuario			
Código	HU_06	**Nombre**	Gestión de lanzamiento
Tipo	Funcional	**Complejidad**	9
Actor	Usuario	**HU_Relacionadas**	HU_05
Característica	Gestión del proyecto		
Descripción	El usuario necesita registrar la fase TSP de lanzamiento para designar los roles a los miembros del proyecto, definir metas individuales, de rol y de proyecto, y adjuntar el acta de proyecto.		

Criterios de aceptación	
Condición	**Resultado**
El proyecto debe estar en fase de 'Lanzamiento'. Los roles deben ser asignados a los integrantes del proyecto y las metas deben ser definidas.	Los usuarios son vinculados al proyecto y las metas son asignadas al ciclo.

Tabla 21. Historia de usuario de gestión de estrategia. Fuente autor

Historia de usuario			
Código	HU_07	**Nombre**	Gestión de estrategia
Tipo	Funcional	**Complejidad**	10
Actor	Gerente de desarrollo	**HU_Relacionadas**	HU_05, HU_06
Característica	Planeación del proyecto		
Descripción	El usuario, como gerente de desarrollo, con ayuda de su equipo, construye la estrategia del proyecto para definir las partes de éste y sus respectivas funciones con estimados de tiempo y tamaño.		
Criterios de aceptación			
Condición		**Resultado**	
El proyecto debe estar en fase de 'Estrategia'. Cada parte debe asignársele por lo menos una función y, a cada función, asignarle el estimado de tamaño y tiempo para cada ciclo del proyecto.		Formato STRAT publicado satisfactoriamente. Se da comienzo a la siguiente fase del proyecto (fase actual del proyecto se actualiza a 'Planeación').	

Tabla 22. Historia de usuario de gestión de fases. Fuente autor

Historia de usuario			
Código	HU_08	**Nombre**	Gestión de fases
Tipo	Funcional	**Complejidad**	7
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_06
Característica	Gestión del proyecto		
Descripción	El usuario necesita gestionar la actual fase y pasar a la siguiente fase para continuar con la ejecución del proyecto.		
Criterios de aceptación			
Condición		**Resultado**	
Se cumplen todos los criterios de salida de la fase; los validados automáticamente y aceptados en el momento.		El proyecto cambia de fase de 'Lanzamiento' a 'Estrategia', de 'Estrategia' a 'Planeación', de 'Planeación' a 'Requerimientos', de 'Requerimientos' a 'Diseño', de 'Diseño' a 'Implementación', de 'Implementación' a 'Pruebas', de 'Pruebas' a 'Postmortem' y de 'Postmortem' a 'Lanzamiento'.	

Tabla 23. Historia de usuario de control de riesgos. Fuente autor

Historia de usuario			
Código	HU_09	**Nombre**	Control de riesgos
Tipo	Funcional	**Complejidad**	7
Actor	Usuarios	**HU_Relacionadas**	HU_05, HU_07, HU_08
Característica	Control de información		
Descripción	El usuario necesita registrar los resultados de control de riesgos para asegurar que el módulo implementado o el diseño se alinean al plan del proyecto.		
Criterios de aceptación			
Condición		**Resultado**	
Partes del sistema registradas.		Registro ITL, de riesgo/problema, generado satisfactoriamente.	

Tabla 24. Historia de usuario de métricas de proyecto. Fuente autor

Historia de usuario			
Código	HU_10	**Nombre**	Métricas de proyecto
Tipo	Funcional	**Complejidad**	9
Actor	Usuario	**HU_Relacionadas**	
Característica	Evaluación cualitativa		
Descripción	El usuario necesita visualizar los resultados cualitativos del proyecto para concluir acerca de los logros y fracasos presentados en la ejecución de éste.		
Criterios de aceptación			
Condición		**Resultado**	
Proyecto iniciado y solicitud de métrica a representar cualitativamente seleccionada.		Métrica representada gráficamente con base en los datos recolectados a la fecha.	

Tabla 25. Historia de usuario de métricas personales. Fuente autor

Historia de usuario			
Código	HU_11	**Nombre**	Métricas personales
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_13, HU_14, HU_16, HU_17, HU_18, HU_19
Característica	Evaluación cualitativa		
Descripción	El usuario necesita visualizar los resultados cualitativos y cuantitativos de un conjunto de proyectos en los que ha participado para concluir acerca de los logros y fracasos presentados en total.		
Criterios de aceptación			
Condición		**Resultado**	
El usuario debe contar con al menos un proyecto.		Resumen de valores en tiempo y tamaño listados con gráficas asociadas.	

Tabla 26. Historia de usuario de registro de tiempo. Fuente autor

Historia de usuario			
Código	HU_12	**Nombre**	Registro de tiempo
Tipo	Funcional	**Complejidad**	9
Actor	Usuario	**HU_Relacionadas**	HU_13, HU_14, HU_16, HU_17, HU_18, HU_19
Característica	Revisiones de progreso		
Descripción	El usuario necesita registrar el tiempo invertido en cada una de sus actividades para supervisar el tiempo invertido en las tareas que le corresponden.		
Criterios de aceptación			
Condición		**Resultado**	
Fase actual igual o posterior a la fase de planeación. Requiere la existencia de las tareas definidas para el ciclo.		Registro de actividad completado y actualizado en el formato LOGT, especificando la parte que se llevó a cabo y si fue completada o no la tarea.	

Tabla 27. Historia de usuario de gestión de tareas. Fuente autor

Historia de usuario			
Código	HU_13	**Nombre**	Gestión de tareas
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08, HU_12
Característica	Control de información		

Descripción	El usuario necesita registrar las tareas a llevar a cabo en cada parte del proyecto para definir claramente las actividades a realizar y evaluar el desempeño a partir de valores reales y estimados.
Criterios de aceptación	
Condición	**Resultado**
Fase actual igual a 'Planeación'.	Formato TASK creado satisfactoriamente.

Tabla 28. Historia de usuario de rastreo del progreso. Fuente autor

Historia de usuario			
Código	HU_14	**Nombre**	Rastreo del progreso
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08, HU_12, HU_17
Característica	Revisiones de progreso		
Descripción	El usuario necesita rastrear el progreso de las tareas para identificar el valor ganado del proyecto.		
Criterios de aceptación			
Condición		**Resultado**	
Existencia del formato TASK; fase de 'Planeación' o posterior.		Nuevo registro del progreso de la semana en el formato SCHEDULE.	

Tabla 29. Historia de usuario de progreso en calendario. Fuente autor

Historia de usuario			
Código	HU_15	**Nombre**	Progreso en calendario
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08, HU_12, HU_17
Característica	Revisiones de progreso		
Descripción	El usuario necesita registrar el progreso semanal para evaluar el progreso del proyecto y dar muestra detallada y objetiva a los clientes al respecto.		
Criterios de aceptación			
Condición		**Resultado**	
Existencia del formato TASK; fase de 'Planeación' o posterior.		Reporte WEEK generado satisfactoriamente.	

Tabla 30. Historia de usuario de formatos complementarios. Fuente autor

Historia de usuario			
Código	HU_16	**Nombre**	Formatos complementarios
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08
Característica	Control de información		
Descripción	El usuario necesita registrar elementos complementarios al proyecto de desarrollo de software para promover la mejora del equipo en el desarrollo de proyectos.		
Criterios de aceptación			
Condición		**Resultado**	
Fase posterior a la fase de 'Estrategia'		Generación de formato TSP complementario (INFO, PIP o PEER)	

Tabla 31. Historia de usuario de rastreo de defectos. Fuente autor

Historia de usuario			
Código	HU_17	**Nombre**	Rastreo de defectos
Tipo	Funcional	**Complejidad**	9
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08
Característica	Control de información		
Descripción	El usuario necesita dar evidencia de los defectos localizados así como de las medidas para su solución y tiempo requerido.		
Criterios de aceptación			
Condición		**Resultado**	
Fase posterior a la fase de 'Estrategia'		Nuevo registro de defecto en formato LOGD guardado satisfactoriamente.	

Tabla 32. Historia de usuario de control de cambios. Fuente autor

Historia de usuario			
Código	HU_18	**Nombre**	Control de cambios
Tipo	Funcional	**Complejidad**	7
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_07, HU_08
Característica	Control de información		
Descripción	El usuario necesita gestionar el control de cambios para asegurar que las partes a unir del proyecto satisfacen las expectativas tanto del cliente como de los integrantes del equipo.		
Criterios de aceptación			
Condición		**Resultado**	
Fase posterior a la fase de 'Estrategia'		Formato CCR generado satisfactoriamente.	

Tabla 33. Historia de usuario de resumen del proyecto. Fuente autor

Historia de usuario			
Código	HU_19	**Nombre**	Resumen del proyecto
Tipo	Funcional	**Complejidad**	10
Actor	Usuario	**HU_Relacionadas**	HU_05, HU_09, HU_12, HU_13, HU_14, HU_15, HU_16, HU_18
Característica	Planeación del proyecto		
Descripción	El usuario necesita generar la documentación de resumen para asegurar que el proyecto cuenta con información clara y precisa del proceso llevado a cabo.		
Criterios de aceptación			
Condición		**Resultado**	
Estimación de resumen diligenciada		Formatos SUMP y SUMQ actualizados.	

Tabla 34. Historia de usuario de documentación del proyecto. Fuente autor

Historia de usuario			
Código	HU_20	**Nombre**	Documentación del proyecto
Tipo	Funcional	**Complejidad**	8
Actor	Usuario	**HU_Relacionadas**	HU_05
Característica	Gestión del proyecto		

Descripción	El usuario necesita almacenar la información del proyecto en el cuaderno del proyecto para centralizar la documentación necesaria.
Criterios de aceptación	
Condición	**Resultado**
Encontrarse en fase de Estrategia o posterior; deben existir los roles de ciclo asignados.	Nuevo directorio y/o archivo adjunto al "Cuaderno de proyecto".

Tabla 35. Historia de usuario de pruebas. Fuente autor

Historia de usuario			
Código	HU_21	**Nombre**	Pruebas
Tipo	Técnico	**Complejidad**	10
Actor	Equipo de desarrollo	**HU_Relacionadas**	Todas las historias de usuario
Características	Proceso de ingeniería		
Descripción	El equipo de desarrollo necesita hacer pruebas sobre el producto software resultante a fin de asegurar la calidad del mismo.		
Criterios de aceptación			
Condición		**Resultado**	
Subproducto terminado		Errores identificados y corregidos.	

Tabla 36. Historia de usuario de documentación de software. Fuente autor

Historia de usuario			
Código	HU_22	**Nombre**	Documentación de software
Tipo	Técnico	**Complejidad**	10
Actor	Equipo de desarrollo	**HU_Relacionadas**	Todas las historias de usuario
Característica	Proceso de ingeniería		
Descripción	El equipo de desarrollo necesita documentar el producto software construído para proveer a los usuarios capacidades sobre el uso de éste y a futuros desarrolladores y analistas de información para la mejora y agregación de componentes.		
Criterios de aceptación			
Condición		**Resultado**	
Subproductos software terminados		Información técnica y de usuario del subproducto.	

9.1.5 Dependencias entre historias de usuario

Para una adecuada gestión de cambios se definió la matriz de dependencias entre historias de usuario, lo que brindó mayor eficacia en el control indirecto y directo de medidas para la recolección de métricas a través de los formatos TSP. Esta información fue obtenida rápidamente gracias a la *matriz de trazabilidad* [107] mostrada en la tabla 37. A partir de ella se deduce que las historias de usuario más dependientes son: HU_10 Métricas de proyecto, HU_11 Métricas personales, HU_15 Progreso en calendario y HU_19 Resumen del proyecto.

Tabla 37. Matriz de trazabilidad para las historias de usuario funcionales. Fuente autor

		Historias de usuario dependientes															
		HU_05	*HU_06*	*HU_07*	*HU_08*	*HU_09*	*HU_10*	*HU_11*	*HU_12*	*HU_13*	*HU_14*	*HU_15*	*HU_16*	*HU_17*	*HU_18*	*HU_19*	*HU_20*
Historias de usuario independientes	*HU_05*		*x*	*x*	*x*	*x*	*x*		*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*
	HU_06			*x*	*x*												
	HU_07					*x*			*x*	*x*	*x*	*x*	*x*	*x*	*x*		
	HU_08					*x*			*x*	*x*	*x*	*x*	*x*	*x*	*x*		
	HU_09															*x*	
	HU_10																
	HU_11																
	HU_12									*x*	*x*	*x*				*x*	
	HU_13						*x*	*x*	*x*							*x*	
	HU_14						*x*	*x*				*x*				*x*	
	HU_15															*x*	
	HU_16						*x*	*x*								*x*	
	HU_17						*x*	*x*			*x*	*x*				*x*	
	HU_18						*x*	*x*								*x*	
	HU_19						*x*	*x*									
	HU_20																

9.2 GESTIÓN DE DATOS TSP

9.2.1 Mediciones, indicadores y métricas

La gestión de las métricas nace de la medición, tanto del proceso como del producto, los cuales son recolectados en algunos de los formatos de la metodología TSP. De igual forma, estas mediciones brindan información suficiente para establecer indicadores y con ellos obtener las métricas asociadas. A continuación se especifican algunos indicadores y métricas, únicamente pertenecientes a los formatos SUMP y SUMQ, que brindan mayor información del proyecto.

$$Código\ nuevo\ y\ cambiado\ total = Código\ modificado + Código\ añadido$$

$$Código\ total = Código\ nuevo\ y\ cambiado\ total + Código\ base - Código\ eliminado - Código\ modificado + Código\ reutilizado$$

$$Tiempo\ en\ fase\ (horas) = \sum_{Gerencia\ y\ miscelanea}^{Postmortem} t\ iempo$$

$$Defectos\ inyectados = \sum_{Planeación\ y\ estrategia}^{Prueba\ de\ sistema} cantidad$$

$$Defectos\ removidos = \sum_{Planeación\ y\ estrategia}^{Prueba\ de\ sistema} cantidad$$

$$Prueba\ de\ aceptación_{Defectos\ removidos} = (Defectos\ inyectados - Defectos\ removidos) * Prueba\ de\ aceptación_{(Rendimientos\ de\ fase)}$$

$$Vida\ de\ producto_{Defectos\ removidos} = Defectos\ inyectados - Defectos\ removidos + Prueba\ de\ aceptación$$

$$\frac{Código}{hora} = \frac{Código\ nuevo\ y\ cambiado\ total}{Tiempo\ en\ fase\ (horas)}$$

$$\frac{Costo}{Desempeño} = \frac{Tiempo\ en\ fase\ (horas)_{Plan}}{Tiempo\ en\ fase\ (horas)_{Real}}$$

$$\%\ Reuso = \frac{Código\ reutilizado}{Código\ total}$$

$$Inspección\ de\ requerimientos_{Defectos/Página} = \frac{Inspección\ de\ requerimientos_{Defectos\ removidos}}{Páginas\ de\ requerimientos\ SRS\ _{Tamaño\ del\ producto}}$$

$$Inspección\ de\ diseño\ de\ alto\ nivel = \frac{Inspección\ de\ diseño\ de\ alto\ nivel\ (HLD)_{Defectos\ removidos}}{Páginas\ de\ diseño\ de\ alto\ nivel\ (SDS)_{Tamaño\ del\ producto}}$$

$$Compilación_{\frac{Defectos}{Código}} = \frac{Compilación_{defectos\ removidos}}{Código\ nuevo\ y\ cambiado\ total}$$

$$Desarrollo\ total_{\frac{Defectos}{Código}} = \frac{Defectos\ removidos}{Código\ nuevo\ y\ cambiado\ total}$$

$$Prueba\ de\ aceptación_{\frac{Defectos}{Código}} = \frac{Prueba\ de\ aceptación_{Defectos\ removidos}}{Código\ nuevo\ y\ cambiado\ total}$$

$$Vida\ de\ producto_{\frac{Defectos}{Código}} = \frac{Vida\ de\ producto_{Defectos\ removidos}}{Código\ nuevo\ y\ cambiado\ total}$$

$$\frac{Defectos}{Código} = (Defectos\ removidos + Prueba\ de\ aceptación_{Defectos\ removidos} + Vida\ de\ producto_{Defectos\ removidos})$$

Gran parte de la calidad del producto software prototipo, recae sobre la exactitud de los valores obtenidos durante la ejecución de proyectos que son gestionados mediante la misma herramienta. Deben tenerse en cuenta las unidades de medición de modo que los valores sean convertidos con la mayor precisión posible y por ende provean información real, y en consecuencia, útil para quienes hagan uso de la herramienta.

9.2.2 Número de semanas de proyecto

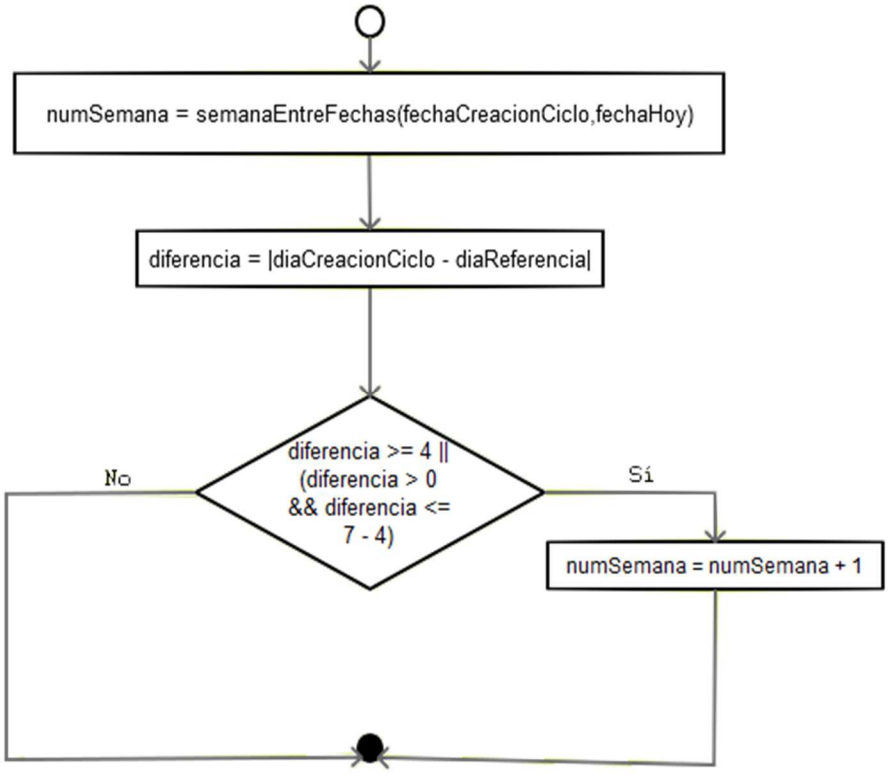


Ilustración 10.Algoritmo para calcular el número de semana de TSP. Fuente autor

La gestión del tiempo transcurrido en el proyecto es fundamental en el proceso de la metodología, y una vez evaluado el progreso semanal se identificó la diferencia entre el número real de semanas transcurridas y el número de semanas basadas en el día de referencia del proyecto. Esto sucede porque la diferencia de días entre la fecha de creación del proyecto y la fecha próxima del día de referencia no es lo suficientemente grande para considerarse una semana completa de trabajo, de modo que fue necesario estandarizar el número de semana con base en dicha referencia, lo que finalmente se denominó como *Número de semana TSP*, calculado mediante el algoritmo mostrado en la ilustración 10.

9.3 EVALUACIÓN DEL DISEÑO

9.3.1 Prototipos del prototipo

Con base en la metodología utilizada, se dio comienzo a la construcción de prototipos que permitieran orientar el desarrollo del producto hacia los logros esperados: exactitud, automatización y amigabilidad. Esta exploración tanto tecnológica como de diseño brindó un total de dos prototipos que serán explicados brevemente a continuación.

El primer prototipo se centró en ofrecer una interfaz capaz de soportar las necesidades del usuario como la gestión de métricas con base en los formatos de la metodología TSP, proveer un acceso simple al registro de las tareas y gestionar todos los proyectos en los que se ve involucrado el usuario. Gracias al primer prototipo se tuvo conocimiento sobre las limitaciones tecnológicas (como la desincronización de un reloj en tiempo real) y el modo de gestionar los registros de tiempo.

El segundo prototipo trató de resolver algunos detalles asociados a la amigabilidad, sin embargo se enfocó en mayor medida en la gestión de la información recolectada a partir de formatos y la generación de componentes para facilitar el desarrollo. Gracias al segundo prototipo se adquirió más experticia tecnológica y se tuvo una idea mucho más clara de las expectativas del software.

En la ilustración 11 se puede observar el diseño acordado para el front-end del producto software. La distribución muestra un menú superior con la intención de ofrecer información de los

proyectos, accesibilidad a la documentación y cierre de la sesión. Se sigue con el recuadro del proyecto; una ficha informativa del proyecto utilizada para la sesión y que permite cambiar a otro

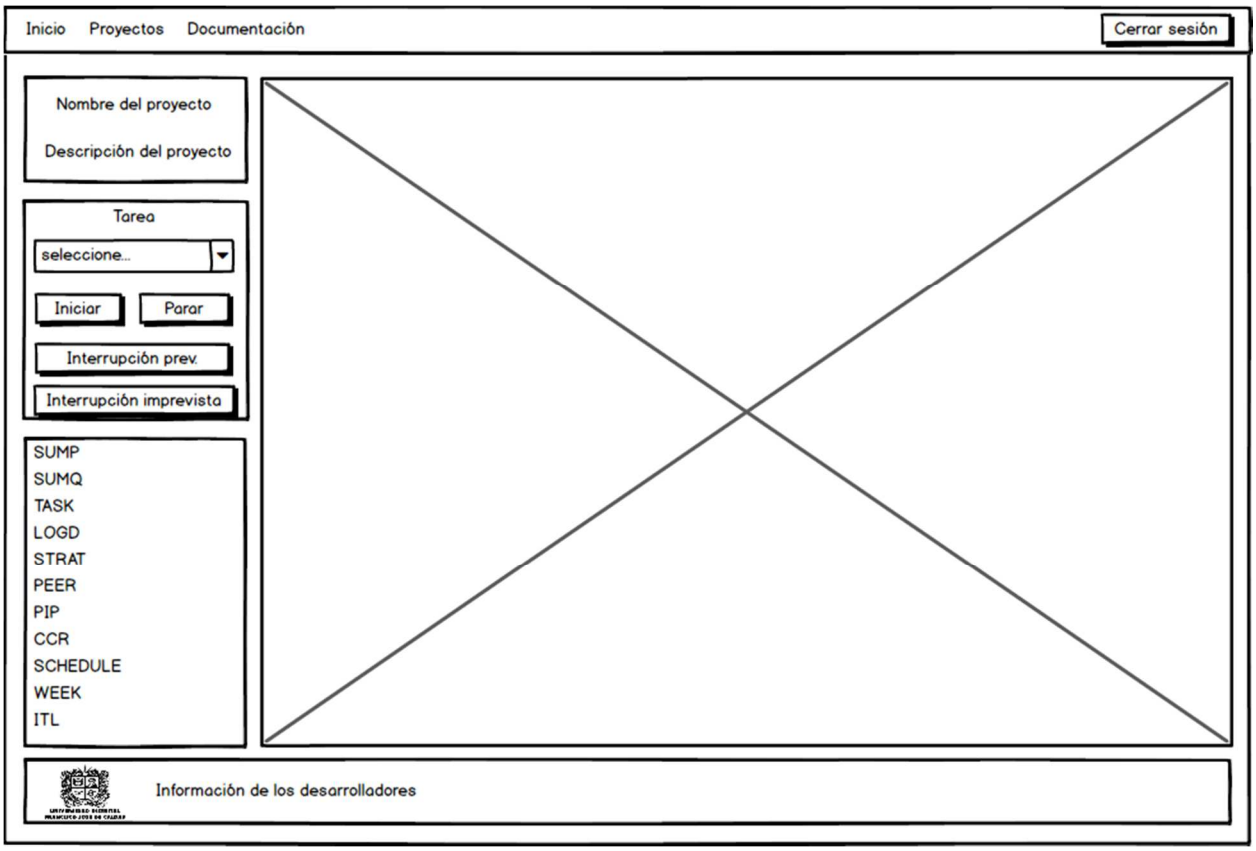


Ilustración 11. Interfaz piloto para el proyecto. Fuente autor

proyecto al que se encuentra vinculado el usuario. Posteriormente se localiza el recuadro de tareas para registrar los tiempos dedicados por el usuario en el proyecto durante el actual ciclo de desarrollo. Los demás formatos se ofrecen al usuario mediante enlaces y finalmente se expone información acerca de la entidad y los miembros involucrados en el desarrollo. Cabe mencionar que el recuadro marcado con una *equis* (*X*) es el espacio de trabajo del usuario para almacenar la información del proyecto o hacer todo tipo de gestiones asociadas a la metodología y al software.

9.3.2 Prototipo final

A lo largo del desarrollo del prototipo final, se hizo más evidente la necesidad de cambiar el diseño base para brindar constante accesibilidad, separar contextos y por ende, ofrecer mayor amigabilidad al usuario. El diseño final se puede observar en la ilustración 12, en la que se evidencia que el menú superior ha sido separado en el contexto de la sesión y el contexto de los proyectos del usuario en sesión. El menú de la izquierda se separa en tres módulos: gestión de

fases, gestión de tiempo y gestión documental; componentes que deben brindarse en todo momento para satisfacer las variadas necesidades del usuario que hace uso de la metodología TSP.

El menú de gestión pretende desplegar la información de la fase actual como modo de guía en un menú con varias opciones disponibles. El menú de gestión de tiempo especifica el rol y la tarea que se va a llevar a cabo, actualizando su contenido con los tiempos marcados en la ejecución de una tarea. En cuanto al menú de la documentación, se provee de un directorio bastante claro sobre el proyecto que despliega su contenido con base en el estado actual del proyecto.

En la ilustración 12 también es rápidamente notable el nuevo componente alojado en la parte superior del contenido general de la página. Este componente provee información clara y amigable al usuario acerca del proyecto, mostrando el logo de éste, su información general y qué roles tiene el usuario en sesión para el ciclo actual. Adicionalmente, este componente posee un ícono al final que permite efectuar la transacción para pasar a la siguiente fase o ciclo en el desarrollo de la metodología TSP.

Otro componente, agregado en el prototipo final, es el menú de ayuda y acceso rápido representado por el ícono *?* donde se provee acceso rápido y amigable a los formatos de la

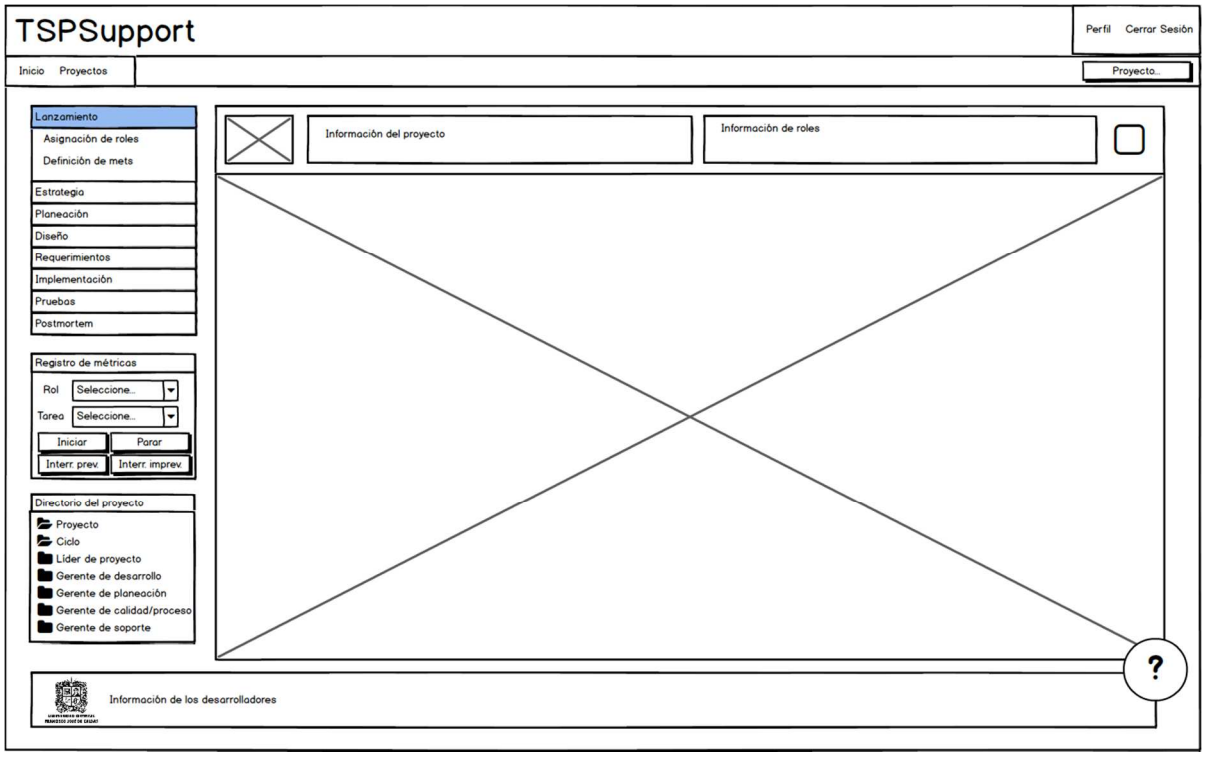


Ilustración 12. Interfaz definitiva para el proyecto. Fuente autor

metodología TSP (ordenados alfabéticamente) y tutoriales para capacitación del uso de la herramienta.

Junto a estos diseños se consideró brindar cuatro modales para la carga de documentos, para la gestión de tiempo, para la ayuda y el acceso rápido, y para el cambio de proyecto. Desde el principio la tecnología *Materialize* fue candidata para la implementación de estos componentes dado su nivel de personalización, amigabilidad y despliegue de múltiples funcionalidades sobre una misma interfaz general.

Otro aspecto importante que se consideró respecto a la amigabilidad del software fue el uso de un color representativo por cada fase de la metodología TSP; de esta forma, los usuarios pueden identificar cada vez más rápido y en forma inmediata, la fase sobre la que están actualmente. En la ilustración 13 se muestra el color asignado para cada fase.

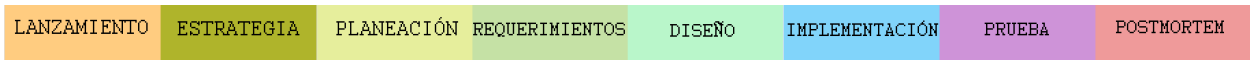


Ilustración 13. Colores asignados para cada fase TSP en la herramienta. Fuente autor.

9.4 MODELADO DE REQUERIMIENTOS

En este apartado se describe la incorporación del modelo ágil utilizado en [107], al proyecto, a fin de asegurar la calidad del producto software a la vez que se desprende de la documentación de las metodologías formales.

Se llevaron a cabo los diagramas de casos de uso asociados a las historias expuestas en el apartado 9.1, el diagrama de formatos para especificar dependencias de valores recolectados, el diagrama de paquetes y de clase para conocer la distribución del producto software, diagramas de secuencia para comprender la colaboración entre las entidades para almacenar y consultar información del proyecto dentro de los formatos y el proyecto en general, el diagrama de despliegue para conocer las tecnologías utilizadas y un diagrama de estado para representar el paso de una fase a otra durante la ejecución del proyecto.

9.4.1 Modelado UML ágil

El diseño de la solución requiere comprender *qué* servicios debe satisfacer la herramienta software, lo que provoca la necesidad de diseñar historias de usuario capaces de plasmar estas

ideas. Sin embargo, según [107], los casos de uso proporcionan una mayor comprensión sobre las necesidades del cliente, y es por esta razón que se construyó el diagrama de contexto y los respectivos diagramas de caso de uso del siguiente nivel.

El diagrama de contexto mostrado en la ilustración 14 evidencia que los usuarios de la aplicación deben tomar posesión de alguno de los roles de la metodología TSP para llevar a cabo la gestión del proyecto y con ello la gestión de métricas del mismo. Cada caso de uso de este diagrama representa un módulo a satisfacer en correspondencia a las historias de usuario descritas anteriormente, agrupándolas de acuerdo a las necesidades generales de los clientes.

En el anexo 1 se pueden observar los diagramas de caso de uso que especifican los casos de uso del diagrama de contexto, detallando así los requerimientos a satisfacer hasta el siguiente nivel para no extender la documentación de la metodología ágil.

9.4.2 Modelo de formatos

Dado que las complejas dependencias entre los valores correspondientes a los distintos formatos no son claramente especificados en la literatura de TSP, se realizó una evaluación minuciosa que conllevó a la generación del grafo mostrado en la ilustración 15, tal que, cada relación representa la inyección de datos de un formato a otro. Un ejemplo es la dependencia del formato TASK por el formato STRAT en cuanto a las partes sobre las cuales se asignarán las

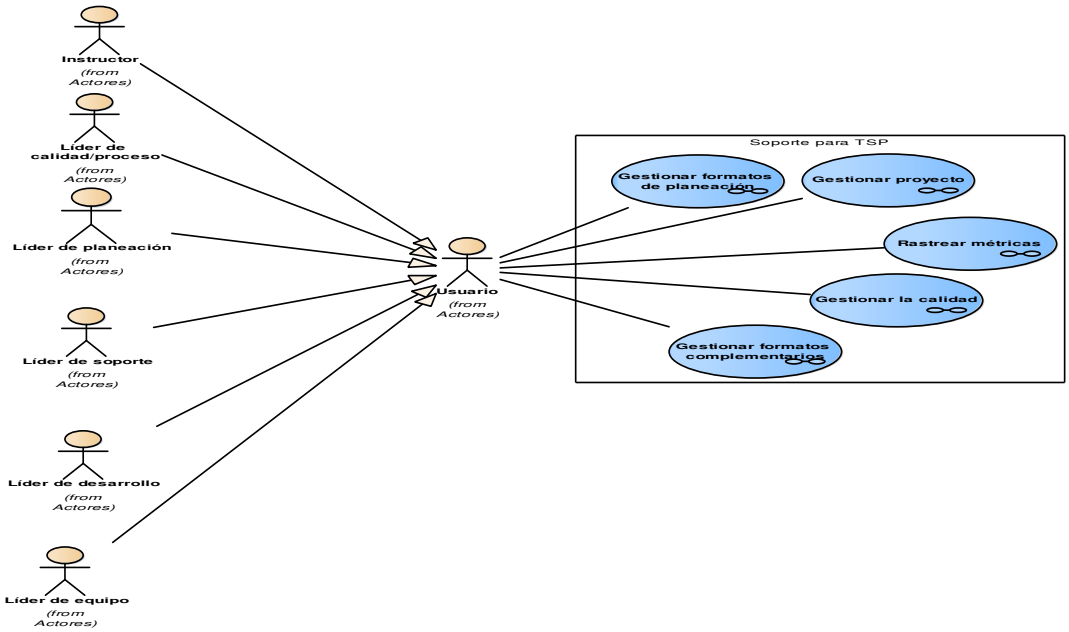


Ilustración 14. Diagrama de contexto. Fuente autor.

tareas, lo que en la relación del grafo se interpretaría como: “el formato STRAT alimenta los datos del formato TASK”.

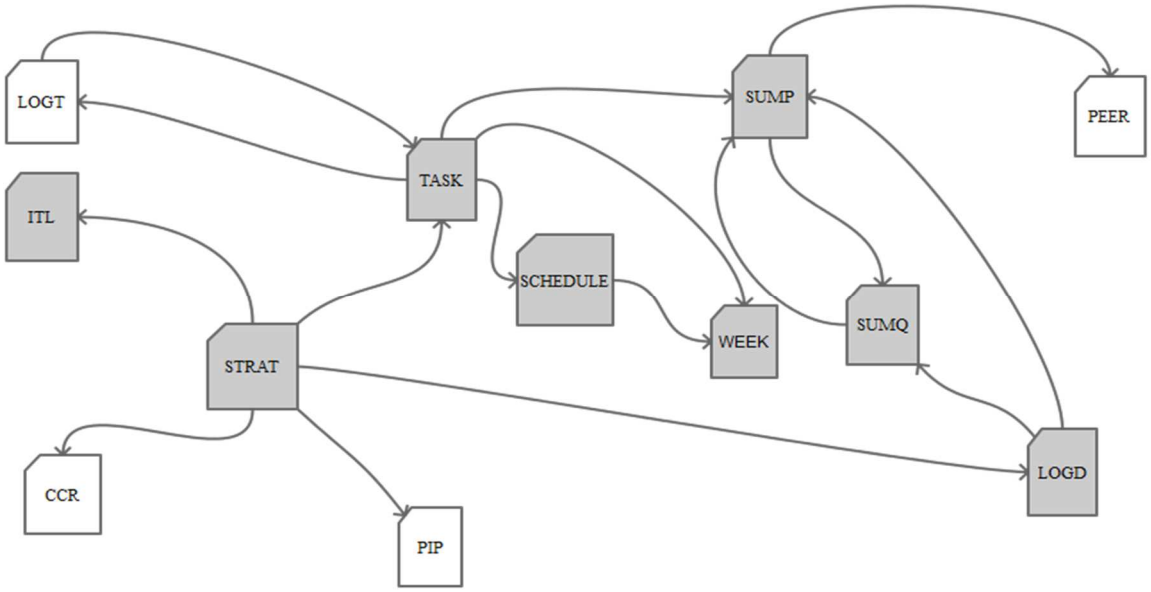


Ilustración 15. Grafo de inyección de información. Realizado con la herramienta Pencil. Fuente autor

Otra característica a destacar en el grafo de la ilustración 15 es el color de cada nodo. Los nodos de color gris corresponden a los formatos diligenciados de manera compartida durante un ciclo de desarrollo de la metodología TSP. Los nodos de color blanco, por el contrario, corresponden a los formatos diligenciados en forma personal, pues la información sólo depende de las características del integrante del equipo que los diligencia.

9.4.3 Diagrama de paquetes

El producto software se compone principalmente de 5 paquetes que pretenden alinearse con la arquitectura Modelo-Vista-Controlador tal como se muestra en la ilustración 16. Cada paquete cuenta con un prefijo correspondiente a una letra del abecedario para establecer un orden secuencial de interacción de componentes con el objetivo de facilitar su interpretación.

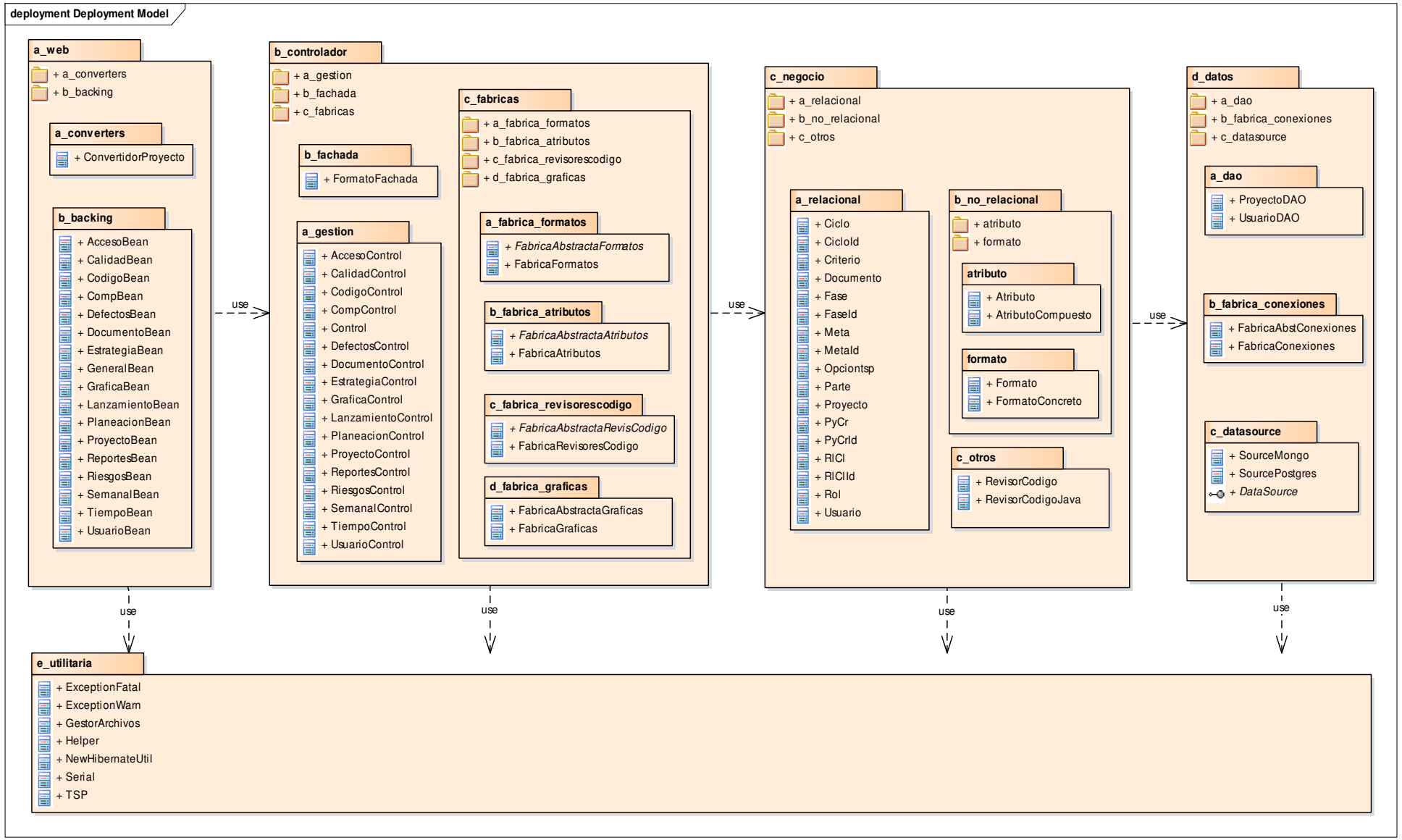


Ilustración 16. Diagrama de paquetes de todo el sistema. Realizado con la herramienta Enterprise Architect. Fuente autor

9.4.4 Diagrama de clases

Dada la cantidad de clases y relaciones, en este apartado se darán a conocer los elementos de mayor relevancia, con especial hincapié en los patrones de diseño utilizados: *Facade*, *Factory*, *Composition* y *DAO*.

9.4.4.1 Paquete Web

Las clases mostradas en la ilustración 17 tienen como finalidad la gestión de las solicitudes y necesidades de los clientes incluyendo la gestión de componentes web. Cada una de estas clases responde al patrón de diseño *Facade* debido a que minimizan la comunicación y dependencias entre las necesidades del cliente y la resolución de éstas.

Todas las clases orientadas a la gestión web poseen una generalización compartida para reutilizar código independiente del contexto del proyecto.

9.4.4.2 Paquete Gestión

Las clases en gestión corresponden al Controlador en la arquitectura MVC. La tabla 38 describe el propósito de cada controlador del prototipo.

Tabla 38. Descripción de controladores del prototipo. Fuente autor

Clase	Descripción
AccesoControl	Permite o niega el acceso del usuario si éste se ha logueado o no.
CalidadControl	Gestiona las métricas recolectadas en los formatos SUMP y SUMQ.
CodigoControl	Herramienta de apoyo a la medición del tamaño de código.
DefectosControl	Gestiona la información de defectos en el formato LOGD.
DocumentoControl	Gestiona el directorio del proyecto en forma centralizada.
EstrategiaControl	Gestiona la planeación de partes y funciones en el formato STRAT.
GraficaControl	Construye las evaluaciones cualitativas de las métricas recolectadas.
LanzamientoControl	Gestiona el proceso de selección de roles y definición de metas de ciclo.
PlaneacionControl	Gestiona la planeación de tareas del ciclo de proyecto en el formato TASK.
ProyectoControl	Gestiona las actividades de proyecto como el paso de fase a fase, creación y carga del proyecto e información relacionada.
ReporteControl	Exporta los formatos a hojas de cálculo.
RiesgoControl	Gestiona los riesgos de proyecto en el formato ITL.
SemanalControl	Controla las métricas semanales en los formatos SCHEDULE y WEEK.
TiempoControl	Gestiona el tiempo dedicado a las tareas asignadas a los integrantes del proyecto.
UsuarioControl	Gestiona las cuentas de usuario del sistema.

9.4.4.3 Gestión de formatos

El prototipo cuenta con una sola clase que cumple el rol de *Facade* como controlador para gestionar la entrega de formatos TSP con el propósito de no conocer los detalles de su obtención. Puesto que el usuario solicita los formatos en su estado actual y éstos pueden o no existir, la clase se encarga de obtener el formato del sistema persistente o en su defecto solicitarlo a la fábrica de formatos, centrándose en proveer formatos al usuario tal como se muestra en la ilustración 17.

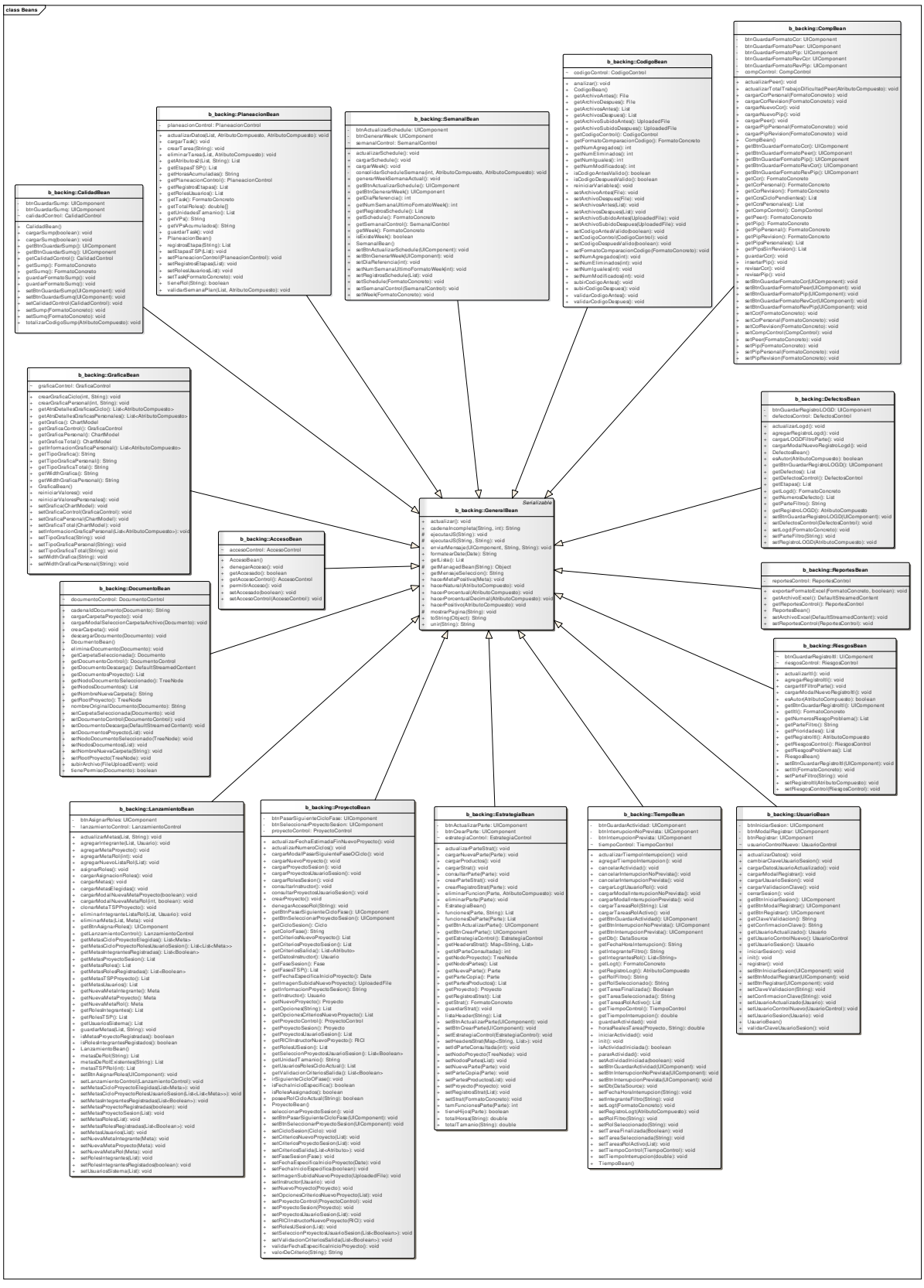


Ilustración 17. Diagrama de clases de Beans. Capa Web. Realizado con Enterprise Architect. Fuente autor

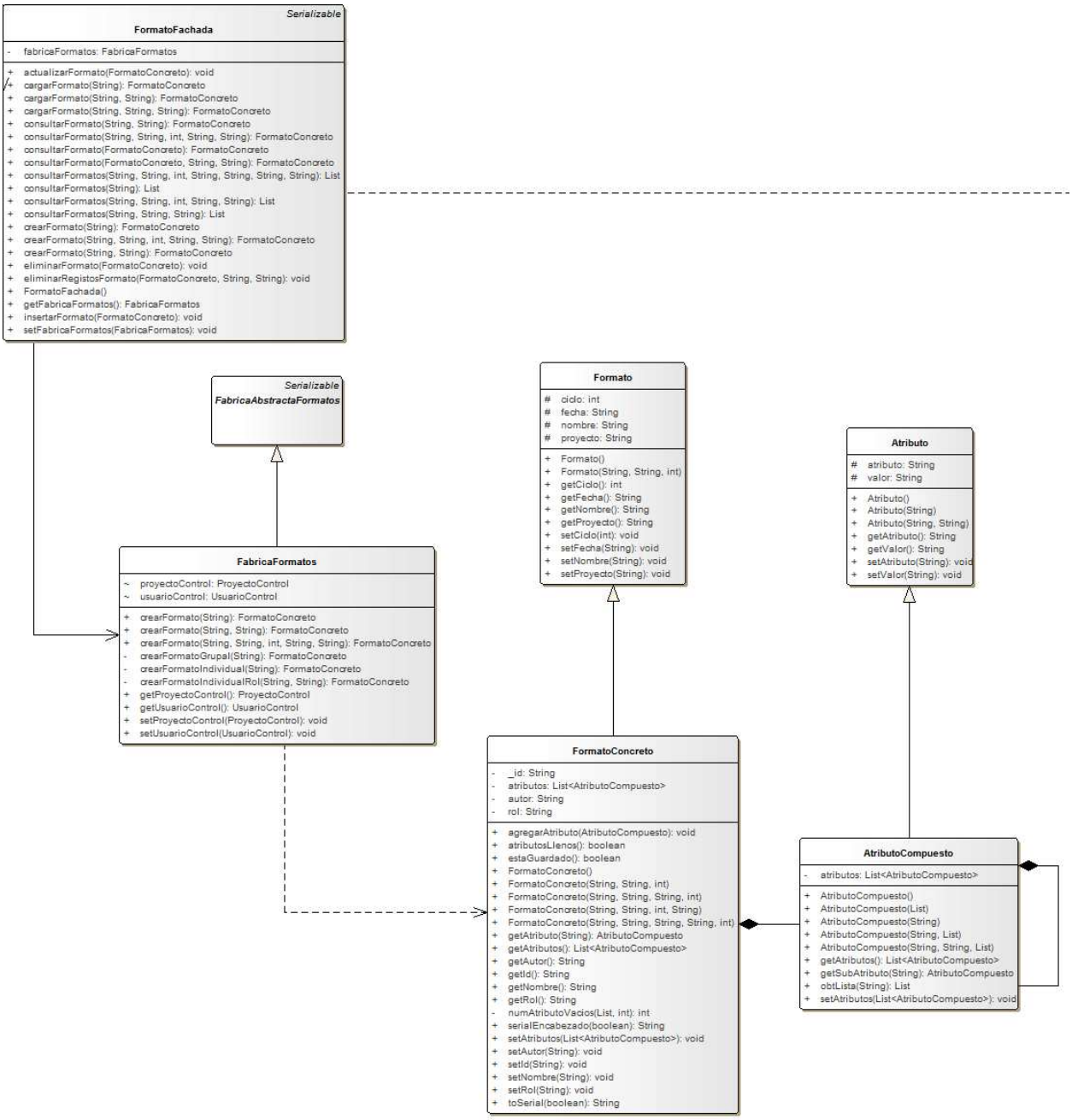


Ilustración 18. Diagrama de clases de gestión de formatos. Realizado con Enterprise Architect. Fuente autor

FabricaFormatos define la construcción de formatos de acuerdo a la clasificación de dominio, es decir, si el formato es compartido por todo el equipo, por todos los integrantes de un rol o sólo por el integrante. De igual manera cabe destacar que los formatos, dadas sus diferencias, gestionan ya sea indicadores, mediciones y/o métricas, por lo que se ha establecido un elemento en común denominado atributo, tal que, cada formato se compone de varios de estos atributos los cuales pueden incorporar a otros atributos dependiendo la complejidad del formato.

Tabla 39. Clasificación de formatos de acuerdo a tres criterios principales. Fuente autor

Formato	***Crecimiento vertical***	***Dependencia entre registros***	***Valores comunes***
TASK	*X*	*X*	*X*
LOGD	*X*	-	-
LOGT	*X*	-	-
STRAT	*X*	*X*	*X*
ITL	*X*	-	-
PEER	-	-	-
CCR	-	-	-
SCHEDULE	*X*	*X*	-
WEEK	-	*X*	-
SUMQ	-	*X*	-
SUMP	-	*X*	-
PIP	*X*	-	-

El análisis de atributos de cada formato se resume en la tabla 38. En esta tabla es posible observar la clasificación con base en el crecimiento vertical que hace referencia a que el formato aumenta su tamaño como si se tratara de una tabla, la dependencia entre registros que define si los valores de un registro depende de los valores de otro, y la columna de valores comunes para indicar si algún registro incorpora a varios subregistros del formato.

9.4.4.4 Gestión de atributos

En colaboración a la gestión de los formatos, la fabricación de atributos define las métricas, indicadores y mediciones de cada formato en particular, apoyándose igualmente en el patrón de diseño *Factory*. Ver ilustración 19.

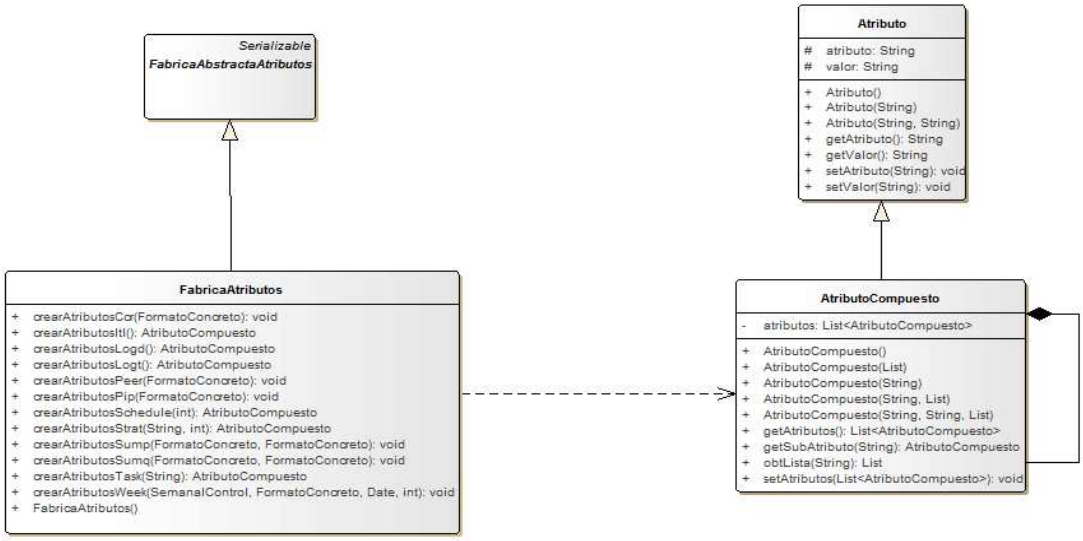


Ilustración 19. Diagrama de clases de gestión de atributos. Realizado con Enterprise Architect. Fuente autor

9.4.4.5 Revisión de código

En forma complementaria se brinda la herramienta comparativa de versiones de código para enriquecer la información de tamaño del software con base en las funciones implementadas. Mediante el patrón *Factory* se construyen los revisores de código del lenguaje de programación sobre el que se desarrolla el producto software del equipo TSP. Ver ilustración 20.

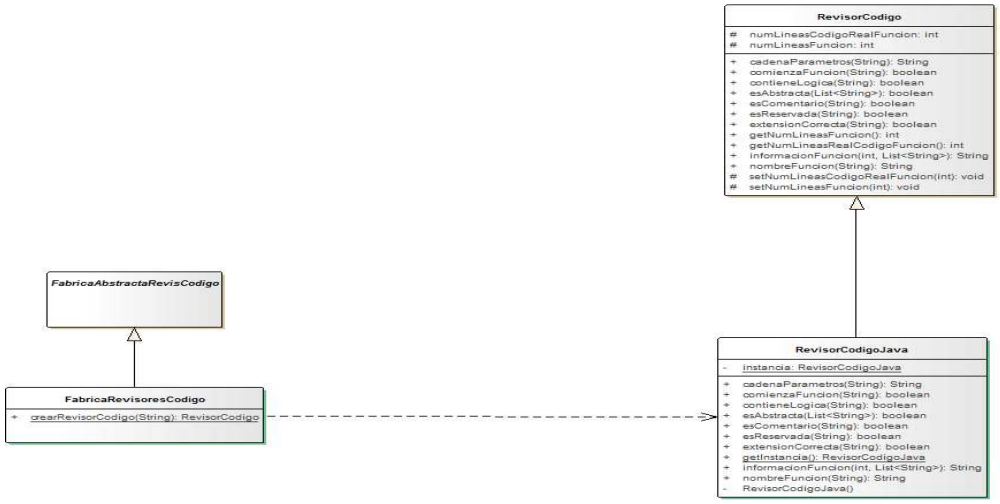


Ilustración 20. Diagrama de clases para componente de revisión de código. Realizado con Enterprise Architect. Fuente autor

9.4.4.6 Revisión cualitativa

Para evaluar cualitativamente los resultados se establece la fabricación de diagramas, entre los que se proveen el diagrama de torta, de líneas y de barras. En la ilustración 21 se puede observar el uso del patrón *Factory* que permite independizar los datos de la gráfica sobre la que se van a representar.

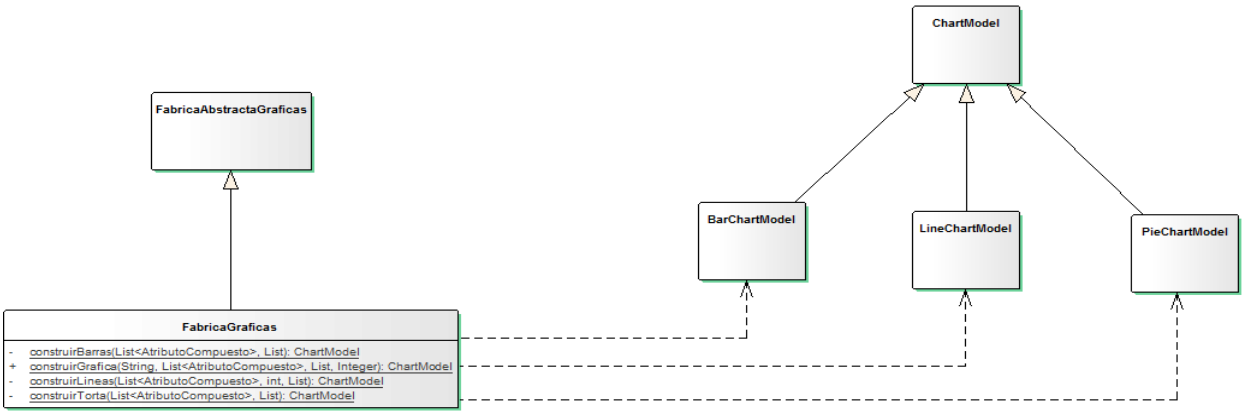


Ilustración 21. Diagrama de clases para la gestión cualitativa. Realizado con Enterprise Architect. Fuente autor

9.4.4.7 Gestión de persistencia

Mediante el patrón *DAO* [110] se contextualizan las transacciones que gestionan el proyecto y los integrantes de éste y del sistema. Ver ilustración 22.

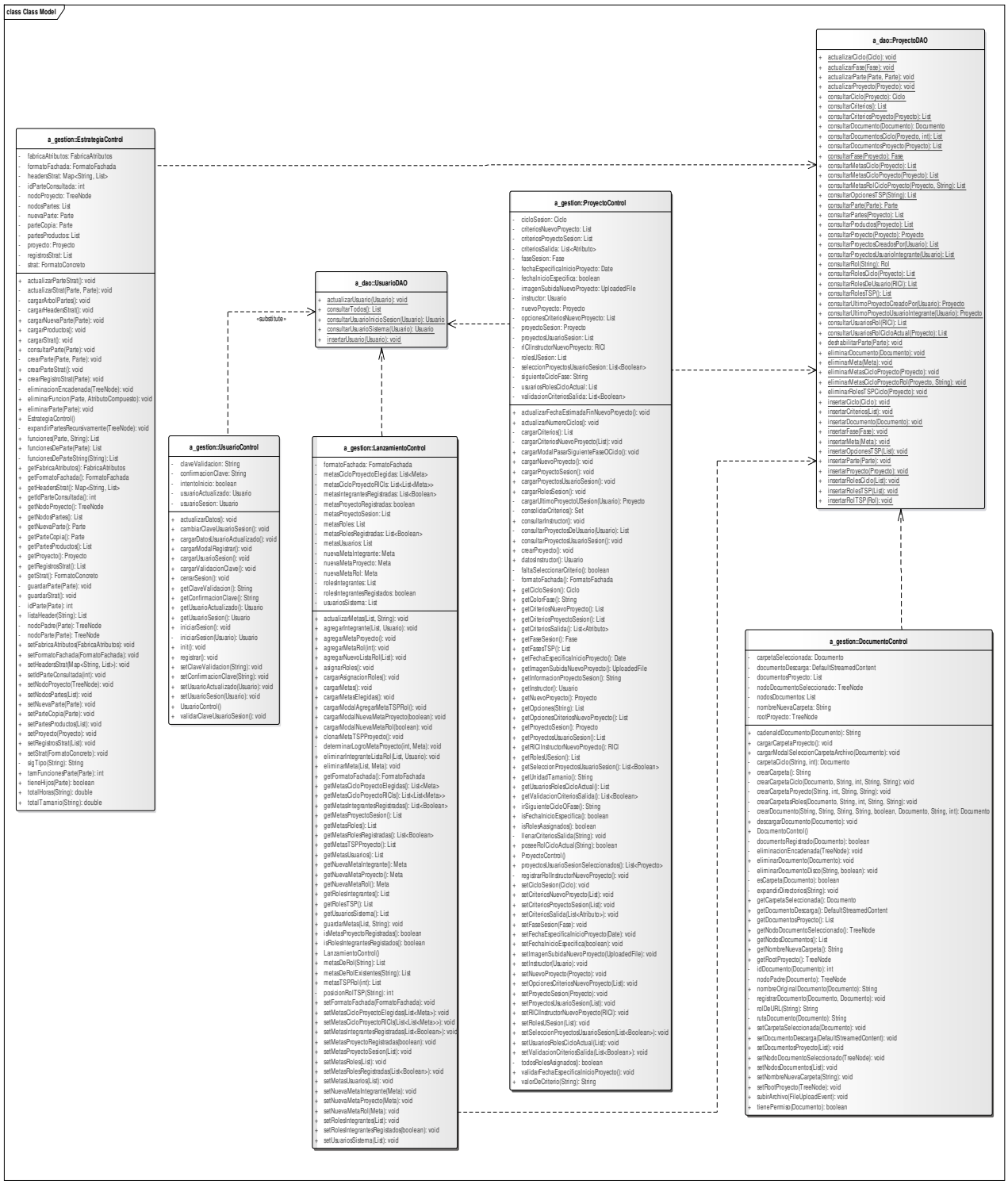


Ilustración 22.Diagrama de clases de gestión de persistencia. Realizado con Enterprise Architect. Fuente autor

9.4.4.8 Conexiones a bases de datos

Una fábrica de conexiones brinda una instancia del *DataSource* del modelo relacional o del modelo no relacional, para este caso PostgreSQL y MongoDB respectivamente. Ver ilustración 23.

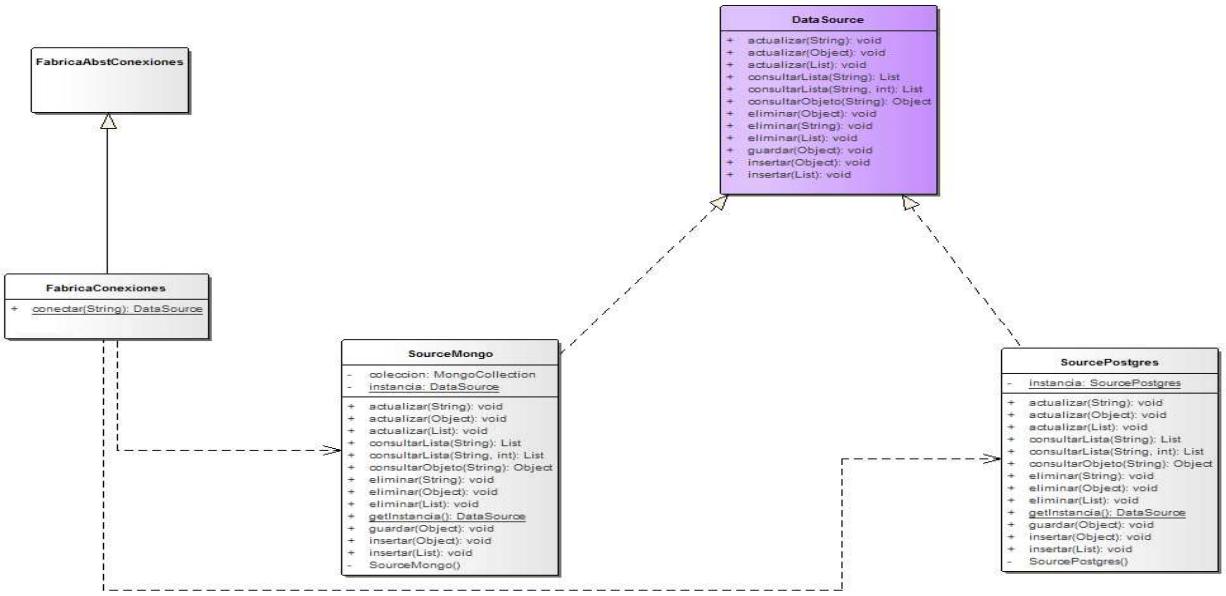


Ilustración 23. Diagrama de clases para la gestión de persistencia. Realizado con Enterprise Architect. Fuente autor

La interfaz *DataSource* define las transacciones típicas sobre un motor de base de datos tal que *SourceMongo* y *SourcePostgres* implementan dichas transacciones a fin de gestionar los formatos y el proyecto respectivamente.

9.4.5 Diagramas de secuencia

En la ilustración 24 se puede observar el diagrama de secuencia para la carga de métricas, indicadores y otras mediciones en torno al formato SUMP. En este diagrama se puede observar cómo la *fachada* de formatos resuelve la solicitud del usuario, solicitando los datos a los gestores de persistencia documental y, en caso de no localizar dicho formato, solicitar su construcción a partir de la generación de los atributos que lo componen, para posteriormente presentar la información al usuario.

Con este diagrama de secuencia se puede dar una clara idea de las responsabilidades que cada elemento del software para satisfacer las necesidades del cliente, especialmente tratándose del formato que sintetiza los datos recolectados durante el desarrollo a través de la metodología

TSP. En el anexo 2 se pueden observar los demás diagramas de secuencia para el proyecto, a fin de conocer el paso de mensajes para los formatos TSP, la gestión documental y gestión de actividades.

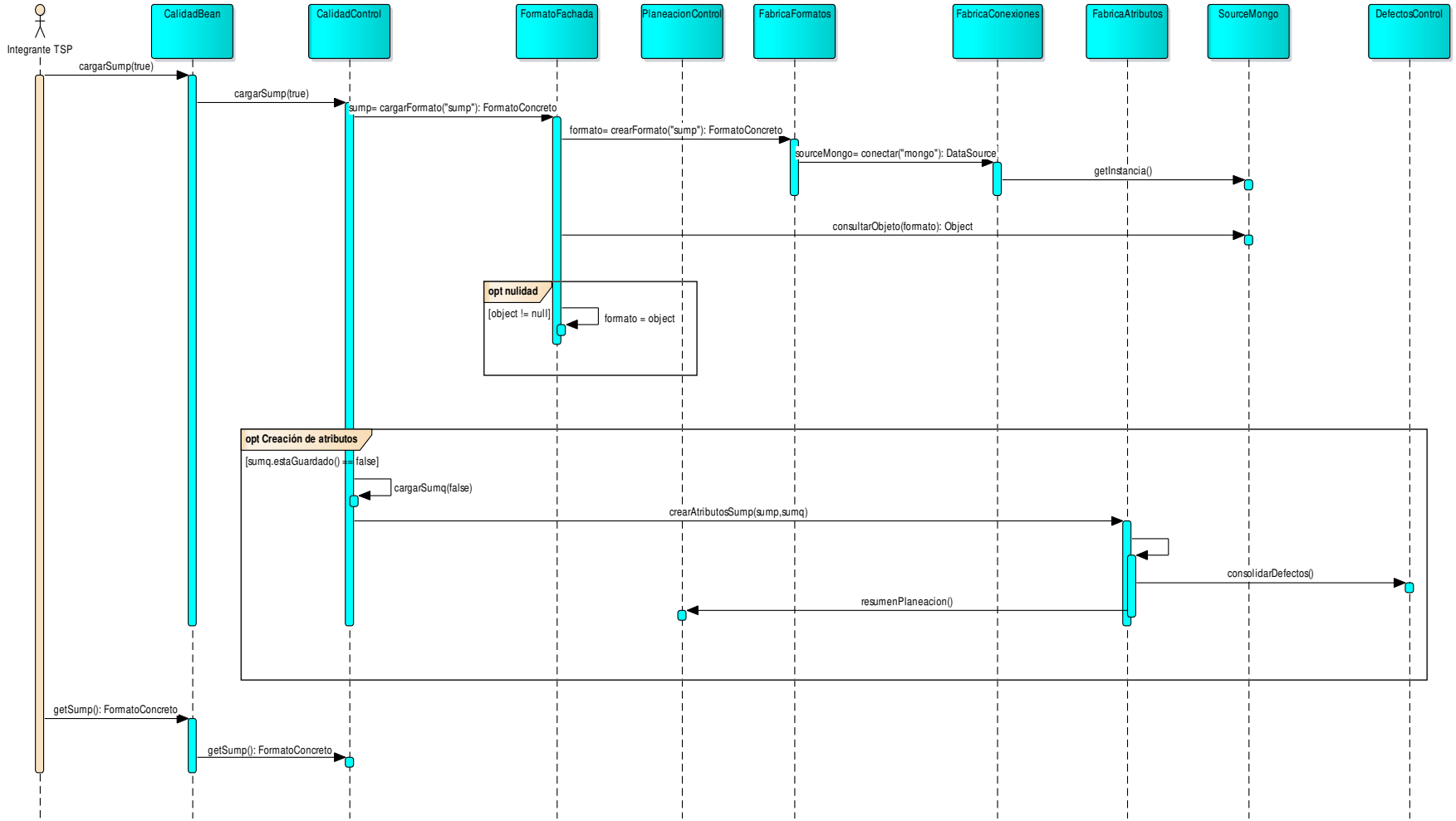


Ilustración 24. Diagrama de secuencia para la carga de métricas, indicadores y mediciones de un proyecto de desarrollo de software. Creado con la herramienta Enterprise Architect. Fuente autor

9.4.6 Diagrama de despliegue

El diagrama de la ilustración 25 muestra la distribución de los dispositivos involucrados en la ejecución del prototipo y las tecnologías utilizadas sobre cada uno de ellos. Cabe destacar que en la capa de presentación se provee una tecnología propia del proyecto que se describirá a detalle posteriormente en el apartado tecnológico (9.6.2.1.2.3.2).

Este diagrama de despliegue separa el dispositivo servidor de los motores de la base de datos, pues aunque el prototipo incorpora los tres dispositivos en uno, se definen como elementos independientes, como lo sería en un caso específico de producción.

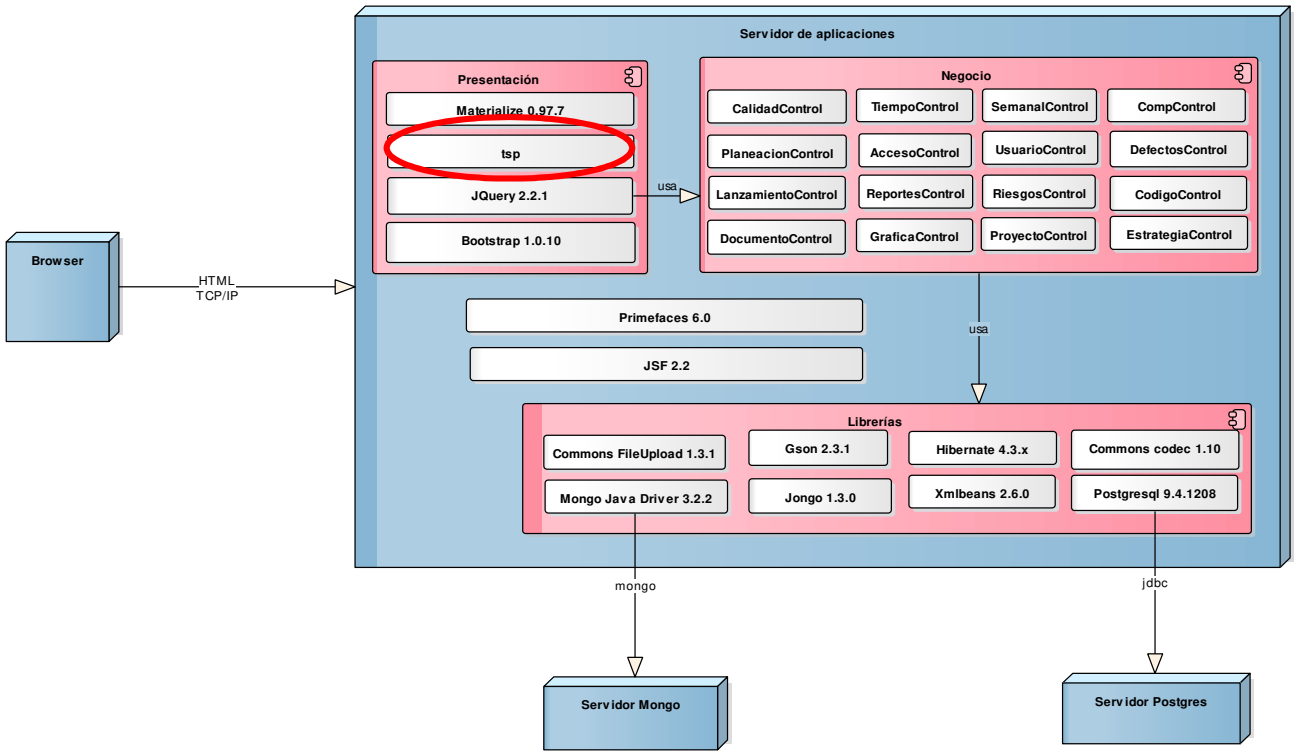


Ilustración 25. Diagrama de despliegue. Realizado con Enterprise Architect. Fuente autor

Este diagrama de despliegue evidencia el bajo requerimiento tecnológico del cliente, ya que solo depende del navegador. Asimismo, el nodo correspondiente al servidor representa los componentes separados, dejando en la capa de negocio únicamente los controladores que satisfacen los procesos requeridos por el cliente. Además, librerías como Primefaces y JSF son requeridas tanto por la capa de presentación como por la capa de negocio, para poder mapear y gestionar la información presentada en el navegador con las estructuras de negocio.

9.4.7 Diagrama de componentes

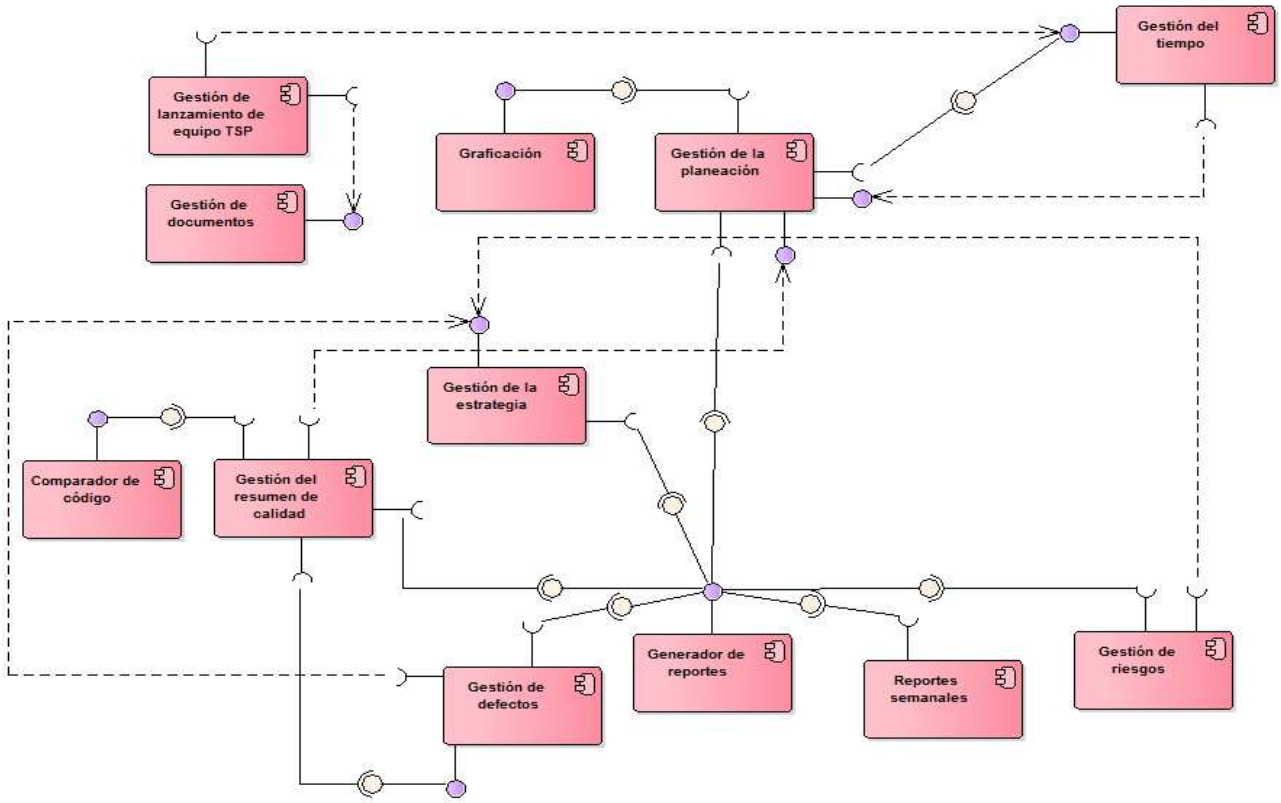


Ilustración 26. Diagrama de componentes. Realizado con Enterprise Architect. Fuente autor

Las piezas que componen al prototipo se evidencian en el diagrama de componentes de la ilustración 26. En este diagrama se puede observar cómo cada servicio de la aplicación permite que varios componentes los consuman a fin de lograr sus objetivos, como lo es el caso de la gestión del tiempo y la planeación, o el proveer un valor agregado como es el caso del generador de reportes y la gestión de la estrategia.

En este diagrama es posible reconocer la dependencia lógica entre las métricas recolectadas en forma independiente en cada componente, consolidando la información principalmente en la gestión del resumen de calidad donde se dirige gran parte de la atención de los usuarios, esto debido a que sintetiza la información para su concreto análisis y por ende su rápida retroalimentación.

9.4.8 Diagrama de estado

Cada ciclo del proyecto se distribuye secuencialmente por las fases mostradas en el diagrama de estado de la ilustración 27 hasta el último ciclo donde, luego de la fase de Postmortem, finaliza el proyecto.

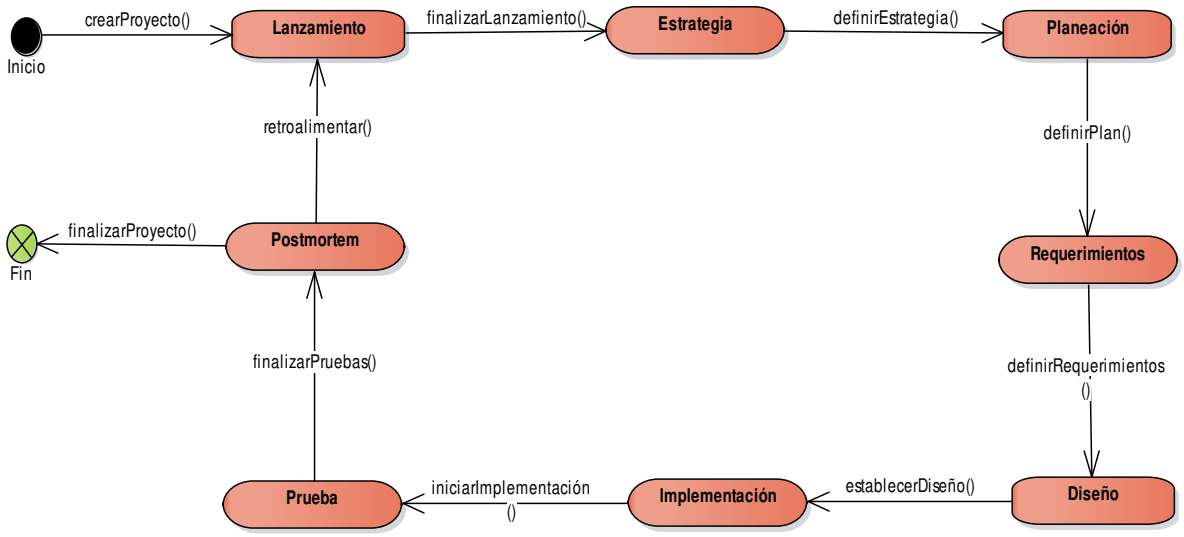


Ilustración 27. Diagrama de estado. Realizado con Enterprise Architect. Fuente autor

Para dar inicio a la fase de lanzamiento, primero debe crearse el proyecto. El siguiente evento para cambiar de estado corresponde a la finalización del lanzamiento, el cual corresponde a la establecimiento de roles y metas, llegando a la fase de Estrategia. Para posteriormente pasar a la fase de planeación se requiere haber completado la estrategia del ciclo. El evento correspondiente a la definición del plan permite pasar a la fase de requerimientos, en la que se definen éstos para seguir a la fase de diseño, luego a la fase de implementación y pruebas; cada una de estas fases se completa al ejecutar en su totalidad lo que sus respectivos nombres indican. Finalmente, en la fase de Postmortem, se lleva a cabo una retroalimentación y la adquisición de un producto mejorado, que permite ir al siguiente ciclo, o por el contrario finalizar el proyecto de acuerdo al número de ciclos de éste.

9.5 MODELOS DE PERSISTENCIA

Al evaluar el funcionamiento de la metodología TSP, se determinó la necesidad de utilizar un modelo relacional y un modelo no relacional para satisfacer la integridad de la información del proyecto y sintetizar la gestión de formatos respectivamente.

9.5.1 Modelo relacional

La integridad referencial toma prioridad en la gestión del proyecto para garantizar el correcto estado del mismo a lo largo de los ciclos y fases que lo componen, garantizar la pertenencia de los integrantes a un proyecto, su respectiva asignación de roles, la definición de metas y el control jerárquico de la documentación y partes de cada ciclo.

En la ilustración 28 se observa el modelo que soporta la ejecución de la metodología TSP tal que cada tabla cumple una función específica como se muestra en la tabla 40. Esto permite soportar eficazmente la integridad de los datos, reducir tiempos de consulta y facilitar la internacionalización del prototipo.

Tabla 40. Descripción de tablas de modelo relacional. Fuente autor

Tabla	Descripción
Usuario	Usuarios del sistema
Rol	Roles de la metodología TSP
Rl_Cl	Roles asignados para el ciclo del proyecto
Proyecto	Proyectos creados por los usuarios registrados en el sistema
Ciclo	Ciclos de cada proyecto registrado
Fase	Fases de cada ciclo registrado
OpcionTSP	Opciones de la metodología TSP
Criterio	Criterios TSP disponibles
Py_Cr	Criterios para cada proyecto creado
Meta	Metas de cada ciclo y/o roles
Documento	Documentos de cada proyecto
Parte	Partes de cada ciclo

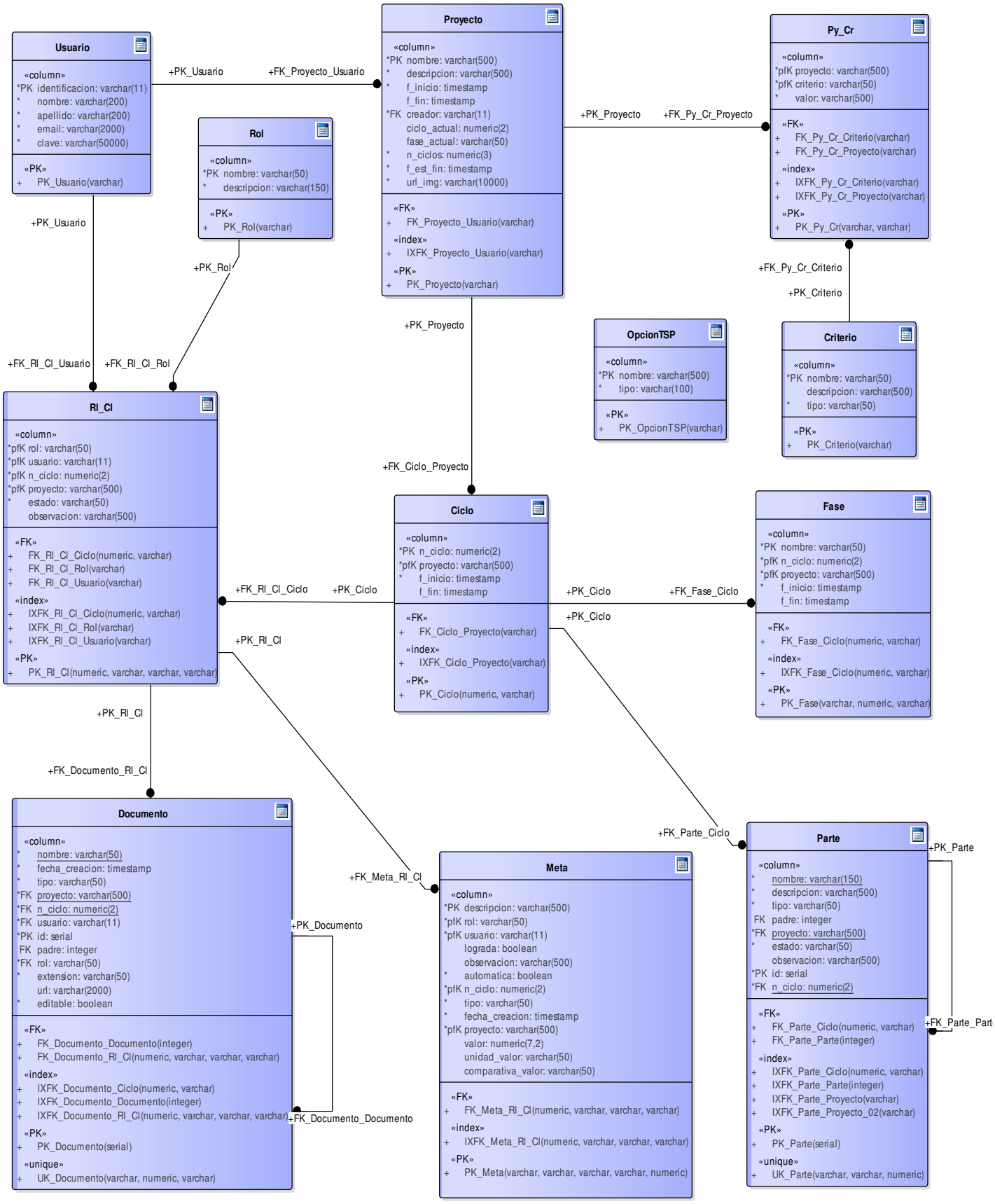


Ilustración 28. Modelo de base de datos relacional. Realizado con Enterprise Architect. Fuente autor

El proyecto posee los atributos dependientes de ciclo actual y fase actual con el fin de reducir el costo de las consultas, lo que requirió la implementación de dos disparadores para el ciclo y fase respectivamente. De igual manera, el modelo de datos permite asociar un documento a otro para construir el directorio del proyecto y asociar una parte a otra de acuerdo a la jerarquía: sistema, producto, componente y módulo.

De igual forma, se establecieron los privilegios de los usuarios sobre las tablas del modelo relacional tal como se muestra en la tabla 41. Todos los usuarios tienen privilegios de consulta e inserción sobre todas las tablas y únicamente pueden hacer eliminación de registros sobre las tablas relacionadas al cuaderno de proyecto de TSP, el establecimiento de metas, asignación de roles y criterios específicos del equipo. En cuanto a las actualizaciones, en la tabla se pueden observar los campos que están permitidos modificar para permitir cambios a lo largo de la ejecución de un proyecto TSP.

Tabla 41. Privilegios de los usuarios sobre las talas del modelo relacional. Fuente autor

TABLA	***C***	***I***	***ACTUALIZACIÓN***	***E***
USUARIO	*X*	*X*	*NOMBRE,APELLIDO,EMAIL,TELEFONO,CLAVE*	-
ROL	*X*	*X*	-	-
PROYECTO	*X*	*X*	*NOMBRE,DESCRIPCION,F_INICIO,F_FIN,CICLO_ACTUAL,FASE_ACTUAL,N_CICLOS,F_EST_FIN,URL_IMG*	-
CICLO	*X*	*X*	*F_INICIO,F_FIN*	-
FASE	*X*	*X*	*F_INICIO,F_FIN*	-
DOCUMENTO	*X*	*X*	-	*X*
META	*X*	*X*	-	*X*
CRITERIO	*X*	*X*	-	-
OPCIONTSP	*X*	*X*	-	-
RL_CL	*X*	*X*	*ESTADO,OBSERVACION*	*X*
PY_CR	*X*	*X*	*VALOR*	*X*
PARTE	*X*	*X*	*NOMBRE,DESCRIPCION,TIPO,PADRE,ESTADO,OBSERVACION*	-

9.5.1.1 Tabla Usuario

Entidad que describe a las personas que se registran y hacen uso de la herramienta software.

Tabla 42. Atributos de tabla Usuario. Fuente autor

Atributo	**Tipo**	**Descripción**
identificacion	varchar(11)	Identificador único de usuario
nombre	varchar(200)	Nombre del usuario
apellido	varchar(200)	Apellido del usuario
email	varchar(2000)	Correo electrónico del usuario
clave	varchar(50000)	Contraseña de acceso al sistema del usuario

9.5.1.2 Tabla Rol

Entidad que describe a los roles de la metodología TSP.

Tabla 43. Atributos de tabla Rol. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(50)	Nombre del rol de la metodología TSP
descripcion	varchar(150)	Información descriptiva del rol TSP

9.5.1.3 Tabla Proyecto

Entidad que describe a los proyectos a desarrollar mediante la metodología TSP.

Tabla 44. Atributos de tabla Proyecto. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(500)	Nombre del proyecto
descripcion	varchar(500)	Descripción breve del proyecto
f_inicio	timestamp	Fecha de inicio del proyecto
f_fin	timestamp	Fecha finalización real del proyecto
creador	varchar(11)	Identificador del usuario creador del proyecto
ciclo_actual	numeric(2)	Ciclo actual en el que se encuentra el proyecto
fase_actual	varchar(50)	Fase actual en la que se encuentra el proyecto
n_ciclos	numeric(3)	El número de ciclos del proyecto
f_est_fin	timestamp	Fecha estimada de finalización del proyecto
url_img	varchar(10000)	Dirección donde se localiza el logo del proyecto

9.5.1.4 Tabla Py_Cr

Entidad que establece la relación entre los proyectos y los criterios del mismo.

Tabla 45. Atributos de tabla Py_Cr. Fuente autor

Atributo	Tipo	Descripción
proyecto	varchar(500)	Nombre del proyecto
criterio	varchar(50)	Nombre del criterio
valor	varchar(500)	Valor del criterio para el proyecto

9.5.1.5 Tabla Criterio

Entidad que describe los criterios de todo proyecto a desarrollar mediante TSP.

Tabla 46. Atributos de tabla Criterio. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(50)	Nombre del criterio de la metodología TSP
descripción	varchar(500)	Descripción del criterio de la metodología TSP
tipo	varchar(50)	Tipo de criterio que especifica el modo de entrada del dato por el usuario

9.5.1.6 Tabla OpcionTSP

Entidad que describe a las opciones de datos sobre la metodología TSP.

Tabla 47. Atributos de tabla OpcionTSP. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(500)	Nombre de la opción de la metodología TSP
tipo	varchar(100)	Contexto de la opción

9.5.1.7 Tabla Ciclo

Entidad que describe a los ciclos de proyecto.

Tabla 48. Atributos de tabla Ciclo. Fuente autor

Atributo	Tipo	Descripción
n_ciclo	numeric(2)	Número del ciclo
proyecto	varchar(500)	Nombre del proyecto al que pertenece el ciclo
f_inicio	timestamp	Fecha de inicio del ciclo
f_fin	timestamp	Fecha de finalización del ciclo

9.5.1.8 Tabla Fase

Entidad que describe a las fases de cada ciclo de proyecto.

Tabla 49. Atributos de tabla Fase. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(50)	Nombre de la fase
n_ciclo	numeric(2)	Número del ciclo al que pertenece la fase
proyecto	varchar(500)	Nombre del proyecto en el que se localiza la fase
f_inicio	timestamp	Fecha de inicio de la fase
f_fin	timestamp	Fecha de finalización de la fase

9.5.1.9 Tabla Rl_Cl

Entidad que describe los roles asignados para un ciclo en particular.

Tabla 50. Atributos de tabla Rl_Cl. Fuente autor

Atributo	Tipo	Descripción
rol	varchar(50)	Rol asignado para el ciclo
usuario	varchar(11)	Usuario que posee el rol para el ciclo
n_ciclo	numeric(2)	Número de ciclo sobre el que se asigna el rol
proyecto	varchar(500)	Nombre del proyecto al que pertenece el ciclo
estado	varchar(50)	Estado de la asignación
observacion	varchar(500)	Observación opcional de la asignación

9.5.1.10 Tabla Parte

Entidad que describe las partes en las que se divide el proyecto.

Tabla 51. Atributos de tabla Parte. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(150)	Nombre de la parte del proyecto
descripcion	varchar(500)	Descripción de la parte del proyecto
tipo	varchar(50)	Tipo de parte: Sistema, Producto, Componente, Módulo
padre	integer	Parte padre que compone
proyecto	varchar(500)	Nombre del proyecto de la parte
estado	varchar(50)	Estado de la parte
observación	varchar(500)	Observaciones opcionales sobre la parte
id	serial	Identificador secuencial de la parte
n_ciclo	numeric(2)	Número de ciclo al que pertenece la parte

9.5.1.11 Tabla Meta

Entidad que describe las metas definidas para cada ciclo de proyecto.

Tabla 52. Atributos de tabla Meta. Fuente autor

Atributo	Tipo	Descripción
descripción	varchar(500)	Descripción de la meta
rol	varchar(50)	Rol asignado a la meta
usuario	varchar(11)	Usuario asignado a la meta
lograda	boolean	Identifica si la meta fue lograda o no

observacion	varchar(500)	Observación opcional de la meta
automatica	boolean	Identifica si la meta se calcula o no por el sistema
n_ciclo	numeric(2)	Número de ciclo al que pertenece la meta
tipo	varchar(50)	Tipo de meta: Rol o de proyecto
fecha_creacion	timestamp	Fecha de creación de la meta
proyecto	varchar(500)	Proyecto al que pertenece la meta
valor	numeric(7,2)	Valor asignado a la meta
unidad_valor	varchar(50)	Unidad de la meta: porcentaje, unidad o día(s)
comparativa_valor	varchar(50)	Identifica si el valor debe ser igual, menor o mayor

9.5.1.12 Tabla Documento

Entidad que describe los documentos del cuaderno de proyecto (carpetas y archivos).

Tabla 53. Atributos de tabla Documento. Fuente autor

Atributo	Tipo	Descripción
nombre	varchar(50)	Nombre del documento
fecha_creacion	timestamp	Fecha de creación del documento
tipo	varchar(50)	Tipo de documento: archivo o carpeta
proyecto	varchar(500)	Proyecto al que pertenece el documento
n_ciclo	numeric(2)	Número de ciclo al que pertenece el documento
usuario	varchar(11)	Usuario que creó el documento
id	serial	Identificador secuencial del documento
padre	integer	Documento padre donde se localiza el documento
rol	varchar(50)	Rol que ha subido el documento
extension	varchar(50)	Extensión del documento
url	varchar(2000)	Localización del documento
editable	boolean	Especifica si es editable o no, es decir, si fue creado por el sistema o por el usuario.

9.5.2 Modelo no relacional

A diferencia de los modelos de datos existentes a la fecha de la metodología TSP, esta propuesta incorpora 12 formatos cuya complejidad produjo un modelo relacional muy grande y difícil de gestionar (33 tablas en total), de modo que, reconociendo un formato como un documento, se optó por un modelo no relacional.

La estructura de cada documento resultante se observa en el anexo 3. Cada formato específico de TSP posee sus atributos particulares con base en la literatura de la metodología TSP, y por tanto, la descripción de cada atributo se expone en los anexos del libro *Introduction to the Team Software Process* [7].

Con el modelo documental propuesto es posible añadir nuevos formatos, a solicitud de la entidad que requiera del aplicativo, a un bajo costo de modificación y así, satisfacer rápidamente nuevos requerimientos asociados a los formatos para facilitar la gestión de métricas.

9.6 HERRAMIENTAS TECNOLÓGICAS PARA EL DESARROLLO DEL PROYECTO

Retomando las especificaciones tecnológicas del apartado 9.2.6, a continuación se especifica el ambiente de desarrollo, los componentes requeridos para el funcionamiento del prototipo y los ambientes de prueba.

9.6.1 Ambiente de desarrollo

El desarrollo se hizo sobre una laptop Lenovo G40-30 con procesador Intel Celeron N2830 de arquitectura a 64 bits a 2.16 Ghz y 8GB de memoria RAM. Este ordenador cuenta con dos sistemas operativos (Ubuntu 14.04.2 y Windows 8.1) sobre los cuales se instaló la aplicación en para desligar la dependencia del sistema operativo.

Las revisiones de funcionamiento del producto software, en forma iterativa e incremental, se llevaron a cabo en la misma máquina de desarrollo y sobre un dispositivo celular Samsung Galaxy Ace 4 con sistema operativo Android 4.4.2 mediante una conexión vía WiFi a fin de reconocer la independencia de la plataforma del cliente.

9.6.2 Software utilizado en el desarrollo

9.6.2.1 Back-End

9.6.2.1.1 Java (JDK 1.8.0_101)

Tecnología elegida para el proyecto por ser un lenguaje multiplataforma, disponer de una gran cantidad de recursos y documentación, brindar seguridad, portabilidad, robustez, independencia de la arquitectura, soporte de multihilos, dinamismo y especialmente, proveer de especificaciones a los programas, haciéndoles fáciles de entender y ser extensibles [93].

En la implementación del proyecto, a fin de probar la portabilidad y obtener más eficiencia por parte de los desarrolladores, se migró temporalmente el código al sistema operativo Ubuntu 14.04.2, en el que, a nivel de Java, no se obtuvo diferencia alguna. Además, al inicio del desarrollo, se consideró utilizar un modelo de persistencia relacional con el que se conocía la tecnología de conexión para Java, sin embargo a posterioridad el modelo orientado a documentos se consideró como mejor alternativa (en cuanto a formatos) y Java no fue un impedimento para incorporarlo.

9.6.2.1.1.1 Java Server Faces (JSF 2.2)

Es una especificación que define un framework web para la creación de aplicaciones [94]. Es un estándar Java hacia la construcción de interfaces de usuario para aplicaciones web que simplifican el desarrollo del lado del cliente con base en la arquitectura MVC (Modelo Vista Controlador). Esta versión brinda mejoras en la navegación, control de excepciones, mejoras en la expresión del lenguaje y validación de beans [95].

Esta tecnología posee una arquitectura similar a cualquier otra tecnología Java basada en una aplicación web (ilustración 29), obteniendo un servlet que contiene componentes JavaBeans, etiquetas personalizables y clases auxiliares. JSF provee mecanismos de renderizado individual de componentes, permitiendo gestionar la presentación sin importar el mecanismo utilizado para representarla [94].

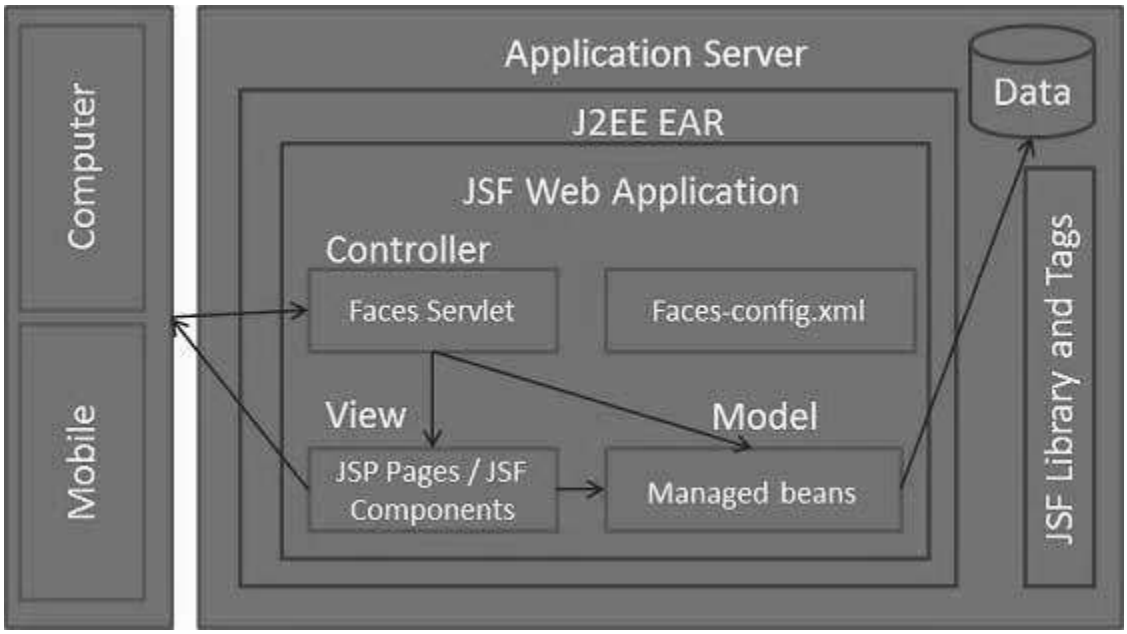


Ilustración 29. Arquitectura JSF. Fuente [94]

La arquitectura específica del software de TSP posee ligeras diferencias con la arquitectura base de JSF mostrada en la ilustración 28, ya que se separó la capa *Modelo* en capa *Web* y capa *Controlador*, con la finalidad de separar la gestión de los componentes web de la gestión de datos. Además, como se describió anteriormente, los datos están depositados en dos servidores independientes.

9.6.2.1.1.1.1 Managed Bean

Es una clase regular Java Bean gestionada por el framework JSF [94], y por tanto, conserva las características básicas de cualquier aplicación Java, a excepción de los comparadores sobre el ambiente web.

Esta tecnología provee diversas funcionalidades, aunque muy útiles, muy difíciles de leer para el equipo, y por tanto, de interpretar. Por esta razón se produjeron las siguientes funciones reutilizables.

- *mostrarPagina(String nombrePagina)***.** Una función que resume el llamado del *redirect* del *ExternalContext* que proviene del *CurrentInstance* que a su vez depende del *FacesContext.* El objetivo es facilitar la muestra de una página web de JSF de modo que al leer el llamado, se interprete fácilmente que se quiere cargar la página solicitada para el usuario.
- *getManagedBean(String clase).* Con el fin de facilitar la comunicación entre los beans de JSF, sin hacer una compleja llamada, se crea esta función capaz de devolver el bean a partir de su nombre. Resume el llamado del *evaluateExpressionGet* del *Application*, del *FacesContext*, haciendo más fácil de interpretar y unificando el llamado.
- *unir(String cadena).* Para facilitar la indización, la función une (remueve espacios) y ajusta una cadena soportada por las páginas JSF.
- *actualizar().* Una función sin contenido, sin embargo, dado el modo de ejecución de JSF, permite hacer el mapeo de los datos diligenciados por los usuarios a las variables del servidor.
- *hacerPositivo(AtributoCompuesto atributo).* Gran parte de las validaciones de la metodología, se centran en garantizar que los valores son positivos, de modo que esta función es reutilizada constantemente, independizando el dato recolectado.
- *formatearDate(Date fecha).* Para presentar las fechas registradas en el sistema, a los usuarios, se brinda una función capaz de mapear la fecha y presentarla en formato *dd/MM/yyyy.*
- *cadenaIncompleta(String cadena, int numCaracteres).* Brinda una subcadena en un total de caracteres igualmente especificado como parámetro para los casos en que, la cadena pueda ser o muy corta o muy extensa, y de ese modo, no afectar la experiencia del usuario.

- *getMensajeSeleccion()*. Función que brinda el texto *"Seleccione…"* en cada selector del sistema a modo de unificar esta información independientemente del contexto.

9.6.2.1.1.1.2 Converters

Los convertidores permiten mapear componentes web con las listas de objetos. Esto fue útil en la implementación para proveer al usuario de la lista de selección de proyectos a los que pertenece.

9.6.2.1.1.2 PrimeFaces (6.0)

Es una librería de componentes visuales de código abierto para JSF, desarrollada y mantenida por *Prime Techology,* con el objetivo principal de ofrecer un conjunto de componentes para facilitar la creación y diseño de aplicaciones web [95]. Aunque una de sus principales características es su kit para la creación de aplicaciones web móviles, la razón por la que se ha elegido esta tecnología, es su uso flexible de JavaScript y proveer herramientas para la gestión de archivos y graficado.

Para sacar mayor provecho y comprensión del uso de lenguaje JavaScript que provee PrimeFaces, se definió la función sobrecargada *ejecutarJS*. Esta función se encarga de ejecutar funciones JavaScript recibiendo el nombre de éstas como parámetro o tanto el nombre como los parámetros alojados en un arreglo.

9.6.2.1.1.3 Commons codec (1.10)

Es una librería apache que permite encriptar la información que viaja a través de internet con ayuda del algoritmo MD5, el cual es muy seguro y casi imposible de descifrar, todo esto con una simple línea de código [98]. Esta tecnología en particular permitió encriptar la contraseña del usuario para asegurar sus datos personales.

9.6.2.1.1.4 Commons FileUpload (1.3.1)

Esta librería provee un framework avanzado para el manejo de archivos de tal forma que permite leerlo mientras está en memoria [99]. Esta librería ha sido utilizada para permitir a los usuarios gestionar los documentos del directorio del proyecto.

9.6.2.1.1.5 Gson (2.3.1)

Gson o también conocida como *Google JSON,* es una librería para trabajar con JSON para facilitar el serializado y deserializado de objetos Java, lo que es más cómodo y fácil de trabajar

[100]. Particularmente se ha utilizado esta tecnología para la gestión de formatos y algunos objetos auxiliares que, gracias al proceso de serialización y deserialización, se pudieron clonar sin esfuerzos significativos.

9.6.2.1.1.6 Hibernate (4.3.x)

Es una herramienta ORM completa y considerada como el producto *OpenSource* líder en este campo, gracias a sus prestaciones, buena documentación y estabilidad. Corresponde a la capa de persistencia objeto/relacional y es un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones y un gran número de tipos de datos [101].

Hibernate ha sido utilizado para poder mapear los objetos correspondientes a las tablas de la base de datos relacional, donde se gestiona principalmente el proyecto, los usuarios y los roles. Esta tecnología permitió ejecutar las transacciones sin necesidad de utilizar procedimientos almacenados y asegurando la integridad de los datos.

9.6.2.1.1.7 Mongo Java Driver (3.2.2)

Proporciona integración sincrónica y asincrónica con MongoDB que se alimenta de una biblioteca BSON independiente que utiliza codificadores de alto rendimiento y decodificadores sin requerir una instancia de mapa intermedia [102]. Esta herramienta brinda soporte a la conexión con el motor de base de datos MongoDB para gestionar la persistencia de los formatos TSP.

9.6.2.1.1.8 Jongo (1.3.0)

Es una librería capaz de mapear objetos Java para ser almacenados en el motor MongoDB y a la vez es capaz de brindar módulos que pueden ejecutar sentencias nativas del motor sin un esfuerzo significativo [103]. Con esta tecnología fue posible mapear y ejecutar transacciones relacionadas con el objeto *Formato* en colaboración con la interfaz definida para la capa de datos, *DataSource.*

9.6.2.1.1.9 Xmlbeans (2.6.0)

Es una tecnología para el acceso a XML mediante Java, que proporciona un sistema que genera, a partir de XML, una hoja de cálculo [104]. Gracias a la utilización de esta herramienta, es posible exportar los datos recolectados en los formatos a partir de un llenado fila a fila del documento resultante.

9.6.2.1.2 Front-End

9.6.2.1.2.1 PrimeFaces: Bootstrap (1.0.10)

Es la tecnología que soporta *PrimeFaces* para modificar el denominado *skin* de la aplicación, es decir, la apariencia. En este caso, los componentes web adquieren una apariencia semejante a la tecnología *Front-End, Bootstrap*.

9.6.2.1.2.2 Materialize (0.97.7)

Framework para el desarrollo web con estilo *Material Design.* Desarrollado en SASS y que hace uso de las buenas prácticas en HTML5, CSS3 y JavaScript [107]. Se ha elegido esta tecnología para el desarrollo del proyecto debido a que brinda un diseño responsivo realista, con sombras, animaciones y colores que simulan la cotidianidad y por lo tanto, brinda una experiencia significativa al usuario.

9.6.2.1.2.3 Páginas JSF

En JSF, los componentes web están contenidos en documentos XHTML, que combina las características XML con HTML [94]. La vista es tratada en forma diferente debido a que la programación de la interfaz se hace a través de componentes y está basada en eventos, lo que además hace de esta tecnología muy flexible y capaz de separar claramente la presentación del comportamiento [97].

9.6.2.1.2.3.1 Subvistas

Debido a que la interfaz de usuario es típicamente el aspecto más volátil de una aplicación web durante su desarrollo, JSF brinda herramientas capaces de crear una interfaz flexible y amigable que brinda una distribución enfocada en la organización típica de una página web: cabecera, menú y contenido [97].

Con base en las subvistas proporcionadas por JSF, se construyó un módulo capaz de brindar en todo momento, la gestión de la sesión y de los proyectos para el usuario, de modo que el componente construido fue reutilizado en cada vista de la sesión.

9.6.2.1.2.3.2 Inclusión

Puesto que JSF permite crear componentes propios, se creó la tecnología propia del proyecto denominada *tsp*; una tecnología capaz de resumir, personalizar y por ende agilizar el

desarrollo en el contenido XHTML incluyendo a *Materialize* para brindar contenidos específicos que aportan una significativa experiencia al usuario.

Las variables de componentes de la tecnología *tsp* se describen a continuación, a fin de comprender el contexto por cual son agregados en un contenido específico XHTML.

- *carg*. Componente que despliega el mensaje de *Cargando,* con una animación basada en *Materialize,* al recibir una solicitud del cliente. Requiere de la variable *objeto* para indicar el elemento XHTML a bloquear hasta obtener una respuesta del servidor, y de la variable *elemento* para asignar un índice para este componente a fin de llamarlo desde el evento de la solicitud.
- *text.* Representa un *textarea* normalmente utilizado en una página web, con características provenientes de *Materialize*. Requiere de la variable *id* que representa el identificador del componente, de la variable *tipo* para indicar el tipo de dato a almacenar, de la variable *valor* para asociar un elemento Java, de la variable *max* para controlar el número de caracteres máximo permitido, de la variable *texto* para indicar lo que se desea que el usuario diligencie, de la variable *deshabilitado* en caso de que se desee que el componente no pueda ser modificado por el usuario, de la variable *i* para indicar el ícono asociado al campo de texto y de la variable *m* para indicar el tamaño del componente en término medio para la característica responsiva de *Materialize.*
- *modalConfirm.* Consiste en un modal, de *Materialize*, para solicitar al usuario la confirmación de las transacciones. Requiere de la variable *id* para identificar el componente, de la variable *texto* para describir exactamente la transacción a ejecutar y de la variable *validacion* para indicar el verdadero elemento para efectuar la solicitud, de modo que el validador sepa a qué botón o enlace, por ejemplo, acudir una vez el usuario confirma la operación.
- *entrada.* Representa un *input* normalmente utilizado en una página web, con características provenientes de *Materialize*. Requiere de las mismas variables que el componente *text* excepto por la variable *max*.
- *msg*. Componente para generalizar las características de la información de respuesta del servidor a una solicitud del usuario. Su única variable es *boton* para apuntar al componente XHTML capacitado para el envío de datos del cliente al servidor.

- *select.* Componente para representar un *selector* de XHTML con las características de *Materialize.* Requiere de la variable *m* para definir el tamaño responsivo, de la variable *i* para asociar un ícono, de la variable *lista* para indicar la lista alojada en el *Bean*, de la variable para representar el iterador de la lista, de la variable *itemValor* para indicar lo que cada ítem del selector va a mapear como elección, de la variable *itemLabel* para indicar lo que cada ítem del selector va a mostrar al usuario y de la variable para indicar la variable que va a recibir los datos de acuerdo a la elección del usuario.

Con estos componentes a disposición del equipo de desarrollo, la creación de formularios se hizo una tarea más sencilla y más fácil de leer, pues sintetiza el código, proporciona características puntuales y brinda elementos compuestos a un costo mínimo.

9.7 PRODUCTO RESULTANTE



Ilustración 30. Logo de la herramienta. Fuente autor

Se desarrolló una herramienta web para la gestión métricas a través de la ejecución de proyectos de desarrollo de software soportados en la metodología TSP. Este prototipo fue desplegado en un servidor para el uso libre de la comunidad universitaria en particular, y se localiza en la dirección http://45.55.74.36:8080/tsp con el propósito de colaborar en el aprendizaje y con ello fomentar una cultura disciplinaria en el desarrollo de software en torno a los temas de gestión de proyectos y logro de la calidad de software.

Esta herramienta software, denominada TSPSupport (cuyo logo se evidencia en la ilustración 30), provee los más destacados servicios para la ejecución de proyectos de desarrollo de software mediante la metodología TSP. Brinda soporte a los grupos de diferentes tamaños, permitiendo la asignación de varios roles por integrante, o varios integrantes por rol. De igual manera, permite ejecutar proyectos independientemente del contexto y del alcance, pues tanto entidades académicas como empresariales pueden sacar provecho de la herramienta de acuerdo a sus necesidades.

La gestión de métricas efectuada por el software se independiza de la tecnología con la que se lleve a cabo el producto, de modo que, tanto en tamaño como en tiempo, la herramienta se ajusta eficazmente a cualquier desarrollo. Unido a ello, estas dos principales métricas pueden medirse de diferentes maneras y unidades sin alterar el funcionamiento del software, dadas las necesidades de cada proyecto de desarrollo de software en particular.

9.7.1 COMPONENTES A DESTACAR

Con base en la literatura de la metodología TSP [7], la experiencia de la directora del proyecto frente a la metodología y la conclusión de necesidades de un equipo TSP descrito en el apartado 7.4, se buscó y logró que la herramienta software satisficiera las necesidades en común por quiénes se han acercado empíricamente a la metodología. Estas necesidades satisfechas corresponden a la recolección y validación de información, componentes estadísticos, acceso a materiales como formato y plantillas, visualización de métricas e indicadores, manejo de defectos, guía en el proceso, alineamiento con la literatura y amigabilidad con el usuario.

Aparte de dichas ventajas que engloban el sistema por completo, a continuación, se describen los componentes del software que cumplen un papel relevante en la gestión de las métricas de proyecto y del proceso en sí.

9.7.1.1 Planeación

El plan es una etapa importante en el proceso de ingeniería para dejar en claro las tareas a satisfacer para la obtención de un producto de calidad. Gracias a TSPSupport, los integrantes completan un único formato TASK, sin recurrir al tedioso proceso de crear formatos personales que posteriormente sean transcritos a un TASK general.

En la ilustración 31 se da a conocer el formato TASK dentro de la herramienta, en la que cabe aclarar, se valida cada dato proporcionado por el usuario, corrigiendo inmediatamente cualquier información no permitida. Con base en esta información, gran parte del proceso logra automatizarse, pues es el punto de referencia para comparar los planes de desarrollo y la ejecución real del proyecto.



Ilustración 31. Formato TASK en herramienta prototipo. Fuente autor

9.7.1.2 Gestión de tiempo

Una parte muy importante en la metodología corresponde al registro de tiempo dedicado sobre cada tarea definida en la planeación. Cada integrante debe registrar el tiempo realmente dedicado a cada una de sus tareas, y con la mayor exactitud posible. Esto se logra fácilmente con la herramienta prototipo, donde simplemente se especifica el rol y tarea a ejecutar (ilustración 32), gestionando los tiempos de interrupción y algunos comentarios, tal como se lleva a cabo en el formato LOGT.

A diferencia del tradicional diligenciamiento del formato, la herramienta permite obtener una mayor exactitud, dado que todos datos se automatizan y únicamente se solicita informar si la tarea ha sido finalizada o no, para registrar la contribución al proyecto o permitir registrar más tiempos a dicha tarea, respectivamente.



Ilustración 32. Registro de tiempo en TSPSupport. Fuente autor



Ilustración 33. Finalización de tarea en TSPSupport. Fuente autor

En la ilustración 33 se puede observar que el software brinda la información de la tarea una vez finalizada (fecha y hora de inicio y finalización, tiempo realmente dedicado y tiempo interrumpido), permitiendo al usuario registrar dicha información en el sistema (botón guardar) o por el contrario, perder la información si el usuario no considera pertinente registrar dicho tiempo (botón cancelar).

También es importante mencionar que la herramienta ofrece dos tipos de interrupciones, a diferencia de los software existentes (mencionados en el apartado 7.2) y del formato LOGT [7]. Las interrupciones fueron clasificadas como interrupciones previstas y como interrupciones imprevistas, es decir, interrupciones que se conocían de antemano y que fueron marcadas en su momento de inicio, e interrupciones que se ejecutaron en forma inesperada, respectivamente.

Puesto que las interrupciones previstas pueden marcar un tiempo exacto de inicio y de finalización, el usuario puede dar inicio a este tiempo dando un sencillo *click* en el botón de interrupción prevista de la ilustración 31, y presionando nuevamente el botón al momento de finalizar la interrupción, lo que le informará a dicho usuario el tiempo total de interrupción y si desea o no guardar esta información sobre la tarea.

En cuanto a la interrupción imprevista, se brinda otro botón (ver ilustración 31) sobre el registro de tiempo para que el usuario ingrese el tiempo que considere se tardó la interrupción que no ha sido marcada en el sistema. En este caso, el software valida que el tiempo ingresado por el usuario, no supere el total de tiempo dedicado y transcurrido por la tarea en ejecución.

9.7.1.3 Recolección de indicadores y métricas

Gracias a la trazabilidad del software, los indicadores, y con ello, las métricas, son recolectadas automáticamente por el software y se almacenan principalmente en el formato SUMP. En la ilustración 34 se puede observar el formato SUMP, el cual es posible actualizar a petición del usuario y completando algunos valores manualmente, que al igual que todos los formatos de la metodología, valida cada dato ingresado por el usuario.

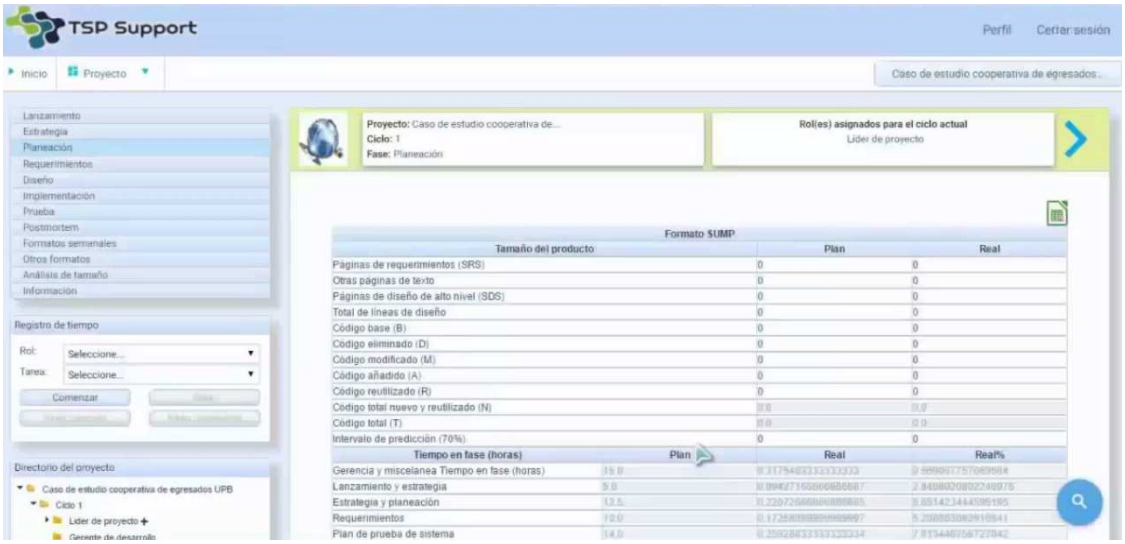


Ilustración 34. Formato SUMP en TSPSupport. Fuente autor

9.7.1.4 Documentación centralizada

Otra de las necesidades fundamentales del desarrollo en equipo TSP corresponde a un repositorio centralizado donde los integrantes puedan subir la documentación asociada al proyecto; todo con el principal propósito de crear y compartir rápidamente la documentación correspondiente a los requerimientos, actas de entrega, entre otros. En la ilustración 35 se puede observar un ejemplo del componente que permite a los usuarios la gestión de los documentos; cada integrante tiene privilegios de agregar, eliminar y descargar los documentos en los directorios de sus roles asignados para el ciclo actual, y únicamente descargar archivos para ciclos anteriores y de otro integrante del equipo, pues el privilegio de agregar y eliminar únicamente le corresponde al dueño del documento.

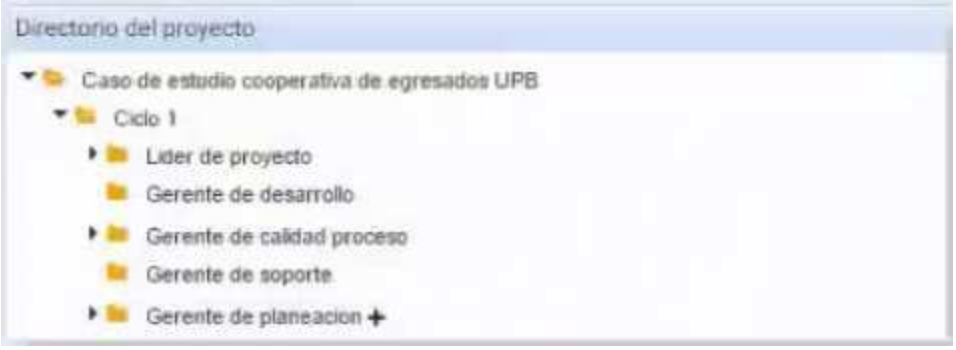


Ilustración 35. Directorio del proyecto en TSPSupport. Fuente autor.

9.7.2 HERRAMIENTAS DE VALOR AGREGADO

Como se ha mencionado anteriormente, no sólo la exactitud y automatización son relevantes en una herramienta de apoyo en la ejecución de un proyecto de desarrollo de software; se requieren herramientas de valor cualitativo que faciliten la implementación de la metodología, sin forzar a los usuarios a utilizarlas, pues la intención es ayudar.

Con base en la experiencia del equipo de desarrollo del prototipo para la gestión de métricas como soporte a la metodología TSP, se propuso brindar tres herramientas adicionales que ofrezcan un servicio auxiliar: Un documento flexible sobre la metodología TSP, de rápida lectura y con información precisa de cada fase y ciclo de desarrollo, un comparador de código para identificar las diferencias entre dos versiones de código y un exportador de datos para manipular a gusto la información recolectada por la herramienta a través de una hoja de cálculo.

9.7.3 Cartilla TSP

Con base en el libro de TSP de Humphrey se construyó una cartilla de tan solo 10 capítulos en los que se describe la metodología a nivel general y las actividades a ejecutar en cada fase de cada ciclo de desarrollo, especificando responsabilidades y criterios de salida de una fase a otra. En el anexo 4 se encuentra dicha cartilla, de la que se destaca su precisión y formato legible a fin de apoyar la comprensión de la metodología, especialmente para estudiantes que están en proceso de aprendizaje de TSP.

9.7.4 Comparador de código

Medir el tamaño del producto software genera costos por la especial atención a los cambios presentados durante el desarrollo; por esta razón se propone una herramienta software capaz de

comparar dos versiones de código a fin de identificar rápidamente las funciones agregadas, eliminadas y modificadas.

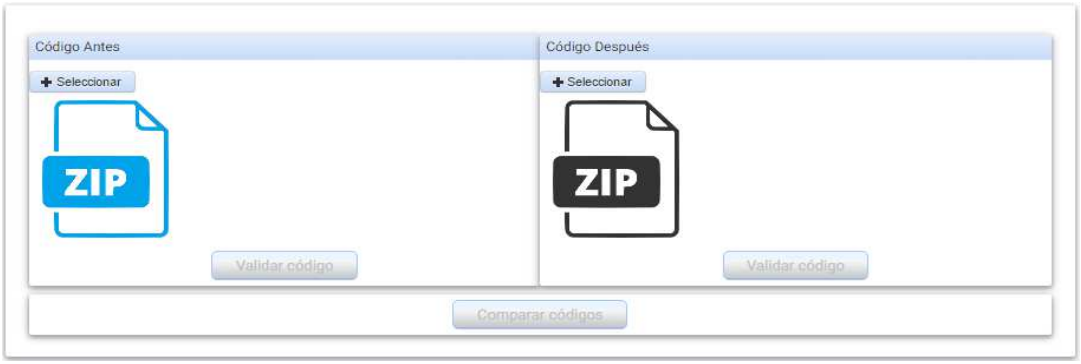


Ilustración 36. Interfaz de comparación de código. Fuente autor

En la ilustración 36 se puede observar la interfaz para agregar las dos versiones de código: *Código Antes* y *Código después,* para validar su contenido y posteriormente comparar los archivos de acuerdo a un lenguaje de programación específico. Esta propuesta brinda comparación de código para lenguaje Java, sin embargo, su diseño permite adicionar cualquier otro lenguaje sin generar grandes costos de implementación.

9.7.5 Exportador de datos

En el prototipo, resultado de este proyecto, se presenta el ícono mostrado en la parte superior derecha de cada formato, tal como se muestra en la ilustración 37 sobre el formato LOGT.



Ilustración 37. Servicio de exportación de datos para el formato LOGT. Fuente autor

Bajo un simple *click,* la herramienta provee los datos del formato TSP en una hoja de cálculo para que los usuarios puedan tomar total control sobre éstos, especialmente para los casos en que deseen documentar las características del proyecto a la fecha.

9.8 PRUEBAS DE PROTOTIPO

Las pruebas sobre el producto se llevaron a cabo en tres escenarios con distintos contextos, dos de ellas como pruebas de validación y la tercera y última como prueba de verificación.

Finalmente, con base en las características evaluadas en [109] (de K.K. Aggarwal), se determina porcentualmente, el grado de completitud y satisfacción del prototipo en su totalidad.

9.8.1 Hardware utilizado

A continuación se describen los dispositivos utilizados en las tres pruebas realizadas al prototipo, asignando a su vez un nombre que los ayude a identificar en posteriores observaciones. Cabe aclarar que se superaron las limitaciones al disponer de un ordenador más para las pruebas.

9.8.1.1 Lenovo G40-30

Es el ordenador portátil donde, además de efectuar el desarrollo del prototipo, se despliega la aplicación para las pruebas, cumpliendo tanto el rol de servidor como de cliente.



Nombre: Lenovo

Procesador: Celeron N2830 2.16 Ghz

Memoria RAM: 8GB

Arquitectura: 64 bits

Sistema operativo: Windows 8.1/Ubuntu 14.04.2

Ilustración 38. Servidor de pruebas y desarrollo.

9.8.1.2 Compaq Mini CQ10-120la

Ordenador portátil cliente con las características mínimas, incluso insuficientes, para oficina.



Nombre: Compaq

Procesador: Intel Atom N270 1.60 Ghz

Memoria RAM: 2GB

Arquitectura: 32 bits

Sistema operativo: Windows 8.1

Ilustración 39. Ordenador portátil cliente

9.8.1.3 Compumax Intel core 2 duo

Ordenador de escritorio, cliente, con características de oficina.



Nombre: Compumax

Procesador: Intel core 2 duo 1.89 Ghz

Memoria RAM: 1GB

Arquitectura: 64 bits

Sistema operativo: Windows 8.1

Ilustración 40. Ordenador de escritorio de cliente.

9.8.1.4 Router ETB ZTE-ZXV10-W300

Router o comúnmente llamado modem, corresponde al dispositivo de comunicación entre ordenadores vía Ethernet y WiFi.



Ilustración 41. Router para conexión LAN de pruebas.

9.8.2 Escenarios

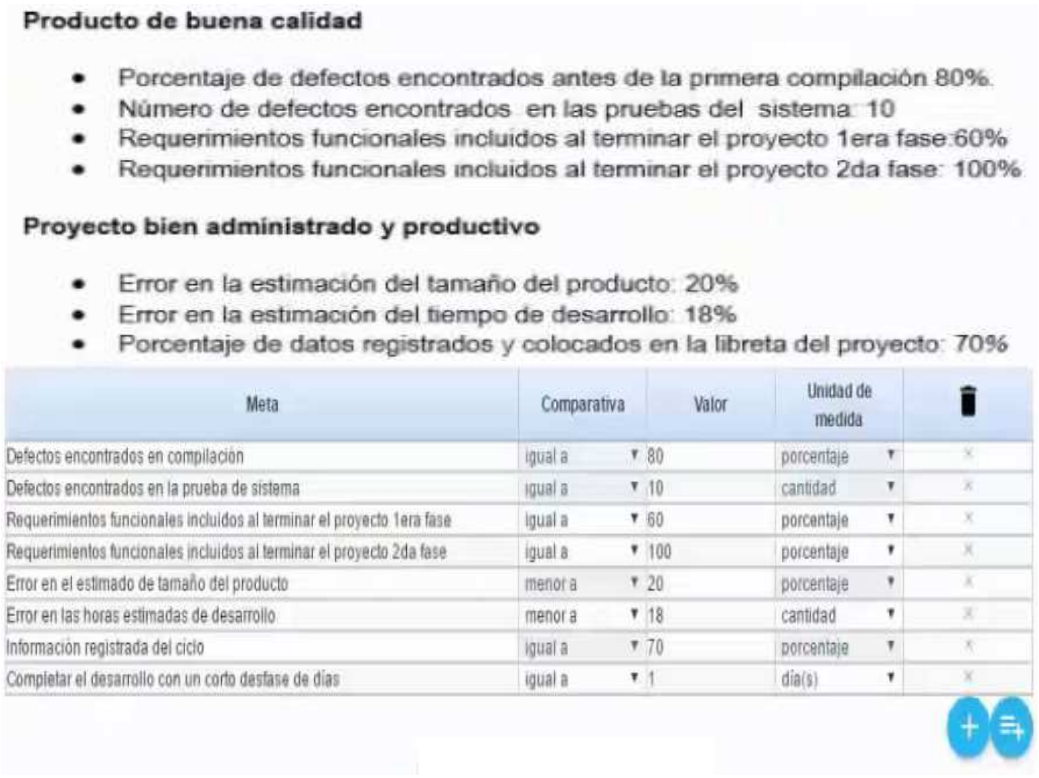
9.8.2.1 Curso de aprendizaje

El prototipo fue alimentado con los datos provenientes de un proyecto desarrollado por estudiantes de un curso de aprendizaje de la metodología TSP. El proyecto, de nombre *Caso de estudio coorperativa de egresados UPB* [111], fue llevado a cabo en el año 2016 con un solo ciclo de desarrollo, en el que se documentó formalmente el proceso de ingeniería y sobre todo, la gestión de las métricas, indicadores y medidas obtenidas a lo largo de la ejecución del proyecto.

A partir de los documentos proporcionados por la instructora de dicho proyecto, se ingresaron los datos del proyecto simulando la ejecución de éste, a fin de garantizar la completitud de las mediciones y con ello la obtención de indicadores y métricas correspondientes. A

continuación se muestran algunas comparaciones entre las vistas de los documentos proporcionados para la prueba y las vistas del software una vez los datos fueron diligenciados en éste.

9.8.2.1.1 Comparación definición de metas



Producto de buena calidad

- Porcentaje de defectos encontrados antes de la primera compilación 80%.
- Número de defectos encontrados en las pruebas del sistema: 10
- Requerimientos funcionales incluidos al terminar el proyecto 1era fase:60%
- Requerimientos funcionales incluidos al terminar el proyecto 2da fase: 100%

Proyecto bien administrado y productivo

- Error en la estimación del tamaño del producto: 20%
- Error en la estimación del tiempo de desarrollo: 18%
- Porcentaje de datos registrados y colocados en la libreta del proyecto: 70%

Meta	Comparativa	Valor	Unidad de medida
Defectos encontrados en compilación	igual a	80	porcentaje
Defectos encontrados en la prueba de sistema	igual a	10	cantidad
Requerimientos funcionales incluidos al terminar el proyecto 1era fase	igual a	60	porcentaje
Requerimientos funcionales incluidos al terminar el proyecto 2da fase	igual a	100	porcentaje
Error en el estimado de tamaño del producto	menor a	20	porcentaje
Error en las horas estimadas de desarrollo	menor a	18	cantidad
Información registrada del ciclo	igual a	70	porcentaje
Completar el desarrollo con un corto destase de días	igual a	1	día(s)

Ilustración 42. Comparación de metas documentadas e ingresadas al software. Fuente autor

La definición de metas pudo llevarse a cabo satisfactoriamente puesto que la herramienta permitió adicionar las metas rápidamente sin excepción alguna y con un alto nivel de amigabilidad. En la ilustración 42 se pueden observar las metas definidas en el documento del plan de calidad que creó el equipo y las metas ingresadas al software.

9.8.2.1.2 Comparación formato STRAT

En la ilustración 43 se pueden observar tres divisiones; las dos primeras corresponden al documento de estrategia del equipo (incompleto, puesto que es de gran tamaño), el cual separó el formato STRAT en dos tablas a diferente nivel de detalle. La tercera división de la imagen unifica los valores ingresados por el equipo al estándar de la metodología TSP en el software.

Módulo	Función	LOC						Total
		25	50	75	100	125	150	
Administración de Afiliados y Familiares	Crear afiliado, familiar				x			425
	Modificar datos del afiliado, activar o desactivar				x			
	Ver detalles			x				
	Listar/Reportar afiliados - Opción de eliminar un afiliado						x	

Módulo	Función	LOC Ciclo		Horas Ciclo	
		1	2	1	2
Administración de Afiliados y Familiares	Crear afiliado, familiar	425		12	
	Modificar datos del afiliado, activar o desactivar				
	Ver detalles				
	Listar/Reportar afiliados - Opción de eliminar un afiliado				

Parte	Función	Pts. funcls	Horas		
		C1	C1		
Afiliados y familiares	Crear afiliado, familiar	100,0	3,0		
	Modificar datos del afiliado, activar o desactivar	100,0	3,0		
	Ver detalles	75,0	3,0		
	Listar/Reportar afiliados - Opción de eliminar un afiliac	150,0	3,0		
Proveedores	Crear proveedor	100,0	2,0		
	Modificar datos	100,0	2,0		
	Ver detalle del proveedor	75,0	2,0		
	Listar proveedores - Opción de eliminar un proveedor	150,0	2,0		
Eventos	Crear evento	125,0	3,0		
	Modificar datos, abrir evento, cerrar evento, cancelar e	150,0	3,0		
	Detalle del evento - número de inscripciones, (total re	125,0	3,0		
	Listar eventos, filtro por estado	150,0	3,0		
Inscripciones	Crear inscripción de afiliado (con o sin grupo familiar)	100,0	2,0		
	Modificar datos, cancelar inscripción	100,0	2,0		
	Detalle de la inscripción	75,0	2,0		
	Listar/Reportar inscripciones - filtros/opciones de cons	150,0	2,0		
Ventana Login	Validar el usuario y el perfil en el sistema (funcionario	150,0	4,0		
Usuarios	Crear usuario	125,0	3,0		
	Modificar contraseña y datos de usuario - Activar/Des	100,0	3,0		
Total		2200,0	50,0		

Ilustración 43. Comparación de formato STRAT documentado e ingresado al software. Fuente autor

La ventaja del software se hace evidente cuando son necesarios los cambios. Si el equipo decide añadir o eliminar una función de algún módulo, requerirá de gran esfuerzo y tiempo, en cambio, en el software, tanto el nuevo registro como los registros dependientes, se ajustarán en forma automática.

9.8.2.1.3 Comparación formato ITL

En la ilustración 44 se puede observar la tabla definida por el equipo de desarrollo para registrar los riesgos del proyecto y un registro de éstos en el software. La clasificación de los registros de la tabla se hizo con colores, y en el software con base en el filtro del nombre del componente, facilitando su interpretación a la vez que brinda la posibilidad de añadir más riesgos a cada uno de éstos sin recurrir a grandes cambios.

Nº	Riesgo	Descripción Riesgo	Plan de Mitigación del Riesgo	Impacto (Alto, medio, bajo)	Probabilidad de ocurrencia	Responsable de seguimiento
1	HUMANO	Personas del equipo de trabajo, sin rol definido claramente en el proyecto.	Definición de roles y responsabilidades de todos los miembros de equipo de acuerdo al plan de trabajo definido.	BAJA	MEDIO	Líder del Equipo
1	HUMANO	Falta de personal en el equipo por algún evento.	Otro miembro del equipo tomará la responsabilidad acorde a la buena documentación que se lleve antes de la eventualidad.	BAJO	MEDIO	Líder del Equipo
2	FINANCIERO	Presupuesto reducido	Seguimiento permanente a las actividades y cumplimientos acorde a lo planeado	MEDIO	ALTO	Líder del Equipo

Formato ITL

Parte	Usuarios					
Fecha	Riesgo/Problema	Número	Prioridad	Dueño	Fecha de seguimiento	Fecha de resolución
11/2/2017	Riesgo	1	L - bajo	1111111111	11/02/2017	11/02/2017

Descripción

Personas del equipo de trabajo, sin rol definido claramente en el proyecto. Definición de roles y responsabilidades de todos los miembros de equipo de acuerdo al plan de trabajo definido.

Ilustración 44. Comparación de formato ITL documentado e ingresado al software. Fuente autor

9.8.2.1.4 Comparación formato TASK

Fase	Parte	Nombre de la Tarea	# de Ingenieros	Líder del Equipo	Administrador de Desarrollo	Administrador de Planeación	Administrador de Calidad / Proceso	Administrador de Soporte *2	Total de Horas del Equipo	Horas Acumuladas	Tamaño Unidad	Tamaño	No. Semana	Valor Planeado	Valor Planeado Acumulado
Lanz	S	Lanzamiento	6	2,50	2,50	2,50	2,50	5,00	15,00	15,00			1	6,02	6,02
Estrat	S	Planeación tamaño	2	0,00	2,50	2,50	0,00	0,00	5,00	20,00			1	2,01	8,02
Estrat	S	Identificación riesgos	4	0,00	0,00	2,50	2,50	5,00	10,00	30,00			1	4,01	12,03
Estrat	S	Administración de la configuración	3	0,00	0,00	2,50	0,00	5,00	7,50	37,50			1	3,01	15,04
Plan	S	Planeación General	6	2,50	2,50	2,50	2,50	5,00	15,00	52,50			1	6,02	21,06
Plan	S	Plan de Calidad	3	2,50	0,00	2,50	2,50	0,00	7,50	60,00			1	3,01	24,07
Plan	S	M&M	6	2,00	3,00	1,50	2,50	2,00	11,00	71,00			1	4,41	28,48
Req	S	Especificación de Requerimientos(ERS)	6	3,00	2,00	2,00	2,00	4,00	13,00	84,00			1	5,21	33,69
Req	S	Inspección ERS	5	1,00	2,00	0,00	2,00	4,00	9,00	93,00			1	3,61	37,30
Req		Plan de Pruebas del Sistema	4	0,00	2,00	2,00	0,00	4,00	8,00	101,00			1	3,21	40,51
Req		Inspección Plan de pruebas del	5	1,00	2,00	0,00	2,00	4,00	9,00	110,00			1	3,61	44,12

Ilustración 45. Comparación de formato TASK documentado e ingresado al software. Fuente autor

En la ilustración 45 se puede observar una muestra del formato TASK realizado por el equipo de desarrollo. El formato TASK resultado del software se puede observar en el anexo 5, el

cual tiene ligeras diferencias relacionadas a la fácil gestión de los cambios (agregar, remover y editar tareas).

Este formato en particular presenta varias mejoras, pues a falta de una herramienta que guiara el diligenciamiento del formato TASK, el equipo tuvo que construir una hoja de cálculo por cada rol, diligenciarlo en forma individual y finalmente unificar los datos en un único formato de equipo. Al diligenciar los datos en la herramienta, bastó con diligenciar un único formato TASK capaz de actualizar los datos en tiempo real, evitando errores en las mediciones y brindando posibilidades de cambio sin recurrir a enormes esfuerzos de versionado y descontrol de información al no poseer una interfaz capaz de enfocar la atención de los integrantes.

9.8.2.1.5 Comparación formato SUMP

Dado el tamaño del formato SUMP, la ilustración 46 muestra únicamente el formato SUMP diligenciado por el equipo de desarrollo. En el anexo 5 se puede localizar el formato SUMP obtenido, con variaciones en los valores dadas las dificultades por obtener exactamente los mismos resultados en simulación.

Program Size	**Plan**	**Actual**	
Total Requirements Pages (SRS)	5	10	
Total HLD Pages (SDS)	3	4	
Total Detailed Design paginas	7	10	
Base LOC (B)	0	0	
Deleted LOC (D)	0	0	
Modified LOC (M)	0	0	
Added LOC (A)	0	0	
Reused LOC (R)	0	0	
New and Changed LOC (N)	1550	1600	
Total LOC (T)	1550	1600	
Total New Reuse LOC	0	0	
Estimated Object LOC (E)	0	0	
Upper Prediction Interval (70%)	0	0	
Lower Prediction Interval (70%)	0	0	
Time in Phase (hours)	**Plan**	**Actual**	**Actual%**
Management and Miscellaneous	8,0	24,45	14,2%
Launch and Strategy	15,0	14,5	8,4%
Planning	15,0	13,8	8,0%
Requirements	13,0	8,9	5,2%
System Test Plan	8,0	8,0	4,7%
REQ Inspection	9,0	9,5	5,5%
High-Level Design	10,0	7,7	4,5%
Integration Test Plan	12,2	8,9	5,2%
HLD Inspection	10,0	8,0	4,7%
Detailed Design	9,0	9,5	5,5%
DLD Review	10,0	7,7	4,5%
Test Development	0,0	0,0	0,0%
DLD Inspection	6,8	4,8	2,8%

Ilustración 46. Comparación de formato SUMP documentado e ingresado al software. Fuente autor

La herramienta software, para el caso de SUMP, provee únicamente los campos a diligenciar por el usuario para centrar su atención en determinados aspectos, como lo es la planeación, pues las mediciones, indicadores y métricas reales se gestionan automáticamente.

9.8.2.1.6 Comparación formato SUMQ

Es muy similar a SUMP al momento de comparar. Parte del formato proporcionado por el equipo de desarrollo se muestra en la ilustración 47 y el resultado obtenido mediante el software se muestra en el anexo 6. Así, también se brindan los campos específicos para los usuarios y se gestiona automáticamente gran parte de la información recolectada.

SYSTEM Percent Defect Free		**Actual**
In Compile		0,0%
In Unit Test		0,0%
In Build and Integration Test		0,0%
In System Test		0,0%
In Acceptance Test		0%
In Product Life		0%
Defects/Page	**Plan**	**Actual**
REQ Inspection		0,00
HLD Inspection		0,00
Defects/KLOC	**Plan**	**Actual**
DLD Review	1,00	0,00
DLD Inspection	0,65	0,63
Code Review	5,16	5,00
Compile	7,74	7,50
Code Inspection	1,94	1,88
Unit Test	2,58	2,50
Build and Integration Test	1,29	1,25
System Test	0,00	1,88
Total Development	32,90	28,75
Acceptance Test	0,00	0,00
Product Life	0,00	0,00
Total	32,90	28,75
Defect Ratios	**Plan**	**Actual**
DLD Review/Unit Test	0,50	0,00
Code Review/Compile	0,67	0,67

Ilustración 47. Comparación de formato SUMQ documentado e ingresado al software. Fuente autor

9.8.2.1.7 Comparación formato LOGT

En la ilustración 48 se puede observar la comparación entre los formatos LOGT del equipo de proyecto y de la aplicación. Básicamente ofrecen la misma información; lo importante en esta comparación es reconocer el grado de dificultad de construcción. Se desconoce exactamente qué metodología personal ejecutó cada integrante del equipo para controlar el tiempo de sus tareas, sin

embargo, se tuvieron que registrar manualmente los tiempos, asegurarse de que las tareas sean las definidas por el equipo y marcar, con alta probabilidad de errores, si la tarea fue finalizada o no.

Estudiante	JAVIER ENRIQUE MORA DIAZ					Fecha	2016/05/28		
Profesor	ALBA CONSUELO NIETO					Clase	GESTIÓN DE CALIDAD DE SOF		
Fecha	Inicio	Fin	(minutos)	Delta (minutos)	Fase/Tarea	omponent	Comentarios	C	U
2016/05/16	13:30	22:00	0	150	Lans / Lanzamiento				1
2016/05/17	16:00	18:30	15	135	Plan / Planeacion General		Descanso	x	1
2016/05/17	18:45	21:30	30	135	Plan /Plan de Calidad		Alimentacion	x	1
2016/05/17	21:15	23:15	0	120	Plan / M & M			x	1
2016/05/18	7:00	10:00	0	180	Req / Especificación de Requerimientos			x	1
2016/05/18	10:00	11:00	0	60	Req / Inspección ERS			x	1
2016/05/18	20:30	21:30	0	60	Req / Inspección Plan de pruebas del sistema			x	1
2016/05/19	7:00	8:00	0	60	Dis / Estándares Diseño Codificacion Glosario			x	1
19/05/2016	16:00	17:00	0	60	Dis / Inspección de EDS			x	1
2016/05/19	20:00	22:00	30	90	Dis / Inspección de diseño Detallado		Descanso	x	1

Formato LOGT

Rol	Gerente de desarrollo
Integrantes	2222222222

Tarea	Fecha-hora de inicio	Fecha-hora de finalización	Tiempo dedicado	Tiempo interrumpido
Gestión documental	2017-02-16 13:59:58.697	2017-02-16 14:00:03.143	150	0.0741

Comentarios (opcional)

Tarea	Fecha-hora de inicio	Fecha-hora de finalización	Tiempo dedicado	Tiempo interrumpido
Lanzamiento	2017-02-16 14:00:15.032	2017-02-16 14:00:20.408	150	0.0896

Comentarios (opcional)

Ilustración 48. Comparación de formato LOGT documentado e ingresado al software. Fuente autor

9.8.2.2 Simulación en red LAN

Con esta prueba se pretendió evaluar la herramienta prototipo en el contexto más real posible, sin embargo, dadas las limitaciones de hardware, el caso ideal de 5 o 6 ordenadores y un servidor no fue posible llevarlo a cabo, y por tanto, se procedió a gestionar tres roles en el servidor Lenovo (que también se utilizó como cliente) y dos roles en el equipo cliente Compumax.

Además de recolectar información sobre los defectos presentados en la herramienta, se obtuvo una comprensión empírica del desempeño del sistema, teniendo en claro que el servidor no cuenta con las características mínimas para ejecutar dichas tareas, y por tanto tuvo que forzar sus recursos para responder a las peticiones presentadas en el software.

La ilustración 49 representa el escenario sobre el cual se hizo la prueba en una red local y la ilustración 50 evidencia el escenario real sobre el que se llevó a cabo la prueba. La distribución de roles sobre cada uno de los ordenadores se hizo como se muestra en la tabla 54.



Ilustración 49. Diagrama representativo de conexión LAN. Fuente autor.

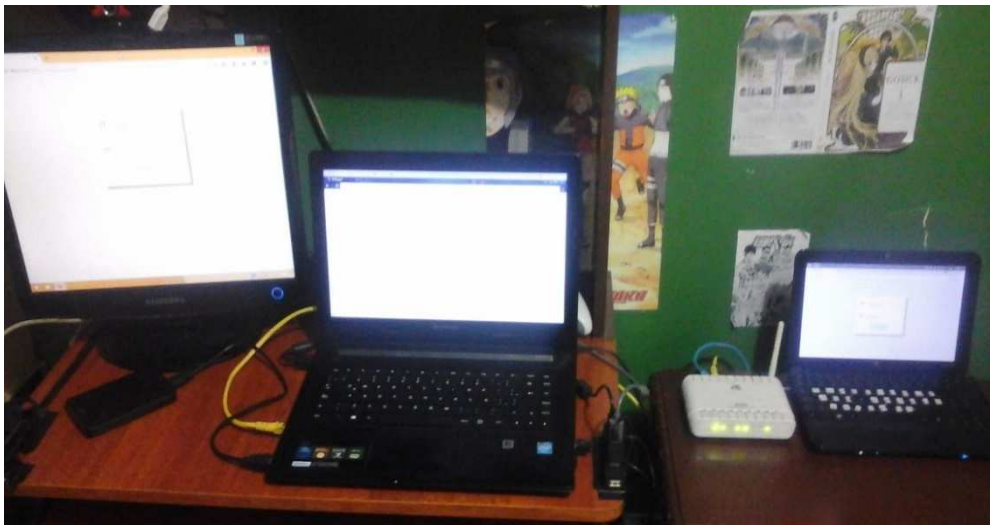


Ilustración 50.Montaje real de prueba en LAN. Fuente autor.

En síntesis, la ejecución de esta prueba fue satisfactoria al permitir ejecutar todas las tareas de la prueba anterior y sobre todo, permitir gestionar las métricas y culminar el proyecto. Los resultados a detalle sobre el rendimiento y defectos encontrados, se describen en forma precisa en el anexo 5.

Tabla 54. Ordenador sobre el que se gestionaron los datos de cada rol. Fuente autor

Ordenador	Rol
LENOVO	Líder de equipo
COMPAQ	Gerente de planeación
COMPUMAX	Gerente de calidad/proceso
LENOVO	Gernte de soporte
LENOVO	Gerente de desarrollo
COMPUMAX	Instructor

9.8.2.3 Verificación final

Como última prueba se realizó una simulación de proyecto con la finalidad de refinar detalles asociados a la amigabilidad, el apoyo a los usuarios de la herramienta y algunas funcionalidades involucradas.

Considerando la última prueba como un proceso de validación, donde el cliente es representado por la docente *Alba Consuelo Nieto* (especial conocedora de las bondades de la metodología TSP), quien concluyó que la herramienta satisface en gran medida las necesidades de la metodología TSP al grado que “no se puede considerar un prototipo, sino la versión 1 de un software en producción”.

A continuación se enlistan los cambios solicitados por la docente desde el punto de vista del cliente y alineándose a la metodología TSP propuesta por Humphrey.

- Vista del logo del grupo de investigación y de la institución universitaria. Adicionar información acerca del grupo de investigación
- Brindar información de ayuda sobre la elección del día de revisión semanal
- Solicitar el número de teléfono de los integrantes
- Brindar la información de gerencia y miscelánea al final del formato TASK.
- Brindar ayuda en las tareas asignando el nombre de la etapa
- Permitir asignar un tiempo para las tareas que no se deseen implementar
- Redondear los valores calculados para facilitar su lectura
- Asegurar el nombre de cada formulario en el encabezado de éstos.

Una vez se realizaron los cambios solicitados, el software se desplegó en un servidor de la comunidad universitaria bajo la intención de que los estudiantes tomen conciencia de la importancia de medir su trabajo, para con ello desarrollar disciplina y mejorar continuamente sus procesos que los permitan destacar a nivel general, y en especial, a nivel laboral.

9.8.3 Cuantificación de las pruebas

Las pruebas realizadas al prototipo resultante se basaron en la filosofía de K.K. Aggarwal [109] la cual consiste en dejar de pretender que las pruebas se fundamentan en demostrar que el software funciona, sino que por el contrario, partir de la idea que el software no cumple ninguna de las funcionalidades esperadas y con base en ello, evaluar cómo responde éste a cada solicitud del cliente.

De acuerdo a la clasificación de pruebas expuesta en el libro *Software Engineer* [109], se realizaron las pruebas descritas en la tabla 55, en la que se especifica el propósito de éstas para el prototipo en específico, resultado de este proyecto.

Tabla 55. Pruebas funcionales y no funcionales sobre el prototipo. Fuente autor

Pruebas funcionales	
Transición de estados	Indica los estados por los que pasa la aplicación. Especifica las fases del proyecto y el estado de los formatos al respecto y con base en los roles saignados.
Tablas de decisión	Describe las acciones que debe tomar el software con base en los criterios a evaluar. Todas las decisiones sobre los roles, metas, formatos, tiempos, entre otros, son probados en este apartado.

	Partición equivalente	**Límites**
Comportamiento	Comprueba casos que son exactamente iguales para otros usuarios. Se utilizael primer escenario descrito en el apartado anterior para probar el funcionamiento del software para un proyecto y con ello, garantizar la funcionalidad para todos los proyectos.	Pone a prueba el software bajo condiciones de gran cantidad de datos o por el contrario poca cantidad. Para el prototipo, cuanto más proyectos tenga un usuario, más caótica puede llegar a considerarse su gestión, por tanto se hicieron pruebas sobre una cantidad grande de proyectos.
Pruebas no funcionales		
Técnicas de pruebas estructurales	**Inyección de defectos**	**Estático**
	Se centra en ingresar datos errados o fuera de contexto para comprobar la respuesta del software frente a ello. Para esta prueba se ingresaron datos errados a fin de dañar las métricas a recolectar.	Consiste en comprobar si los componentes del software cumplen principios y patrones. Puesto que desde un principio se buscó la calidad del producto mediante la utilización de patrones y buenas prácticas, se evaluaron funciones y distribución con base en el propósito de cada componente.
Calidades sistémicas	**Manifiestas**	
	Rendimiento	Consta de la revisión de tiempos de respuesta. Este caso en particular se revisó sobre el segundo escenario.
	Usabilidad	Trata de garantizar la amigabilidad e intuitividad de la herramienta. Dada la subjetividad de éste aspecto, se recurrió a la cuantificación propuesta en el artículo [108].
	Operacionales	
	Compatibilidad	**Internacionalización**
		Corresponde a la facilidad de transformación del software de un lenguaje a otro. Se localizaron los archivos y se estimó el costo de edición.

Cada una de las pruebas descritas en la tabla 55 se exponen a detalle en el apartado 5 donde se cuantificó porcentualmente cada criterio y con ello asignar el grado de satisfacción del software a los requerimientos de los clientes, principalmente centrados en la automatización, el control y la usabilidad.

9.8.4 Resultados

Con base en los detalles de las pruebas realizadas del anexo 5, se construye la tabla 56 que resume el logro de cada criterio a nivel porcentual y por el cual se deduce que el software, en su totalidad, provee un 95,81% de satisfacción de los requerimientos y necesidades de los clientes.

De forma cualitativa se afirma que el software provee la calidad esperada; esto se debe a que las medidas, indicadores y métricas recolectados satisfacen los niveles de exactitud, automatiza la información recolectada, no requiere grandes esfuerzos ni de utilización ni aprendizaje y provee un alto grado de amigabilidad.

Tabla 56. Porcentajes de acierto a las pruebas sobre el prototipo. Fuente autor

Pruebas funcionales							
Transición de estados	**Transición de fases**	**Disponibilidad de fases**	**Modificaciones sobre formatos**	**Rol dominante**	**Formatos transversales**	**Cumplimiento de metas**	**Promedio**
	100	100	98,5	91,7	92,3	100	97,0833333
	Usuario	**Proyecto**	**Asignación de roles**	**Agregar meta**	**Formato STRAT**	**Formato ITL**	
	100	94,4	100	90	95	75	
	Formato TASK	**Formato LOGT**	**Formato SCHEDULE**	**Formato WEEK**	**Formato LOGD**	**Formato CCR**	
	100	100	100	100	100	100	
	Formato PIP	**Formato PEER**	**Formato SUMP**	**Formato SUMQ**	**Confirmación**		
Tablas de decisión	100	100	100	100	91,6		96,8235294
	Partición equivalente			**Límites**			
Comportamiento	100			100			100
Pruebas no funcionales							
Técnicas de pruebas estructurales	**Inyección de defectos**			**Estático**			
	81,8			100			90,9
Calidades sistémicas	**Manifiestas**						
	Rendimiento			95			
	Usabilidad			92			
	Operacionales						
	Compatibilidad	**Internacionalización**					
		90					94,25
						Promedio total	95,81

Este prototipo además, innova en el hecho mismo de ser una necesidad al momento de implementar un proyecto de desarrollo de software bajo la metodología TSP, y no representar una carga en los procesos. Esto sucede porque, tanto la automatización como el control de datos, satisface inmediatamente los parámetros de la propuesta de Humphrey, evitando fallas en las mediciones y mal diligenciamiento de los formatos. Transitivamente la herramienta mitiga las tres principales desventajas de la metodología mencionadas en el apartado 7.4.1: recolección automática de información, documentación centralizada y amigabilidad con el usuario.

En cuanto a la totalidad de artefactos desarrollados, la tabla 57 muestra cada uno de ellos de acuerdo al contexto, permitiendo conocer, en resumen, la complejidad del desarrollo.

Tabla 57. Cantidad de artefactos desarrollados

Artefacto	***Cantidad***
Historias de usuario	*22*
Clases de negocio	*14*
Patrones de diseño	*4*
Diagramas de secuencia	*20*
Tablas del modelo relacional	*12*
Documentos del modelo no relacional	*12*
Diagramas de clase por contexto	*7*
Diagrama de casos de uso	*6*
Otros diagramas	*4*

Finalmente, el prototipo resultante provee el flujo de trabajo mostrado en la ilustración 51, donde se identifica la secuencialidad de los pasos realizados en el software durante el desarrollo de un proyecto. Este diagrama también muestra las actividades que se pueden ejecutar en cualquier momento en paralelo, por ejemplo la gestión de la documentación o presentar los resultados cualitativos, pues ellos dependen de los valores actualmente registrados en el sistema.

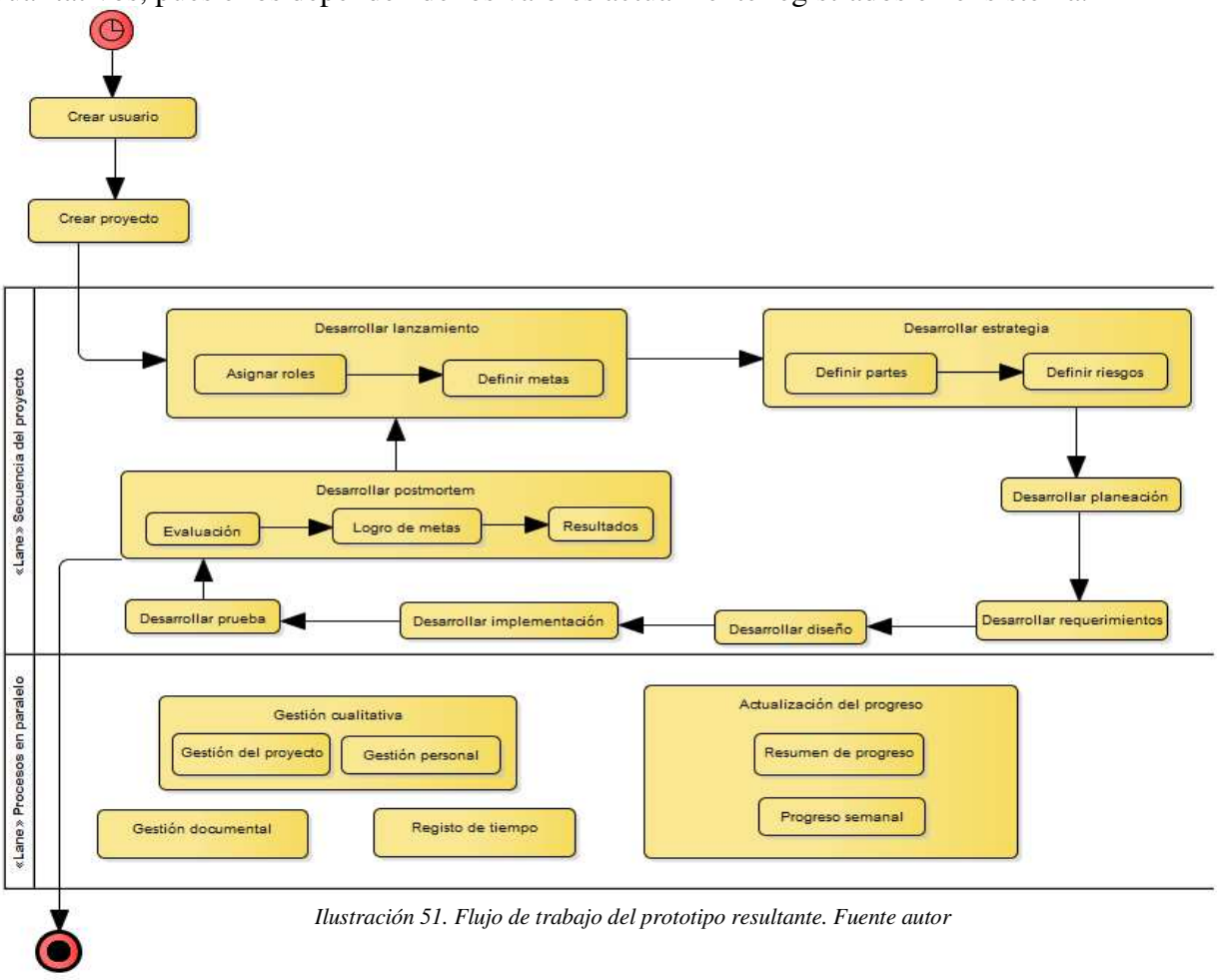


Ilustración 51. Flujo de trabajo del prototipo resultante. Fuente autor

10. CONCLUSIONES

La herramienta prototipo puede satisfacer las necesidades de los equipos de desarrollo de software, sobre la metodología TSP, en un estimado del 95,8% (de acuerdo a la tabla 50). Se logró proporcionar mecanismos de diligenciamiento automático y semiautomático de los formatos TSP con la amigabilidad que ofrecen las tecnologías actuales sobre la capa de presentación y el correspondiente control de datos que aseguran el correcto desarrollo de la metodología.

Se proporcionó satisfactoriamente el acceso remoto y centralizado a las métricas recolectadas a través de los formatos TSP y a la documentación para el libre acceso en un proyecto en específico. Esta documentación se deposita en un directorio ordenado por proyecto, ciclos y roles, brindando una gestión intuitiva para la carga, descarga, interpretación y eliminación de archivos y carpetas.

Gracias al modelo de datos (relacional y documental) y la arquitectura de la aplicación, es posible dar soporte a equipos TSP de diferentes tamaños y ejecutar proyectos en forma independiente al contexto académico y/o empresarial. Asimismo, las unidades de medida se hacen independientes a la recolección de información de cada proyecto, ya que tanto la herramienta como la metodología recolectan las métricas principalmente basadas en el tiempo y tamaño, de forma proporcional a las unidades de medida determinadas por el equipo de desarrollo.

Además de medir los datos recolectados con exactitud, el prototipo provee a los usuarios el mayor grado de proximidad a los procesos reales, especialmente para el control de tiempo. Los integrantes de un equipo TSP pueden marcar los tiempos sobre cada tarea fácilmente, sin recurrir a herramientas como hojas de cálculo, donde es frecuente encontrar retrasos por motivos de forma y control personal de tiempo. La herramienta asegura que los integrantes registren el tiempo real de sus tareas y por tanto, no adulteren la información a conveniencia, a fin de retroalimentar con base en información útil para los mismos.

Con los logros mencionados sobre el producto software resultante, se afirma que la herramienta colabora en el proceso de desarrollo de software mediante la metodología TSP sin grandes costos. Por el contrario, guía a los integrantes del equipo, solicitando la información de

acuerdo a los roles que corresponda y controlando los datos ingresados para asegurar la correcta ejecución de la metodología.

Con la automatización y la amigabilidad de la herramienta, los ingenieros podrán centrarse en los procesos específicos del software a desarrollar, al tiempo que obtienen una cultura disciplinaria de bajo costo. Específicamente en Colombia, el producto software puede motivar a las organizaciones a obtener certificaciones de calidad y trabajar en equipo (clusters) a fin de adquirir mayor reconocimiento y por ende, mayor competitividad en el mercado. De igual manera, estudiantes del área de tecnologías de la información podrán comprobar empíricamente los beneficios de la calidad en el desarrollo de software a tempranas etapas de su aprendizaje, y con ello, acelerar el crecimiento del país en términos de calidad en el software.

11. TRABAJOS A FUTURO

Teniendo en cuenta el diseño y el producto resultante, se proponen los siguientes trabajos a futuro para extender y/o agregar funcionalidades al software:

- Aumentar la cobertura de comparación del tamaño del software. Actualmente el comparador de código permite identificar las diferencias entre dos versiones de código, únicamente en lenguaje Java, sin embargo se pueden añadir otros lenguajes sin grandes costos de implementación.
- Agregar un sistema de publicaciones o foro. Con este componente se puede establecer un modo de comunicación por medio de la misma herramienta software.
- Agregar un sistema de reuniones de modo que se puedan agendar los encuentros entre los integrantes del equipo.

12. GLOSARIO

FEDESOFT. Federación Colombiana de la Industria del Software y Tecnologías Informáticas Relacionadas. Es una organización gremial que trabaja por el fortalecimiento de la industria del software nacional para que sea competitiva y exitosa a nivel internacional.

Hangouts. Hangouts es una aplicación multiplataforma de mensajería instantánea desarrollada por Google Inc. Se creó para sustituir los servicios Google Talk, Google+ Messenger y Google+ Hangouts, unificando todos estos servicios en una única aplicación.

ISO. Es una organización independiente, no gubernamental con una participación de 163 cuerpos estándares nacionales. Mediante sus integrantes, reúne expertos para compartir conocimiento y desarrollar Estándares Internacionales voluntarios, consensuados y relevantes para el mercado que apoyan la innovación y proveen soluciones a retos globales.

KLOC. Mil líneas de código. Referirse a definición de LOC.

Labels. Es un componente que permite incluir palabras, letras o frases en pantalla de forma visual. Este componente se visualiza directamente en el momento de creación de un programa y su manipulación es directa sobre la pantalla, y se le asignan unas propiedades que afectan a su contenido.

LOC. Línea de Código. Es una métrica de software utilizada para medir el tamaño de un programa de software mediante el conteo del número de líneas en el texto del código fuente del programa. Se utiliza comúnmente para predecir la cantidad de esfuerzo que se requerirá para desarrollar un programa, así como estimar la productividad de la programación o el esfuerzo una vez que el software sea producido.

Loops. Es una sentencia que se realiza repetidas veces a un trozo aislado de código, hasta que la condición asignada a dicho bucle deje de cumplirse. Generalmente, un bucle es utilizado para hacer una acción repetida sin tener que escribir varias veces el mismo código, lo que ahorra tiempo, procesos y deja el código más claro y facilita su modificación en el futuro.

Métrica. Es cualquier medida o conjunto de medidas destinadas a conocer o estimar el tamaño u otra característica de un software o un sistema de información, generalmente para realizar comparativas o para la planificación de proyectos de desarrollo de software.

MinTIC. Ministerio de Tecnologías de la Información y las Comunicaciones. Es la entidad que se encarga de diseñar planes y políticas para que la tecnología llegue a todos los departamentos y ciudades de Colombia.

TI. Tecnología de la información. Es la aplicación de ordenadores y equipos de telecomunicación para almacenar, recuperar, transmitir y manipular datos, con frecuencia utilizado en el contexto de los negocios u otras empresas.

Prototipo. Primer ejemplar que se fabrica de una figura, un invento u otra cosa, y que sirve de modelo para fabricar otras iguales, o molde original con el que se fabrica.

PyMES. Son agentes con lógicas, culturas, intereses y espíritu emprendedor específicos. MiPyME es la expansión del término original, en donde se incluye a la microempresa.

Proceso. Conjunto de fases sucesivas de un fenómeno o hecho complejo.

SEI. Software Engineering Institute. Es un centro de desarrollo e investigaciones operado por una de las universidades más importantes en Estados Unidos, Carnegie Mellon University (CMU) y es patrocinado por el Departamento de Defensa de ese país.

13. BIBLIOGRAFÍA

[1] S. E. Rojas and J. J. Borja, "Calidad del software: Camino hacia una verdadera industria del software," Rev. Esc. Adm. Negocios, vol. 38, pp. 38 – 57, 1999.

[2] A. . Alarcón, J. S. González, and S. L. Rodríguez, "Guía para pymes desarrolladoras de software, basada en la norma ISO/IEC 15504," Rev. Virtual Univ. Católica del Norte, vol. 1, no. 34, pp. 285–313, 2011.

[3] R. García, "Capítulo 2. modelos y estándares de calidad de software 2.1," 1998. [Online]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/garcia_r_ci/capitulo2.pdf.

[4] G. Cuevas, J. Calvo Manzano, T. San Feliu, J. Mejia, M. Muñoz, and S. Bayona, "Impact of TSP on software projects," Electron. Robot. Automot. Mech. Conf. CERMA 2007 - Proc., pp. 706–711, 2007.

[5] Dinero.com, "Oscuro Contrato de Movilidad," 2014. [Online]. Available: http://www.dinero.com/pais/articulo/dudosa-prorroga-contrato-movilidad/196645.

[6] W. Humphrey, Introduction to the Personal Software Proces. Nueva Jersey: Addison-Wesley, 1997.

[7] W. Humphrey, Introduction to the Team Software Process. Nueva Jersey: Addison-Wesley, 2000.

[8] I. de I. de Software, "Instituto de Ingeniería de Software." [Online]. Available: sei.cmu.edu.

[9] InterSoftware, "Formación Especializada en PSP/TSP," 2012. [Online]. Available: http://www.intersoftware.org.co/content/formacion-especializada-en-psptsp.

[10] F. Scalone, "Estudio comparativo de los modelos y estándares de calidad del software," Universidad Tecnológica Nacional, 2006.

[11] MEDIUM, "Certificaciones : ¿Cómo está avanzando el sector TI colombiano ?" p. 8, 2015.

[12] Cemiot, "Problemas durante la implementación de ISO 9001," 2012. [Online]. Available: http://www.cemiot.com/inicio/problemas-durante-la-implementacion-iso-9001/.

[13] B. R. Von Konsky and M. Robey, "A case study: GQM and tsp in a software engineering capstone project," Proc. - 18th Conf. Softw. Eng. Educ. Training, CSEE T 2005, pp. 215–222, 2005.

[14] S. A. Fahmi and H. J. Choi, "A survey on team software process supporting tools," Proc. - 3rd Int. Conf. Converg. Hybrid Inf. Technol. ICCIT 2008, vol. 1, pp. 987–990, 2008.

[15] L. E. Pelaez, "Certificación de la calidad del proceso y producto : ruta para PyMES colombianas," Rev. Vent. Inform. Fac. Ing. Univ. Manizales, pp. 41–61, 2011.

[16] Y. Hongyun, B. Xiaohong, and Y. Shunkun, "Research and improvement of team software process," 2009 WRI World Congr. Comput. Sci. Inf. Eng. CSIE 2009, vol. 7, pp. 654–658, 2009.

[17] M. G. Estayno, G. N. Dapozo, L. R. Cuenca Pletsch, and C. L. Greiner, "Modelos y métricas para evaluar calidad de software," XI Work. Investig. en Ciencias la Comput., no. 1, 2009.

[18] N. L. García, "Definiciones." Instituto Tecnológico Superior de Lerdo, 2012.

[19] S. Autor, "Portal ISO 25000," 2015. [Online]. Available: http://iso25000.com/.

[20] INTECO, "Estudio sobre la certificación de la calidad como medio para impulsar la industria de desarrollo del software en España." INTECO, p. 147, 2008.

[21] M. Blanco, "Aseguramiento de la calidad de Software en Administraciones Públicas," Rev. Española Innovación, Calid. e Ing. Softw., vol. 7, no. 2, pp. 48 – 51, 2011.

[22] S. Henao, "Qué es la calidad de software." 2010.

[23] Ministerio del Fomento. Sin fecha. Calidad (Nivel 1). http://www.pasaiaport.eus/images/el-puerto/autoridad-portuaria/ofertas-empleo/responsable-economico-financiero/CALIDAD.pdf.

[24] López, AM., Cabrera, C., Valencia, LE. 2008. Introducción a la Calidad de Software. Scientia et Technica. 14:39. pp. 326 – 331.

[25] Lomprey, G., Hernández, S. 2008. La importancia de la calidad en el desarrollo de productos de software. Universidad de Montemorelos. pp. 13.

[26] Bertoa, MF., Troya, JM., Valecillo, A. 2002. Aspectos de Calidad en el Desarrollo de Software Basado en Componentes. Universidad de Málaga. pp. 18.

[27] Instituto Politécnico Nacional. Sin fecha. 2.3 Métricas de Software. http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_proceso/ANALISIS_Y_DISEnO_DE_SISTEMAS/IngenieriaDeSoftware/CIS/UNIDAD%20II/2.3.HTM.

[28] Islas, JM. 2010. Métricas para una Dirección de Proyectos Exitosa. Guadalajara México Chapter. www.pmigdl.org/conferencias/PMIGDL-Conf-03-2010.pdf.

[29] Pereira, B., et al. Sin fecha. Métricas de Calidad de Software. http://ldc.usb.ve/~abianc/materias/ci4712/metricas.pdf.

[30] Pérez, C. 2012. Métricas para el desarrollo de Software. https://prezi.com/gqnnssmdi3i2/metricas-para-el-desarrollo-de-software/.

[31] Fernández, JL. Sin fecha. Midiendo ITIL®: Métricas e indicadores para la Gestión de Servicios TI. Proactivanet. https://www.proactivanet.com/UserFiles/File/.../Metricas%20ITIL.pdf.
[32] C. Vargas, "APLICACIÓN BASADA EN COMPONENTES DE SOFTWARE PARA EL APOYO A LA CAPTURA DE MÉTRICAS DE PROCESO EN PROYECTOS DE DESARROLLO DE SOFTWARE," 2015.
[33] Olsina, L. Sin fecha. Métricas e Indicadores: Dos conceptos claves para la Medición y Evaluación. Departamento de Ciencias de la Computación. Universidad de Chile. http://www.ciw.cl/recursos/Charla_Metricas_Indicadores.pdf.
[34] Pinilla, N. 2015. La Diferencia entre medición, métrica e indicador. http://bpmprocesos.blogspot.mx/2015/06/diferencia-medicion-metrica-indicador.html
[35] J. L. Cendejas, "Implementación del modelo integral colaborativo (MDSIC) como fuente de innovación para el desarrollo ágil de software en las empresas de la zona centro-occidente en México," Universidad Popular Autónoma del Estado de Puebla, 2014.
[36] Universidad Técnica Federico Santa María. Sin fecha. Métricas Clásicas de Software. https://www.inf.utfsm.cl/~visconti/iswav/documentos/MetricasClasicasDeSoftware.pdf.
[37] O. Gómez, "Estudio del Proceso Software Personal (PSP) en un entorno académico," ReCibe, vol. 3, no. 2, p. 28, 2014.
[38] W. Humphrey and J. Over, "The Personal Software Process (PSP) tutorial," Proc. - 27th Annu. NASA Goddard Softw. Eng. Work. SEW 2002, pp. 39–92, 2002.
[39] E. E. de Excelencia, "Adaptación a la nueva norma ISO 9001:2015," 2015. [Online]. Available: http://www.nueva-iso-9001-2015.com/.
[40] S. Colombia, "Revisión de ISO 9001:2015," 2015. [Online]. Available: http://www.sgs.co/es-ES/Health-Safety/Quality-Health-Safety-and-Environment/Quality/ISO-9001-2015-Revision.aspx.
[41] DNV GL, "Gestión De La Calidad Iso 9001 : 2015." p. 2, 2015.
[42] BSI, "Revisión ISO 9001:2015," 2015. [Online]. Available: http://www.bsigroup.com/es-MX/gestion-de-calidad-ISO-9001/revision-ISO-9001/.
[43] CMMI Institute, "CMMI Institute," 2016. [Online]. Available: http://cmmiinstitute.com/.
[44] M. B. Chrissis, M. Konrad, and S. Shrum, Desarrollo y Gestión de Requerimientos. 2009.
[45] European Software Institute, "IT Mark," 2011. [Online]. Available: http://it-mark.eu/wordpress/?lang=es.
[46] M. Emaldi, I. Garro, and M. Gorostidi, "I.T.Mark – Modelo de Mejora de Procesos para PyMes de T.I." p. 8, 2006.
[47] ArquitecSOFT, "Arquitecsoft es seleccionado por el MINITIC para implementar el modelo de calidad internacional IT Mark®," 2014. [Online]. Available: http://arquitecsoft.com/portal/index.php/noticias/83-it-mark.
[48] RELAIS, "IT Mark," 2016. [Online]. Available: http://relais.sic-learning.com/Servicios/Productores/Certificaciones/IT_Mark.
[49] Llaneza, M., et al. 2013. Análisis comparativo de modelos de calidad orientado al desarrollo de software en PyMes. XV Workshop de Investigadores en Ciencias de la Computación. 601 – 605 p.
[50] M. A. Serrano, C. M. De Oca, and K. Cedillo, "An experience on using the team software process for implementing the Capability Maturity Model for software in a small organization," Proc. - Int. Conf. Qual. Softw., vol. 2003-Janua, pp. 327–334, 2003.
[51] J. A. Calvo-Manzano, G. Cuevas, J. Mejia, M. A. Muñoz, and T. San Feliu, "How is CMMI-DEV applying when using TSP project planning," CERMA 2009 - Electron. Robot. Automot. Mech. Conf., pp. 143–148, 2009.
[52] Suasti, A. 2014. TSP Team Software Process. https://prezi.com/u6kw6yeixjzc/tsp-team-software-process/
[53] CrossTalk. 2005. The Journal of Defense Software Engineering. 18:3. pp. 32.
[54] L. M. G. Amaral and J. P. Faria, "A Gap Analysis Methodology for the Team Software Process," Qual. Inf. Commun. Technol. (QUATIC), 2010 Seventh Int. Conf., pp. 424–429, 2010.
[55] Gutimar Soluciones. 2012. Team Software Process (TSP) y Team Process Software (TPS). https://gutimarsoluciones.wordpress.com/2012/05/09/team-software-process-tsp-y-team-process-software-tsp/
[56] Universidad Tecnológica de Hermosillo. Sin fecha. Team Software Process. Sistemas de Calidad en TI.
[57] Contanzo, MA. 2014. Comparación de modelos de calidad, factores y métricas en el ámbito de la Ingeniería de Software. Universidad Nacional de la Patagonia Austral. pp. 36.
[58] B. O. Sussy, C. M. J. Antonio, C. Gonzalo, and S. F. Tomás, "Using the TSP defined process and improving the project management process," Proc. - 6th ACIS Int. Conf. Softw. Eng. Res. Manag. Appl. SERA 2008, pp. 197–204, 2008.
[59] N. Tadayon, "Software engineering based on the team software process with a real world project," J. Comput. Sci. Coll., vol. 19, no. 4, pp. 133–142, 2004.
[60] Y. Park, H. Park, H. Choi, and J. Baik, "A study on the application of six sigma tools to PSP/TSP for process improvement," Proc. - 5th IEEE/ACIS Int. Conf. Comput. Info. Sci., ICIS 2006. conjunction with 1st IEEE/ACIS, Int. Work. Component-Based Softw. Eng., Softw. Arch. Reuse, COMSAR 2006, vol. 2006, pp. 174–179, 2006.
[61] J. I. Perna and C. George, "Author Index," Fifth IEEE Int. Conf. Softw. Eng. Form. Methods (SEFM 2007), pp. 401–401, 2007.

[62] D. Duran and A. Gamboa, “Introduciendo Psp (Procesos Personal De Software) En El Aula,” vol. 2, pp. 1–5, 2010.
[63] MinTIC, “Con apoyo del MINTIC 100 empresas TI se han certificado en modelos de calidad,” 2015. [Online]. Available: http://www.mintic.gov.co/portal/604/w3-article-14174.html.
[64] J. A. Franco, “Revisión de Propuestas de Certificación para PYMES Colombianas que producen Software,” Universidad Católica de Pereira, 2011.
[65] E. al. Peláez, L.E., “En Colombia : Una Mirada Desde Las Pymes Productoras,” Rev. académica e Inst. la UCPR, no. 92, pp. 89–98, 2012.
[66] Intersoftware. 2016. Clúster de Tecnología, Información y Comunicación de Antioquia. http://www.intersoftware.org.co/content/cluster-de-tecnologia-informacion-y-comunicacion-de-antioquia-0.
[67] Amaya, G. 2012. Asociación Alianza Sinertic. http://redclustercolombia.com/clusters-en-colombia/iniciativa/18
[68] Román, S. 2011. Clúster Tecnología, Información y Comunicación. http://redclustercolombia.com/clusters-en-colombia/iniciativa/56
[69] Intersoftware. 2016. Lanzamiento del Clúster Caribe TIC. http://www.intersoftware.org.co/content/lanzamiento-del-cluster-caribetic
[70] Cámara de Comercio de Bogotá. 2016. Clúster Bogotá Software y TI. http://www.ccb.org.co/Clusters/Cluster-de-Software-y-TI
[71] La Nota. 2013. Se crea el Clúster de Tecnología Informática en Salud, CLINSA. http://www.lanotadigital.com/business/se-crea-el-cluster-de-tecnologia-informatica-en-salud-clinsa-37457.html.
[72] Molano, A. 2013. Salud y Tecnología de la mano en el clúster sectorial. Colombia Digital. https://colombiadigital.net/actualidad/experiencias/item/5542-salud-y-tecnologia-de-la-mano-en-el-cluster-sectorial.html.
[73] Bohorquez, C., et al. 2013. The Bogotá Software Cluster. Universidad de Harvard. pp. 30. http://www.isc.hbs.edu/resources/courses/moc-course-at-harvard/Documents/pdf/student-projects/Colombia_Software_2013.pdf.
[74] Tecnósfera, “Cien empresas TI recibieron certificado de calidad con apoyo de MinTic,” 2015. [Online]. Available: http://www.eltiempo.com/tecnosfera/novedades-tecnologia/empresas-ti-con-certificado-de-calidad/16442026.
[75] SoftValores, “Reconocimiento del MinTic en la adopción de modelos de calidad.” .
[76] MinTIC, “Viceministra TI participó en ciclo de conferencias ‘Modelos de calidad para la industria del software en Colombia,’” 2016. [Online]. Available: http://www.mintic.gov.co/portal/604/w3-article-14679.html.
[77] Cámara de Comercio de Bogotá, “Convocatoria para la certificación de calidad de empresas TI,” 2016. [Online]. Available: http://www.ccb.org.co/Clusters/Cluster-de-Software-y-TI/Noticias/2016/Enero/Convocatoria-para-la-certificacion-de-calidad-de-empresas-TI.
[78] MinTIC, “Ministerio TIC otorga créditos condonables para certificación en calidad de software,” 2013. [Online]. Available: http://www.mintic.gov.co/portal/604/w3-article-4415.html.
[79] MinTIC, “MinTIC abre la 3° convocatoria para certificación Internacional en desarrollo de software de excelencia - PSP Y TSP,” 2014. [Online]. Available: http://www.mintic.gov.co/portal/604/w3-article-5153.html.
[80] MinTIC, “MinTIC ofrece becas para formación y certificación en software de excelencia,” 2014. [Online]. Available: http://www.mintic.gov.co/portal/604/w3-article-7806.html.
[81] Guía Solucione TIC, “Certificación en Prácticas de Calidad TSP/PSP,” 2013. [Online]. Available: http://www.guiadesolucionestic.com/educacion/item/1474-certificacion-en-practicas-de-calidad-tsppsp.
[82] ICETEX, “FONDO TSP/PSP - CUARTA CONVOCATORIA,” 2015. [Online]. Available: https://www.icetex.gov.co/dnnpro5/en-us/fondos/fondosparaeldesarrollodeti/fondotsppsp.aspx.
[83] D. O. García, “Análisis Descriptivo del Impacto de la Aplicación de la Metodología CMMI-DEV en Proyectos de Desarrollo de Software en Empresas Colombianas,” 2015.
[84] P. Restrepo, “El potencial del software colombiano en el mundo,” La República, 2015. [Online]. Available: http://www.larepublica.co/el-potencial-del-software-colombiano-en-el-mundo_228816.
[85] IT Institute Colombia, “Curso de Team Software Process en Bogotá - Colombia,” 2016. [Online]. Available: http://www.aprendemas.com/co/curso/team-software-process-902040/.
[86] SEONTI, “SEONTI,” 2010. .
[87] Intersoftware, “Foro Iberoeka 2013.” 2013.
[88] Cintel, “Colombianos primeros en América Latina en certificaciones de calidad de software,” 2016. [Online]. Available: http://cintel.org.co/software-de-calidadcolombia/.
[89] PersonalSoft, “Líder Técnico,” 2014. [Online]. Available: http://www.elempleo.com/colombia/trabajo/Líder-Técnico-_1881677974.
[90] A. Software, “Estamos buscando ingenieros para nuestra fábrica de software,” 2011. [Online]. Available: http://arandasoft.com/estamos-buscando-ingenieros-para-nuestra-fabr/.
[91] Linexperts Consultoría Empresarial, “Ingeniero de Desarrollo (RUP, SCRUM, TSP) CMMI Dev 3,” 2015. [Online]. Available: http://www.elempleo.com/colombia/trabajo/Ingeniero-de-Desarrollo-(RUP,-SCRUM,-TSP)-CMMI-Dev-3_1881884643.

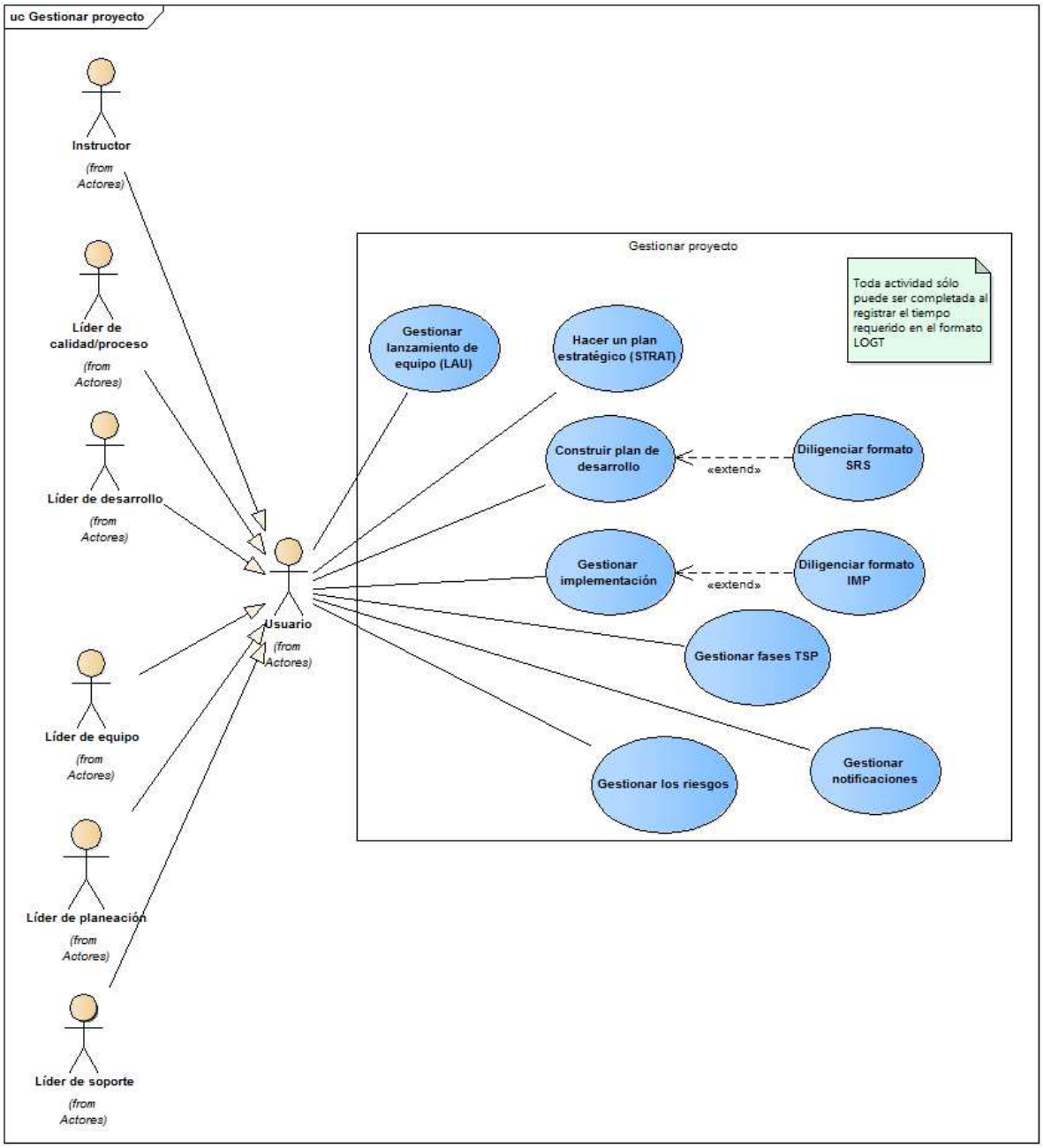
[92] T. Calvo, J.A.;Cuevas G.;Féliz A.; San Feliu, “Quality in the Development of Applications through the Team Software Process TSP Methodology,” Santiago Compost., pp. 3–8, 2010.
[93] MIT AITI. (2012). Accelerating Information Technology Innovation. http://gsl.mit.edu/media/programs/colombia-summer-2012/materials/lec01.pdf
[94] Mois, M. (2015). JSF 2.0 Programming Cookbook. Elixis Media. USA. 129 p.
[95] Pech-May, F., et. al. Sin año. Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces. Instituto Tecnológico Superior de los Ríos. 9 p.
[96] Tutorials Point. (2017). JSF. Tutorials Point. 346 p.
[97] SGI Consulting. Sin año. Tutorial de JavaServer Faces. http://www.sgi-consulting.com/sgi2/jsf/JSF.pdf
[98] TEQI. (2007). Encripta tus Mensajes con Apache Codec. http://www.teqi.org/Codifica-Mensajes-Con-Apache-MD5-Codec-DigestUtils.html
[99] Codeline. (2008). Commons File Upload. https://codeline.wordpress.com/2008/07/15/commons-file-upload/
[100] Pérez, M. Sin año. Trabajando con JSON en Android. https://geekytheory.com/trabajando-con-json-en-android-gson
[101] Suárez, H. (2003). Manual Hibernate. JavaHispano. 19 p.
[102] MongoDB. Sin año. MongoBB Java Driver: The next generation Java driver for MongoBB. https://mongodb.github.io/mongo-java-driver/
[103] Jongo. (2017). Jongo: Query in Java as in Mongo Shell. http://jongo.org/
[104] Atenea Tech. (2007). Desenredando XmlBeans. http://ateneatech.com/blog/desenredando-xmlbeans
[105] Alimam, H. (2014). JSF and Twitter Bootstrap Integration. http://blog.hatemalimam.com/jsf-and-twitter-bootstrap-integration/
[106] Peña, LA. (2016). Materialize, un Framework Front-End a lo Material Design. https://www.leninalbertop.com.ve/blog/materialize-un-framework-front-end-lo-material-design/
[107] Barbativa, A., et. al. (2015). Desarrollo ágil de una aplicación para dispositivos móviles. Caso de estudio: Taxímetro móvil. Universidad Distrital Francisco José de Caldas. Revista Ingeniería. 20.20: 14 p.
[108] Solarte, G., Ocampo, CA. (2010). Usabilidad en Espacio Web. Universidad Tecnológica de Pereira. Scientia et Technica. pp. 251-256
[109] Aggarwal KK., Singh, Y. Software Engineering. India. New Age International Limited. 408 p.
[110] Gamma, E., et. al. (2003). Patrones de Diseño: Elementos de software orientado a objetos reutilizable. Pearson Educación. España. pp. 191
[111] Curso gestión de calidad de software, Especialización en gerencia de sistemsa informátics. Universidad Pontificia Bolivariana. Sede de Bucaramanga 2016

ANEXOS

ANEXO 1: Diagramas de casos de uso

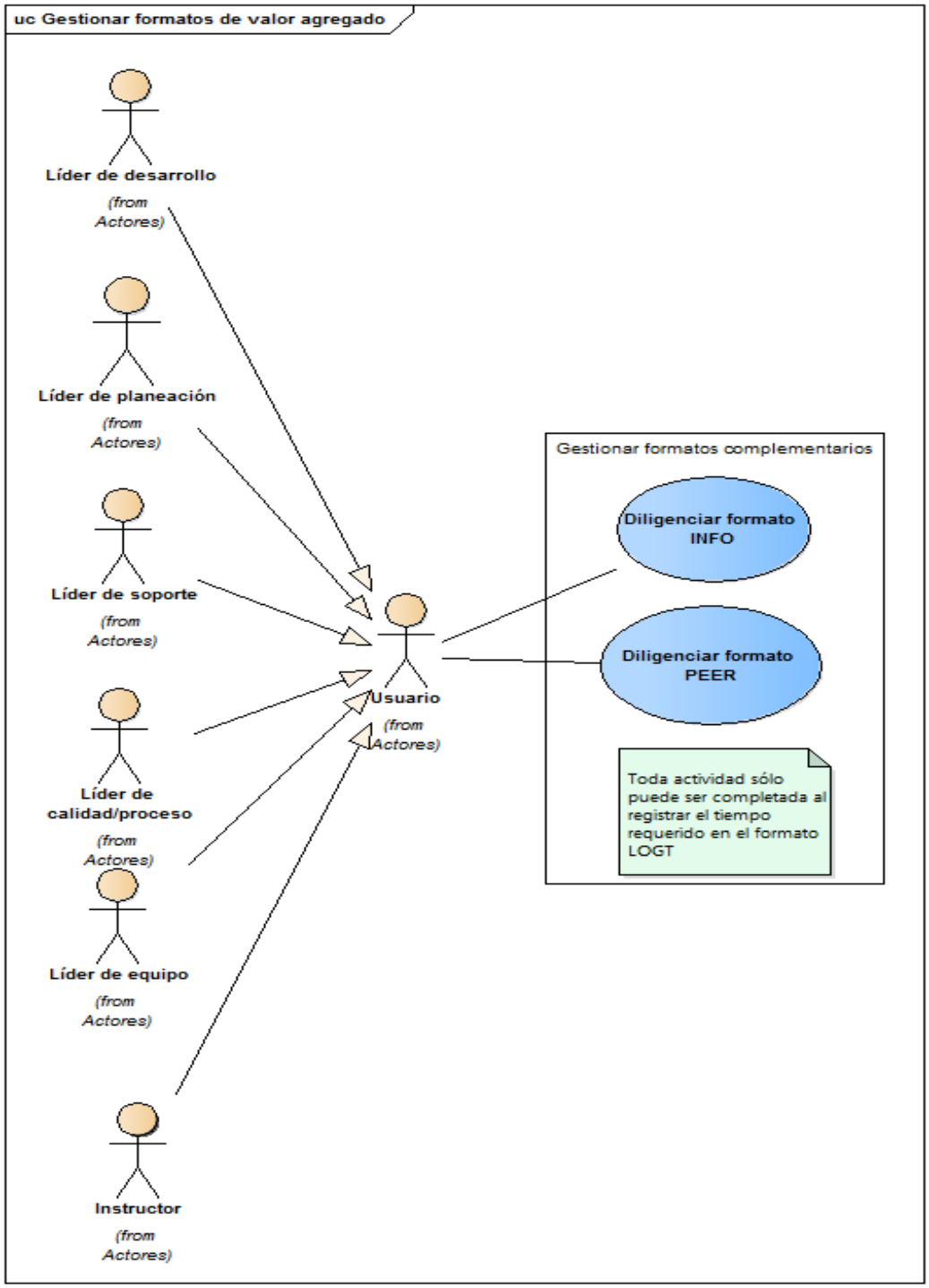
1. Gestión del proyecto

La gestión del proyecto alude a la asignación de roles de la fase de Lanzamiento y el paso general entre fases para cada ciclo. Incluye además los planes, estratégico y de desarrollo, notificaciones para los criterios de salida de cada fase y los riesgos involucrados en cada ciclo.



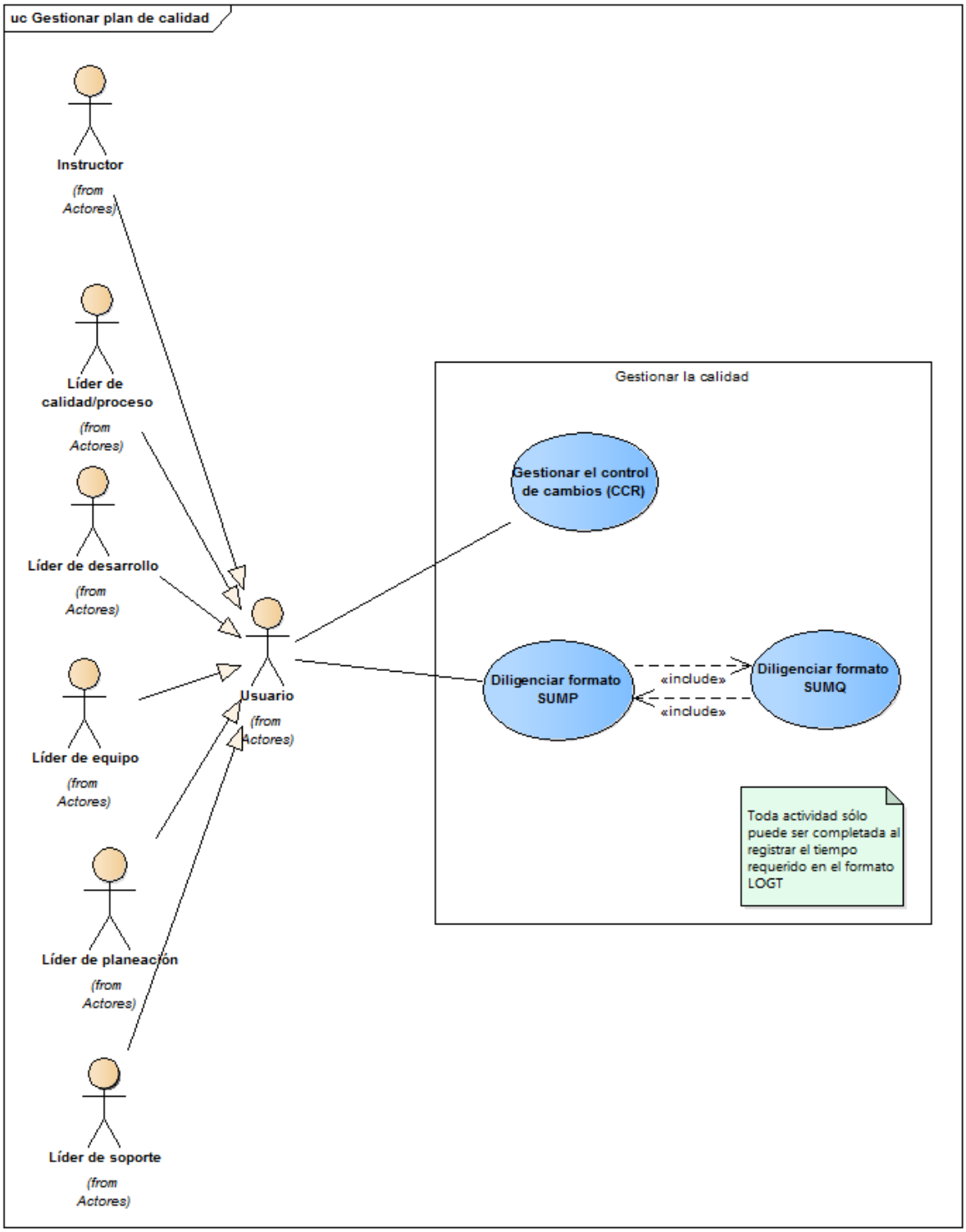
2. Gestión de formatos complementarios

Consiste en el diligenciamiento de información de los formatos que no tienen un impacto significativo en la metodología pero que de una u otra forma aportan un valor.



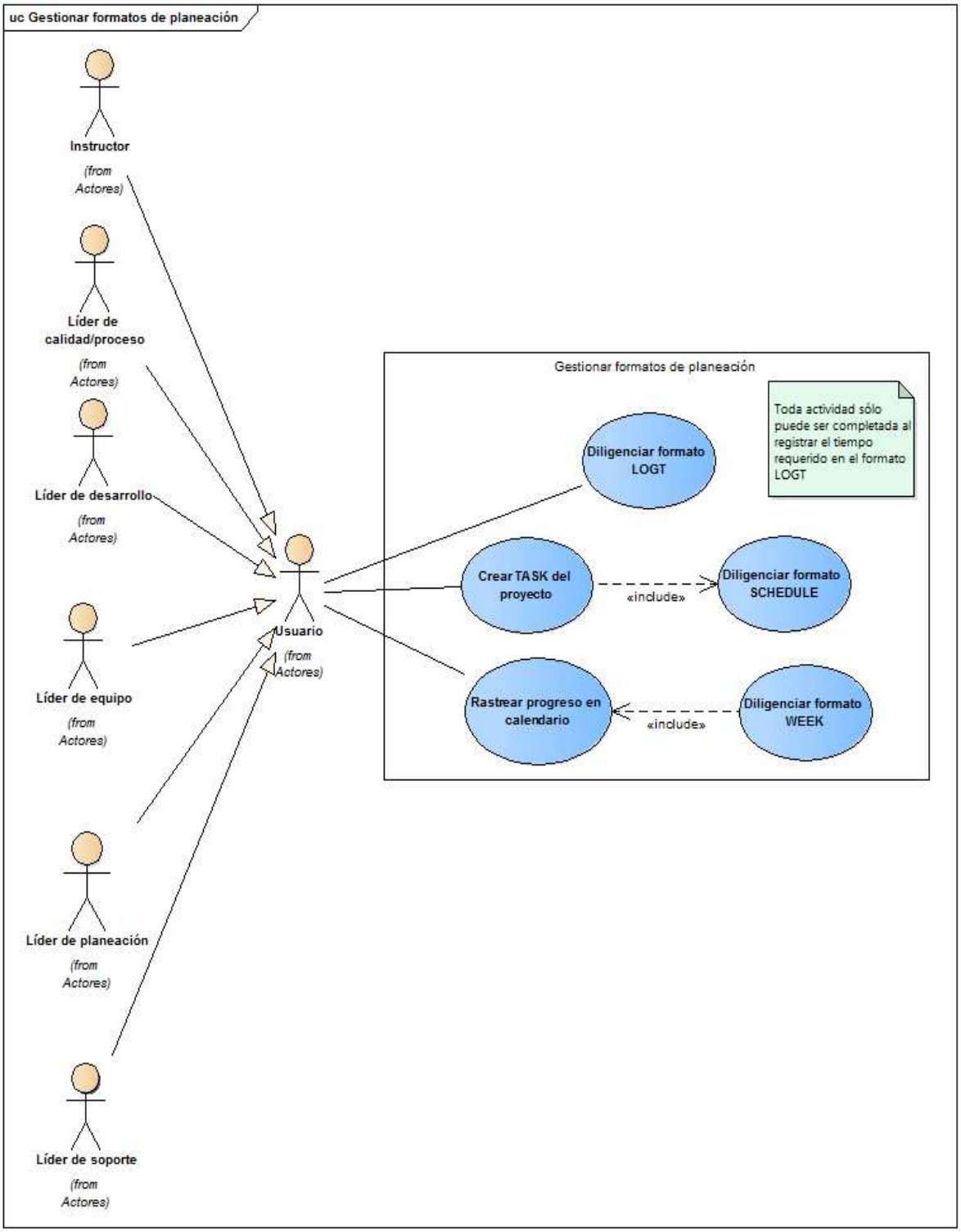
3. Gestión de la calidad

Con la gestión de métricas obtenidas, los formatos SUMP y SUMQ se generan en su totalidad para la obtención de indicadores y métricas que especifiquen la calidad del producto generado con base en los avances de éste en el tiempo transcurrido.



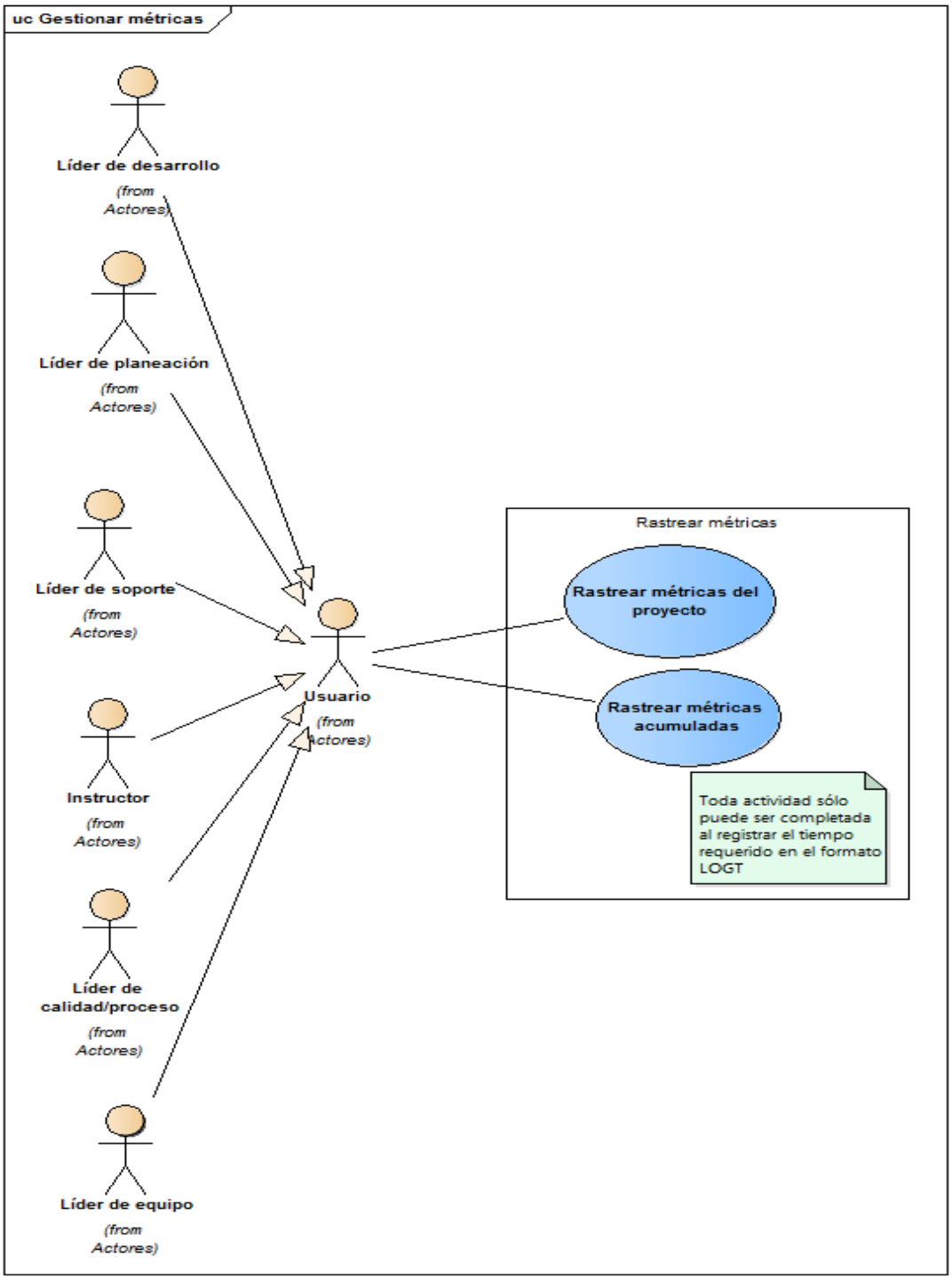
4. Gestión de formatos de planeación

Se centra en los formatos TASK, SCHEDULE, WEEK y LOGT para registrar los avances del proyecto en forma semanal e ir actualizando y comparando las métricas asociadas para los tiempos estimados y el cumplimiento de tareas planeadas.



5. Rastreo de métricas

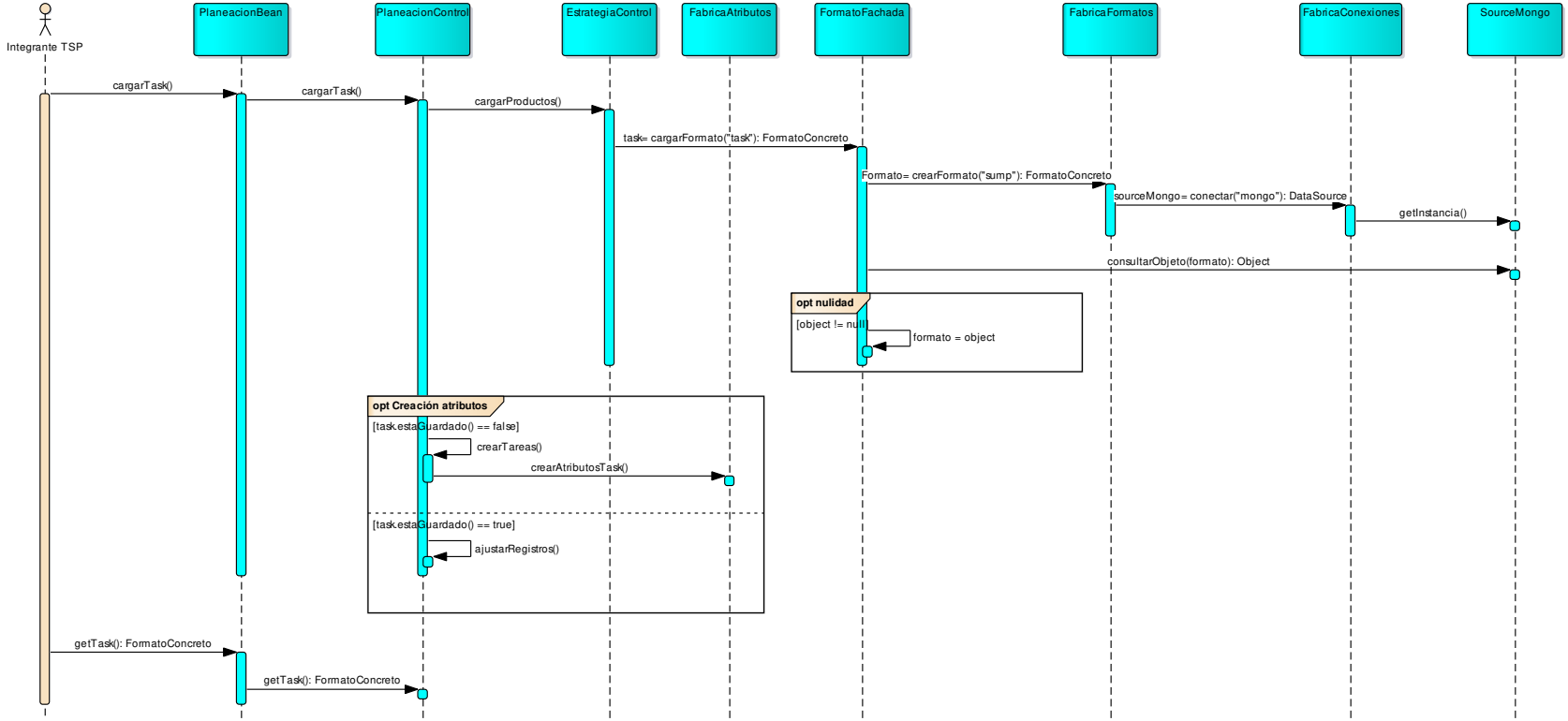
Consiste en identificar las métricas recolectadas y presentarlas al usuario en forma cuantitativa y cualitativa a través del proyecto desarrollado o la comparativa entre varios proyectos en los que un usuario se ve involucrado.



ANEXO 2: Diagramas de secuencia

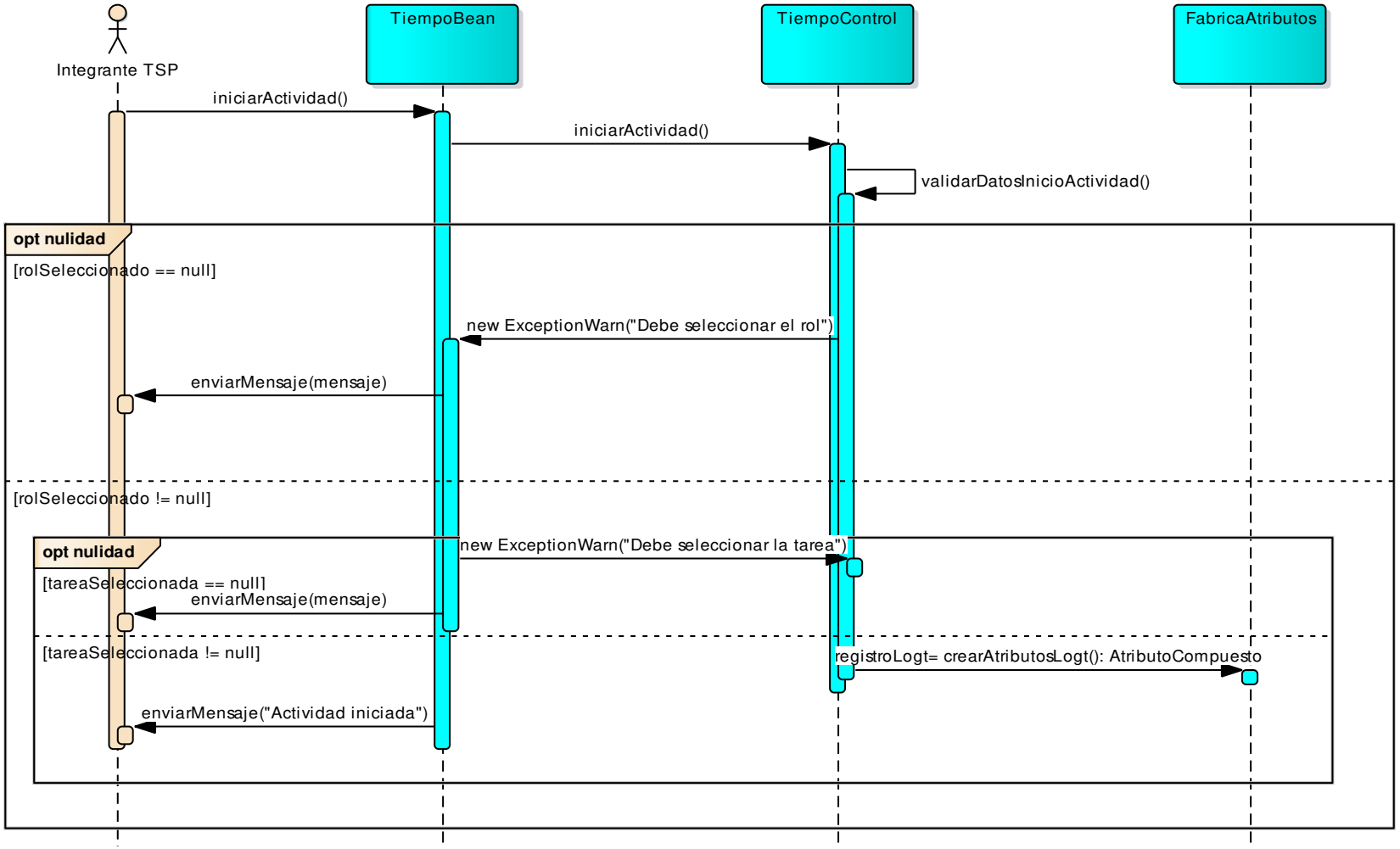
1. Cargar formato TASK.

Cuando el usuario solicita el formato TASK, el controlador *PlaneacionControl* resuelve la solicitud cargando los productos definidos en la fase de Estrategia de modo que para cada etapa se construya un atributo equivalente a cada registro del formato TASK apoyándose en las fábricas *FabricaAtributos* y *FabricaFormatos*. En caso de que el formato TASK ya se encuentre registrado en el sistema, se consulta mediante la conexión brindada por *FabricaConexiones*.



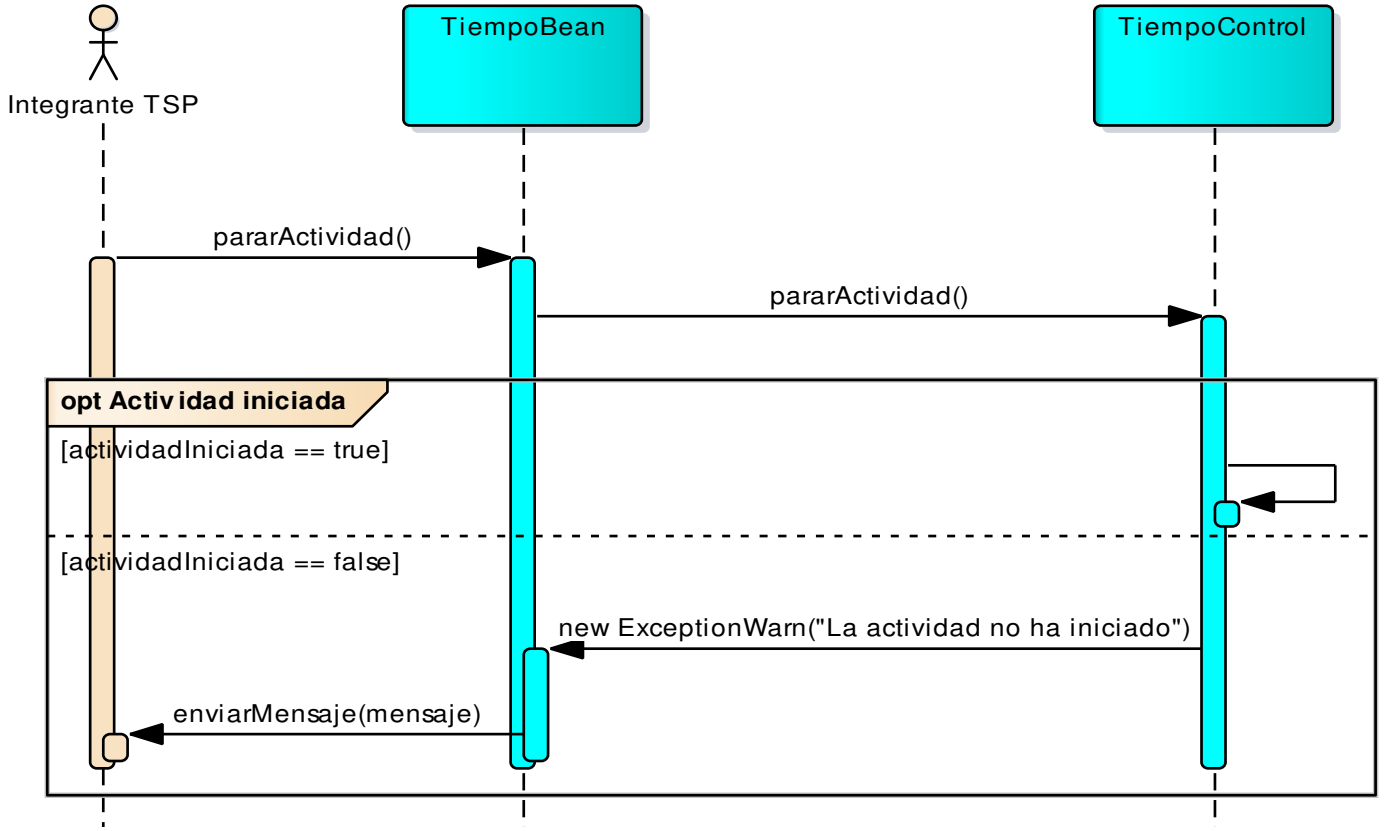
2. Iniciar actividad.

Cuando el usuario desea iniciar una actividad, *TiempoControl* resuelve la solicitud validando los datos de la tarea e instanciando los tiempos de inicio.



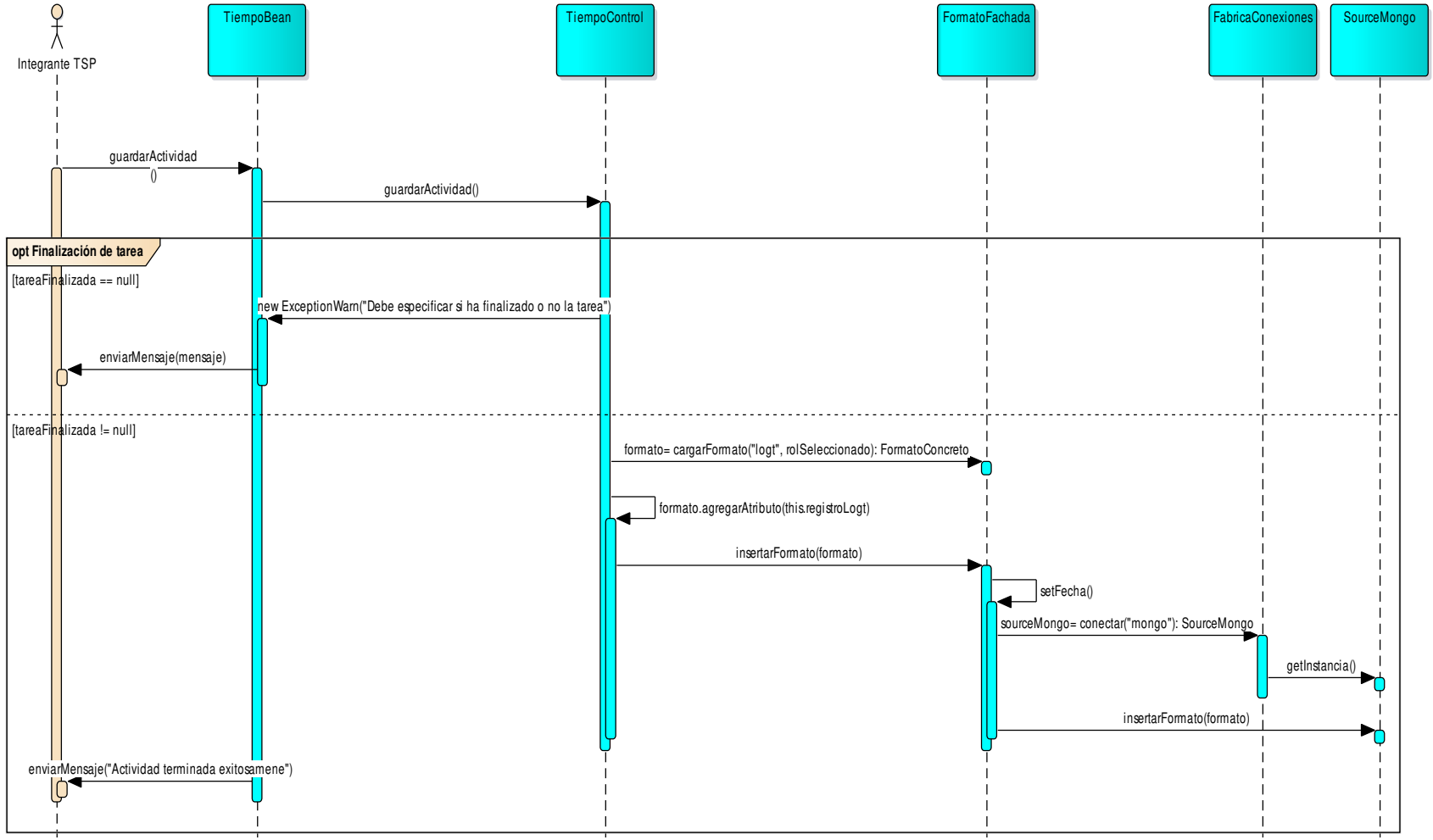
3. Parar actividad.

   El controlador *TiempoControl* marca el tiempo de finalización de la tarea.



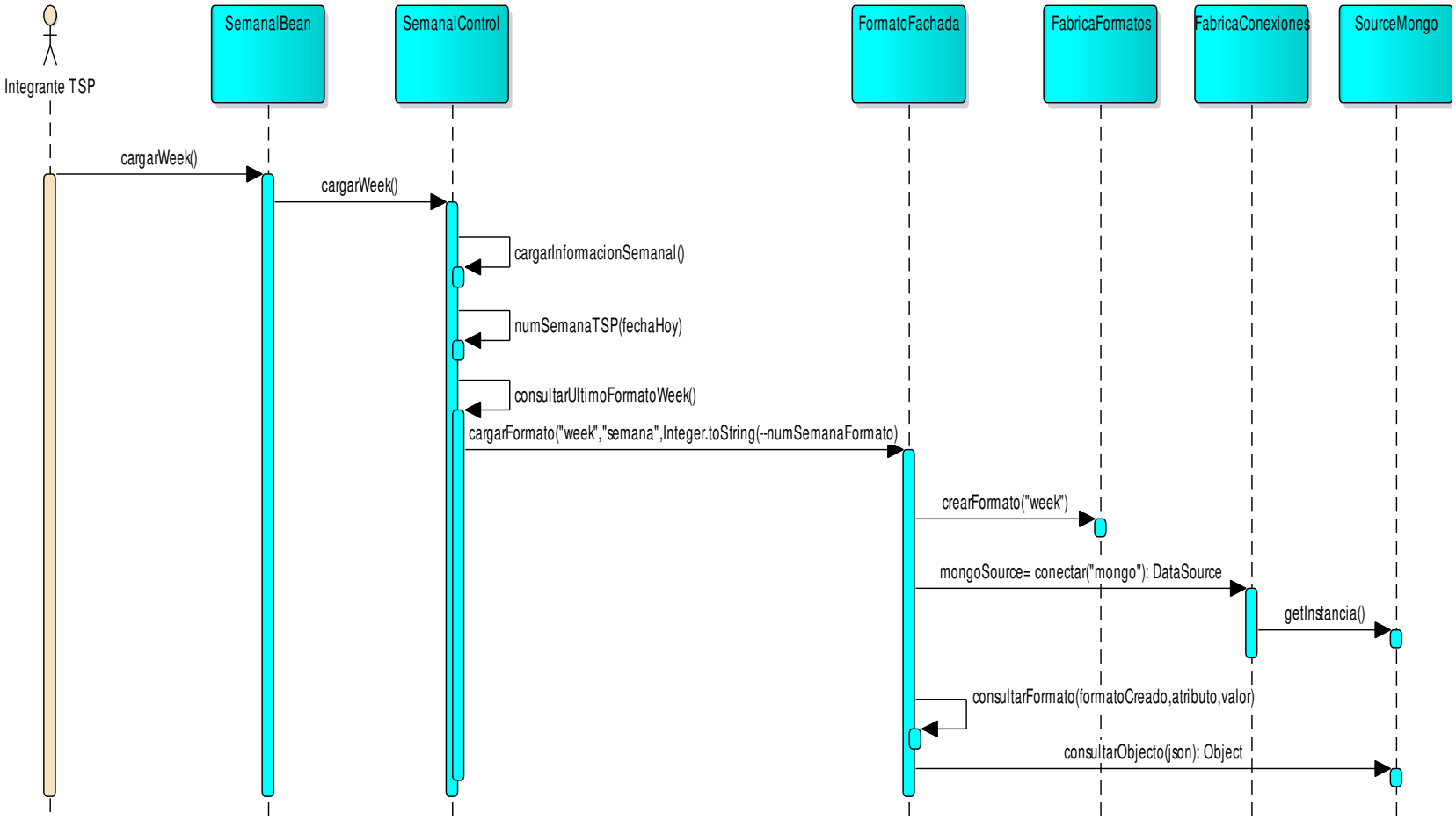
4. Guardar actividad.

TiempoControl resuelve la solicitud de guardar la actividad actualizando el formato que respecta al integrante del equipo que acaba de ejecutar la tarea especificando si ésta fue finalizada o no.



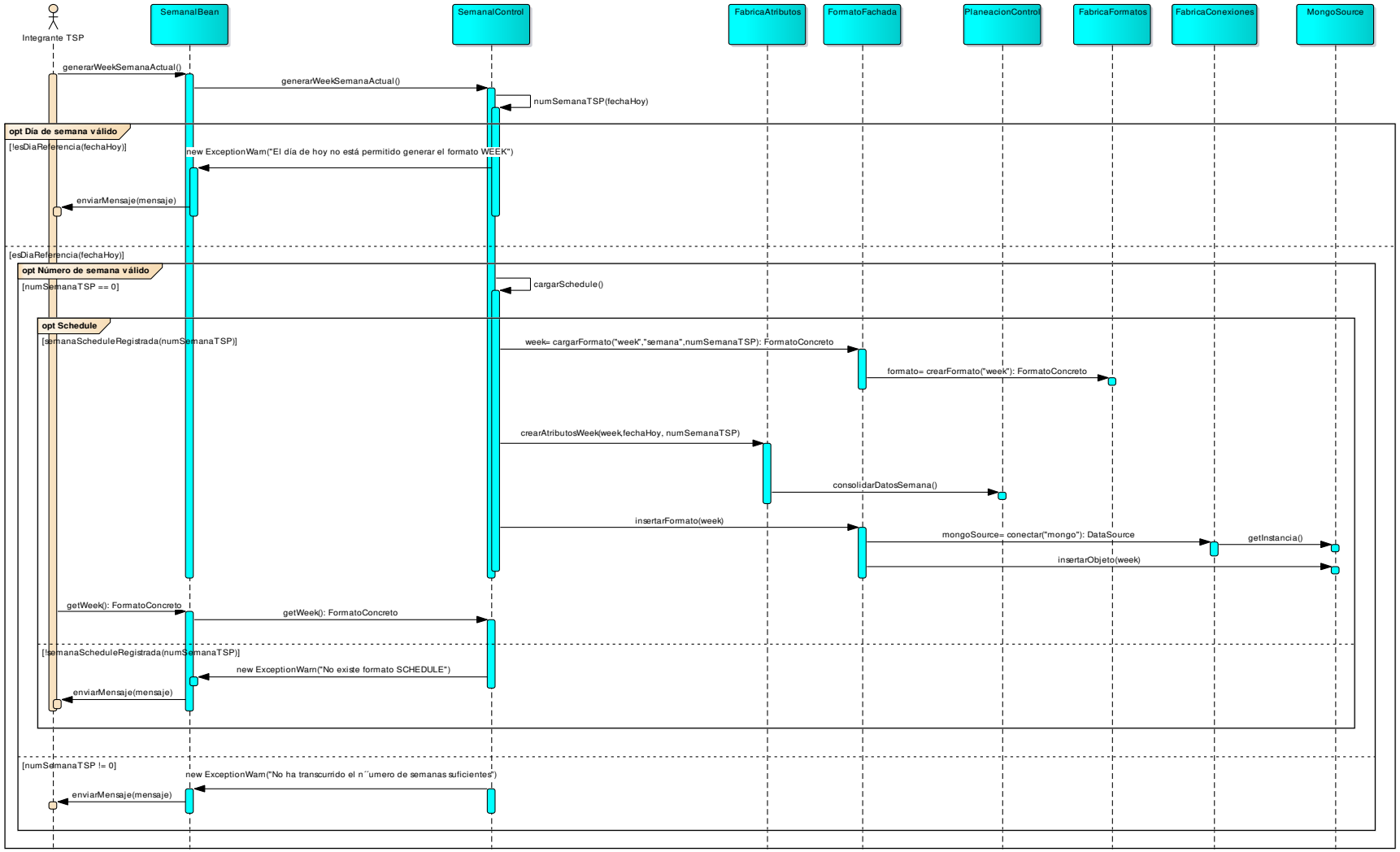
5. Consultar formato WEEK

El controlador *SemanaControl* trae los datos almacenados para el último formato WEEK generado, ya sea el de la semana actual o la anterior. Para evaluar el número de semana actual se ejecuta el algoritmo correspondiente, el cual se soporta en la en el día de revisión semanal.



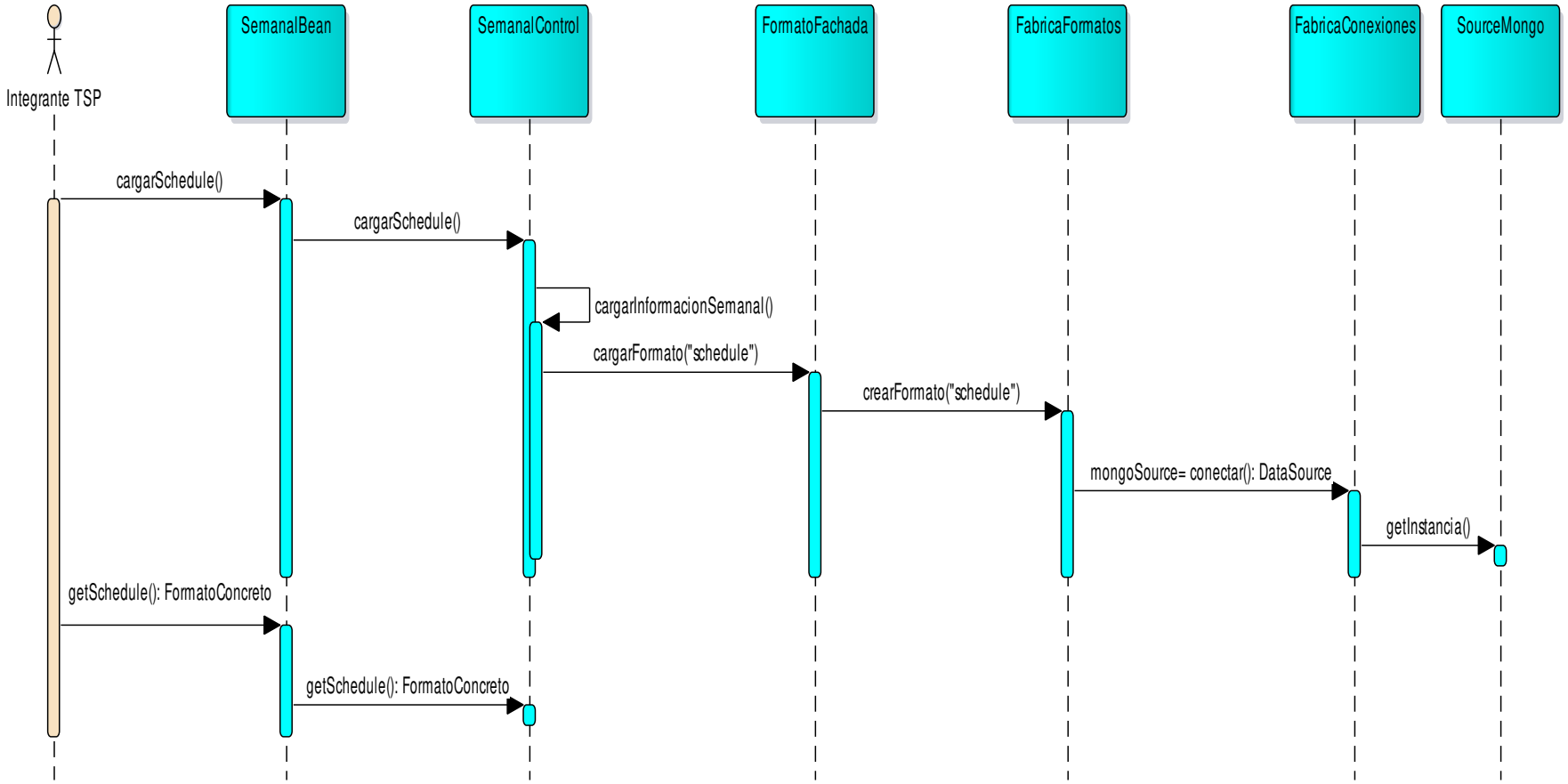
6. Generar formato WEEK

SemanalControl obtiene los avances obtenidos en la última semana y los incorpora al formato para finalmente registrarlo en forma persistente y presentarlo al usuario.



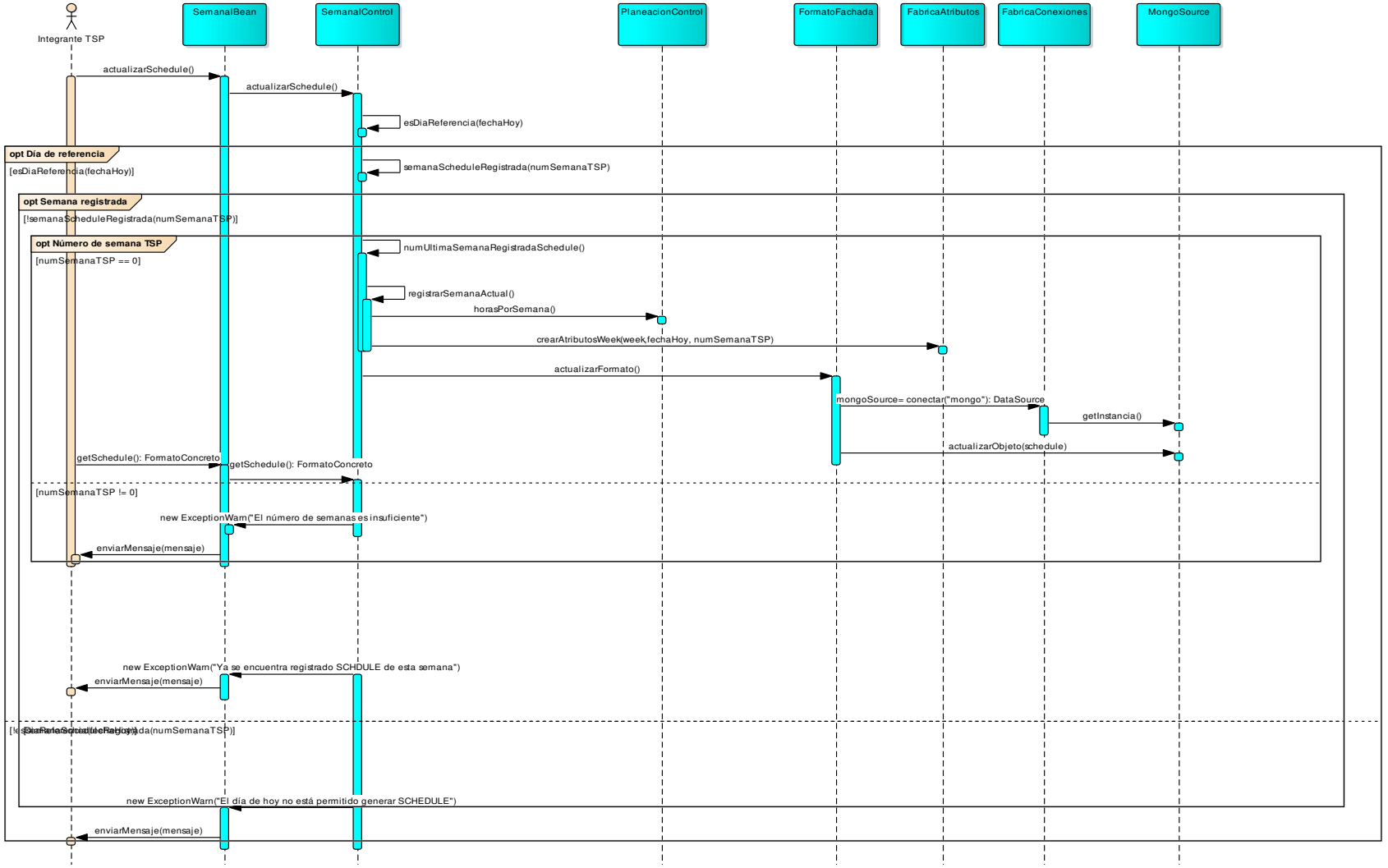
7. Consultar formato SCHEDULE

SemanalControl obtiene el formato del ciclo actual y se presenta al usuario.



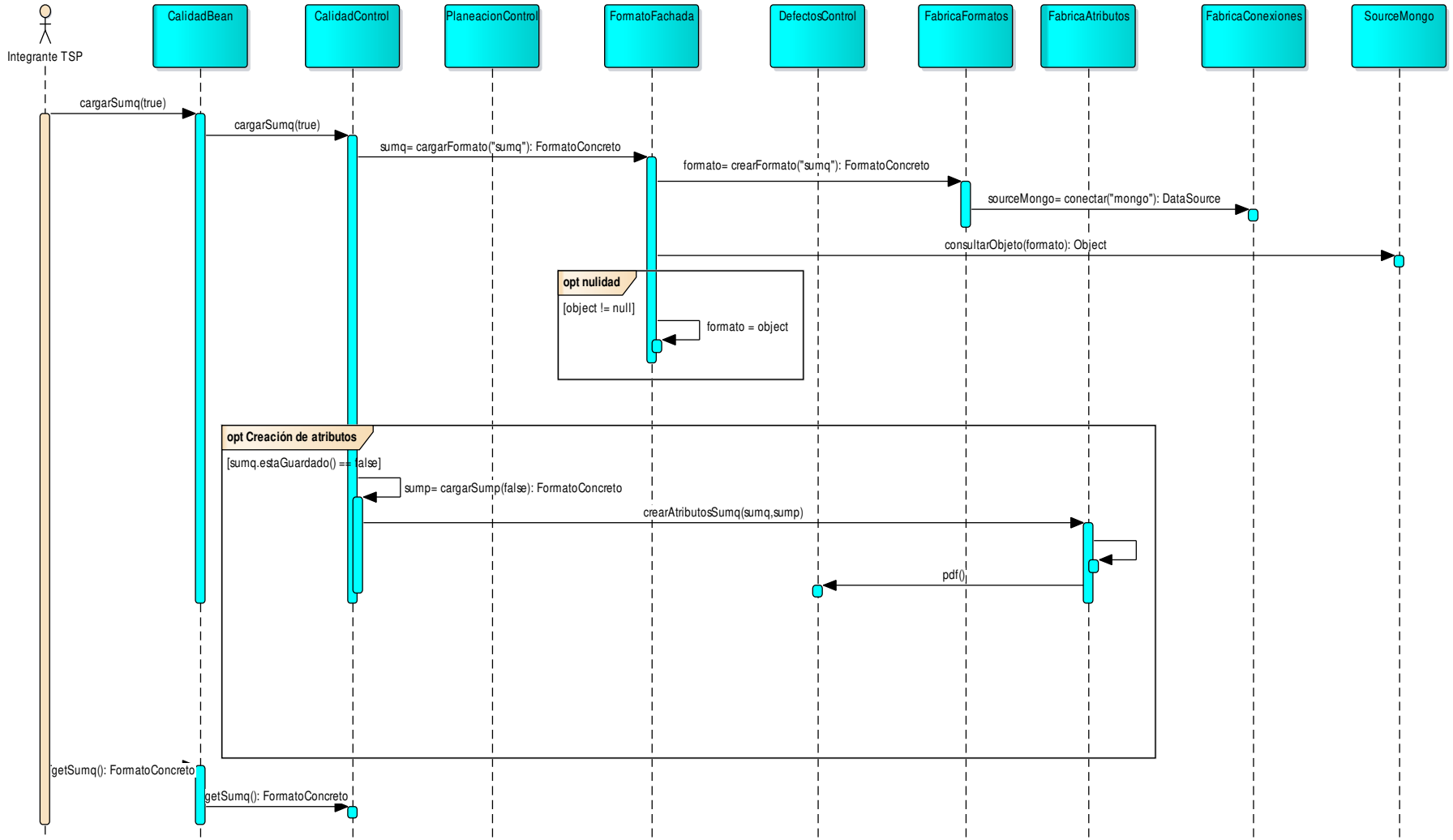
8. Actualizar formato SCHEDULE

SemanalControl evalúa los avances del proyecto en la semana y los añade al formato y actualizándolo en forma persistente.



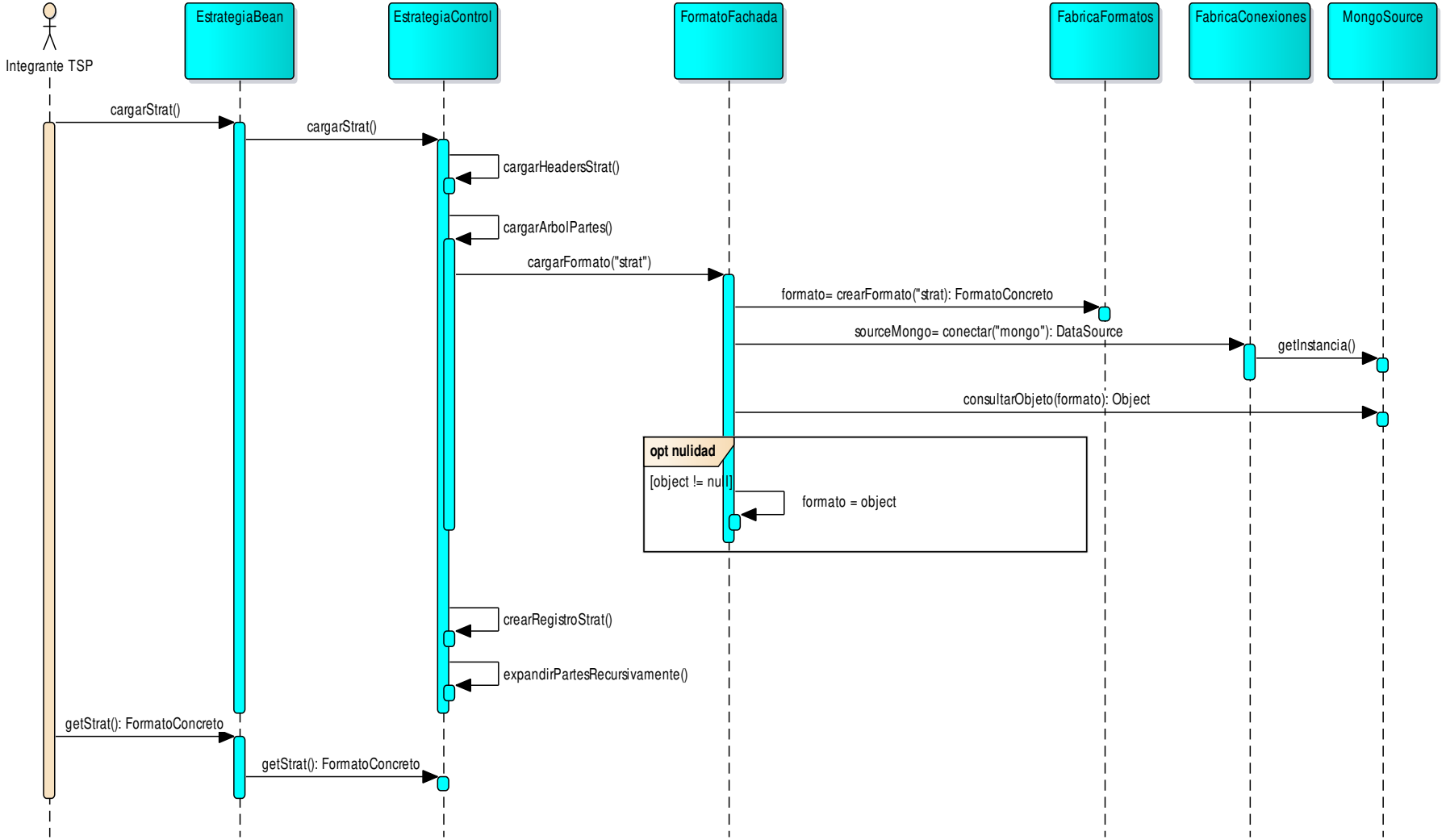
9. Cargar formato SUMQ

CalidadControl calcula las métricas e indicadores del formato SUMQ. En caso de encontrar el formato SUMQ registrado, lo muestra al usuario.



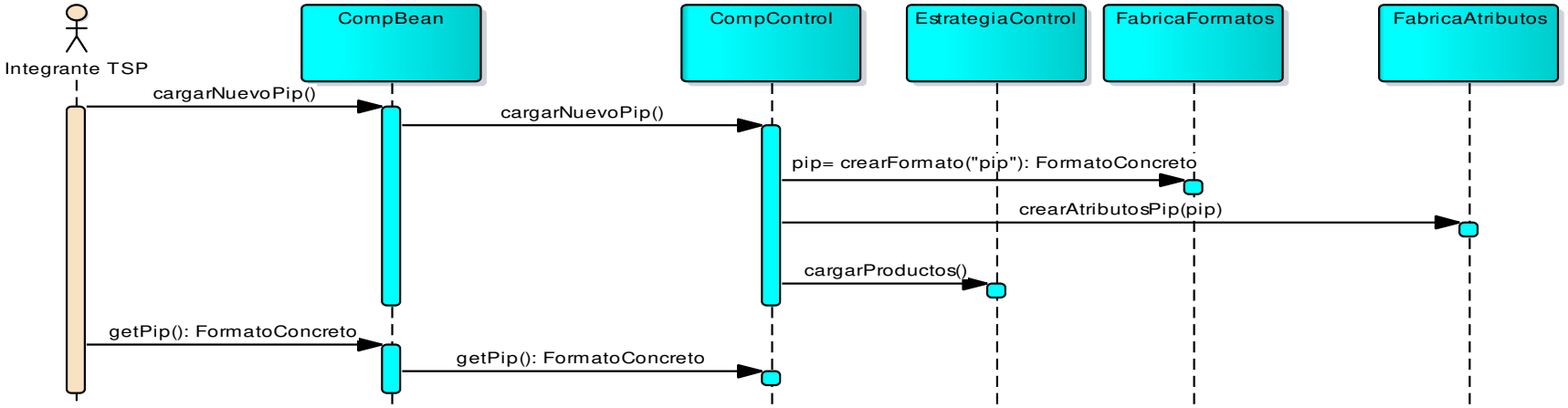
10. Cargar formato STRAT

El controlador *EstrategiaControl* carga el formato STRAT con base en el número de ciclos del proyecto y el árbol de registros correspondientes.



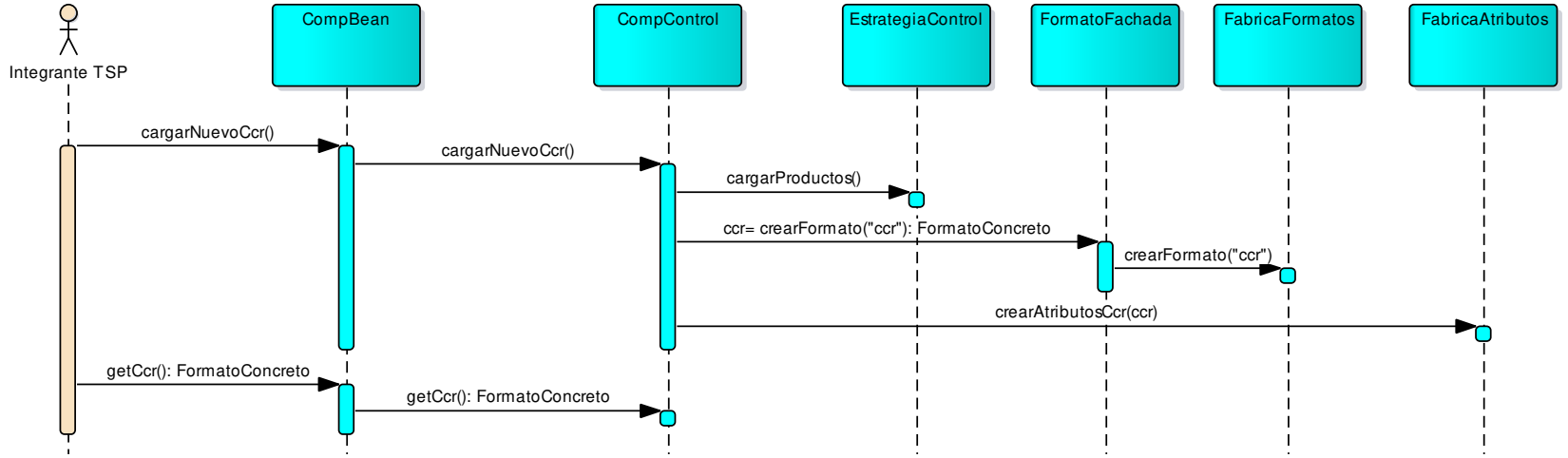
11. Cargar formato PIP

CompControl carga el formato PIP con base en los productos creados en la fase de Estrategia.



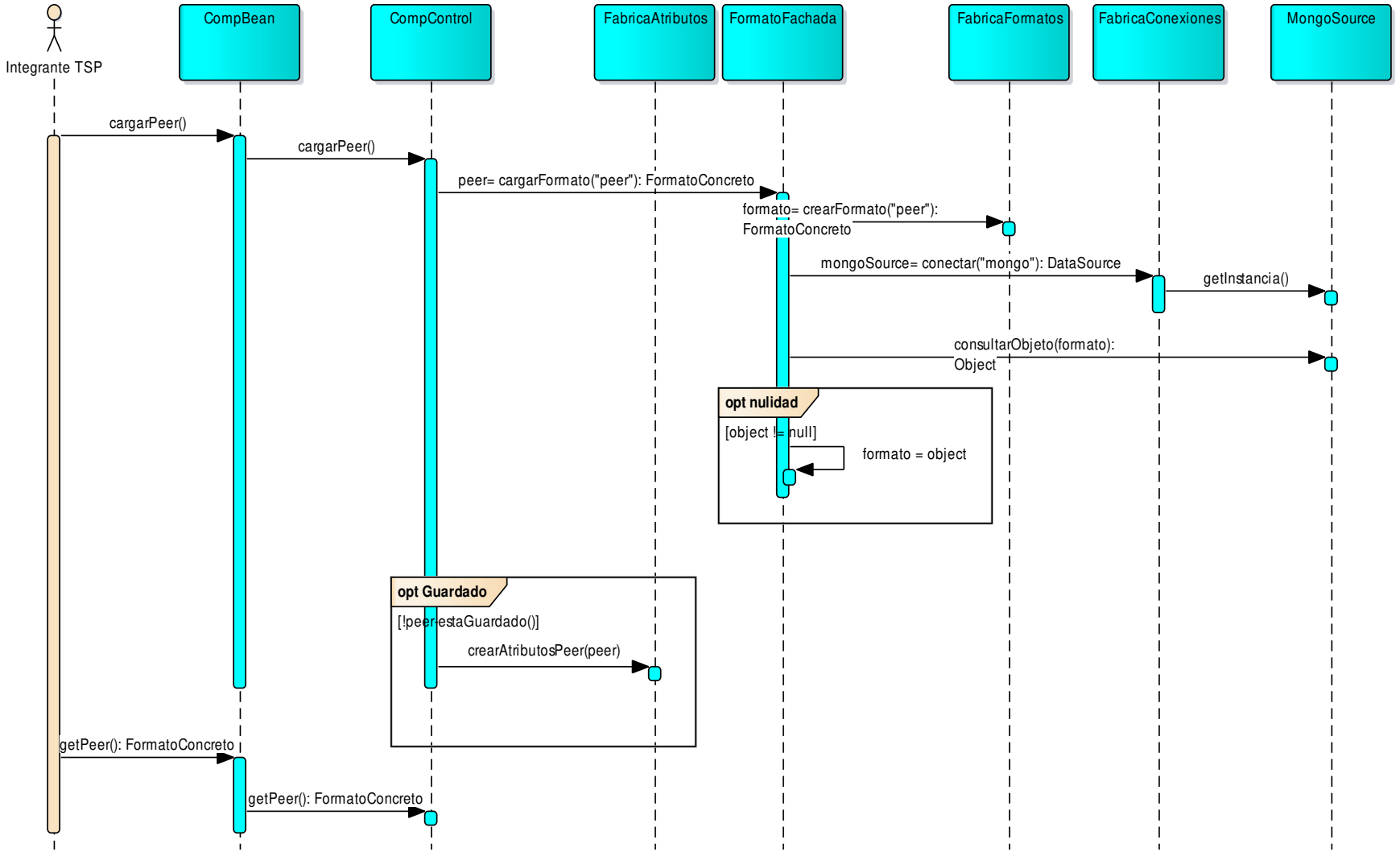
12. Cargar formato CCR

CompControl carga el formato CCR con base en los productos creados en la fase de Estrategia.



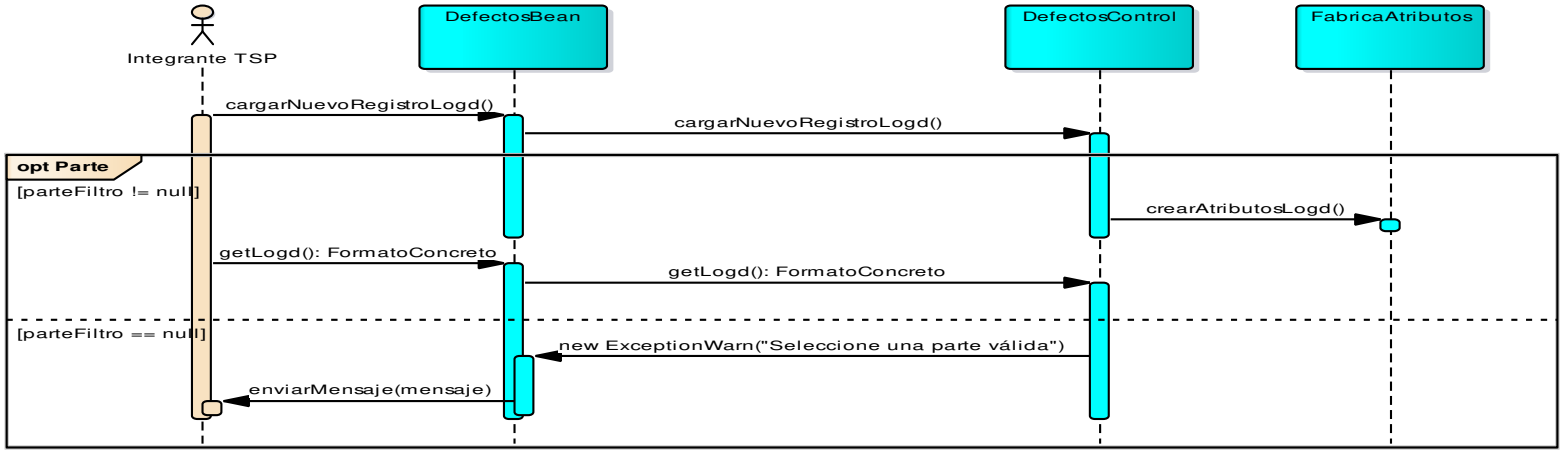
13. Cargar formato PEER

CompControl carga el formato PEER del integrante del equipo para el presente ciclo.



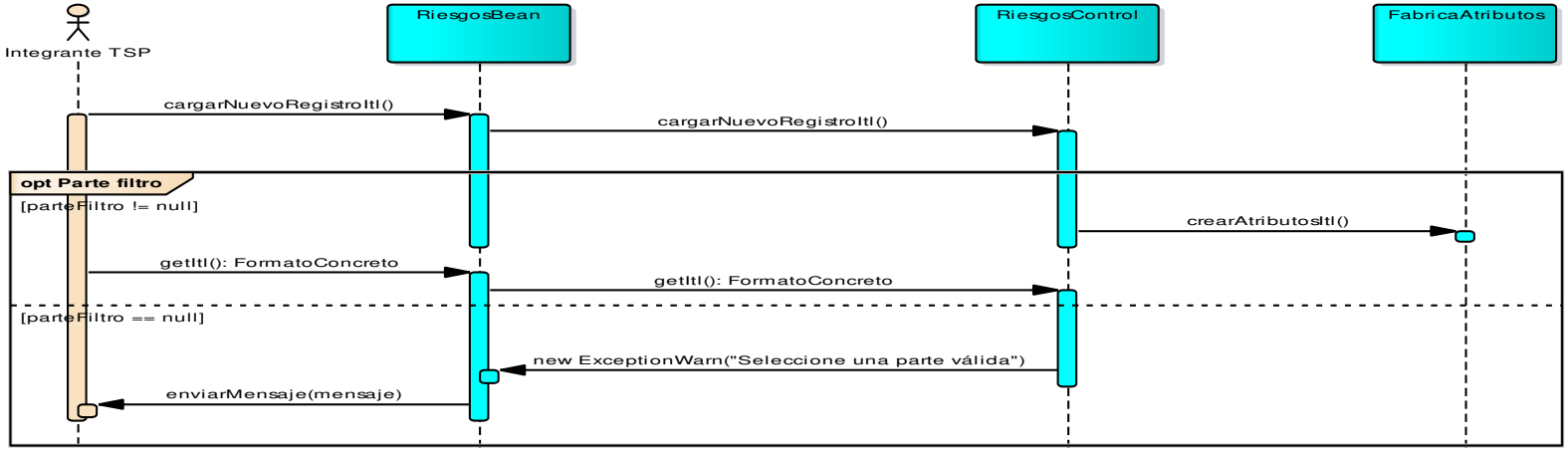
14. Cargar datos de nuevo defecto

El controlador de defectos *DefectosControl* carga los datos del nuevo registro para el formato LOGD a partir de la parte creada en la fase de Estrategia.



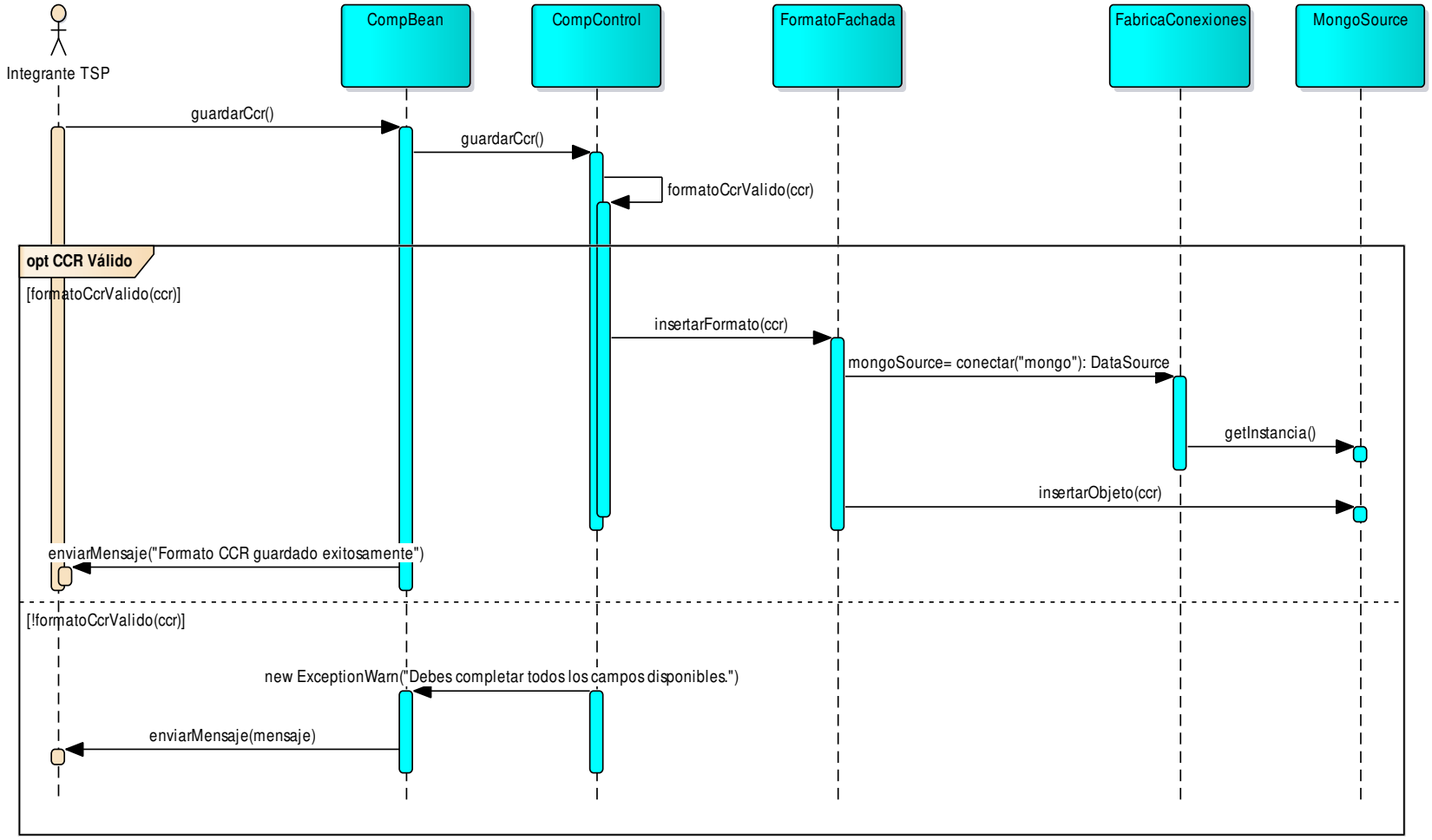
15. Cargar datos de nuevo riesgo/problema

El controlador *RiesgosControl* carga los datos del nuevo riesgo o problema del formato ITL a partir de la parte creada en la fase de Estrategia.



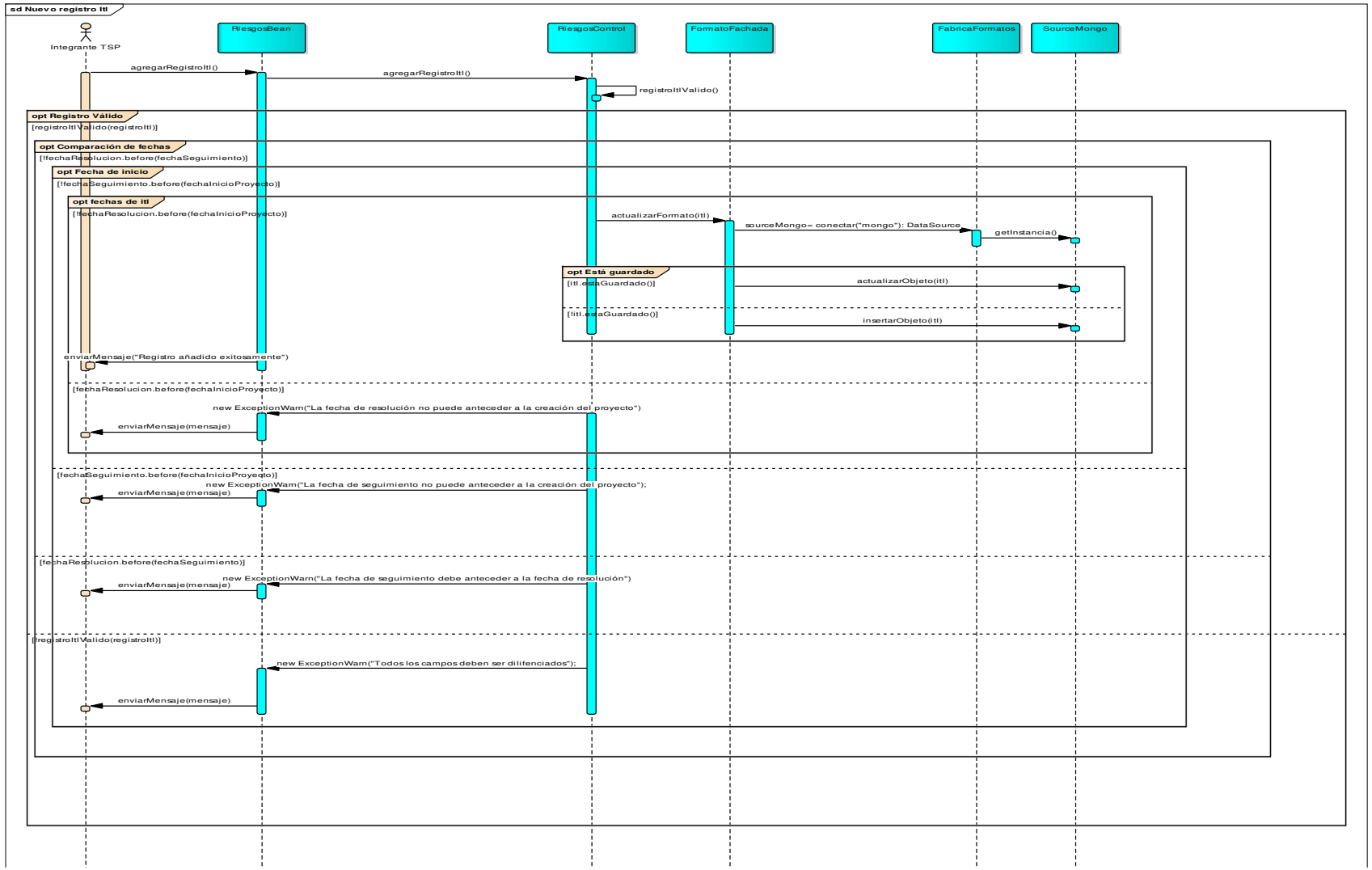
15. Guardar formato CCR.

El controlador *CompControl* gestiona el registro del formato CCR solicitando su inserción a *FormatoFachada* que se apoya en las clases que gestionan la conexión.



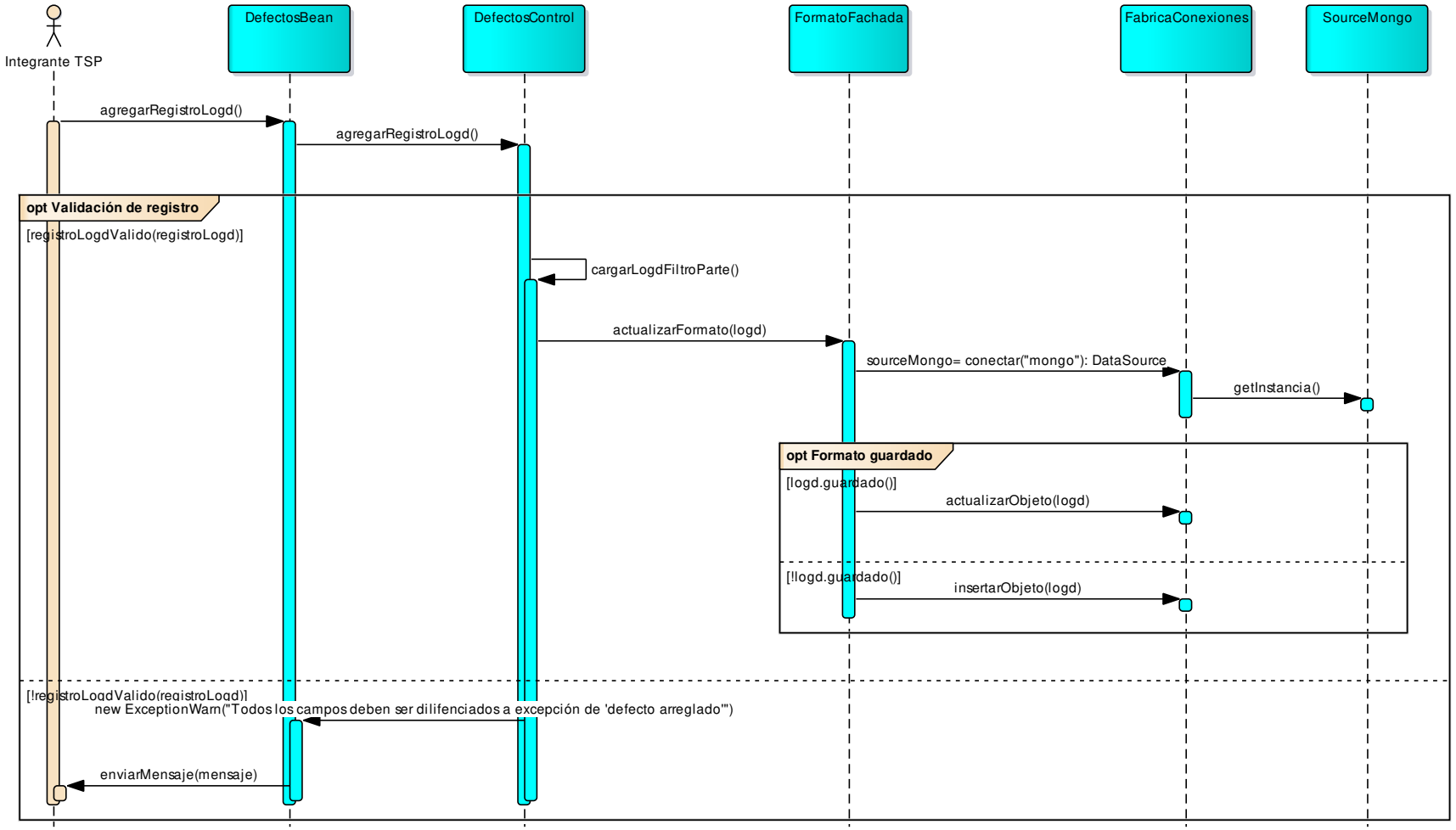
16. Registrar un riesgo/defecto

DefectosControl se encarga de gestionar la inserción del nuevo riesgo o defecto actualizando el formato ITL respectivo con esta información.



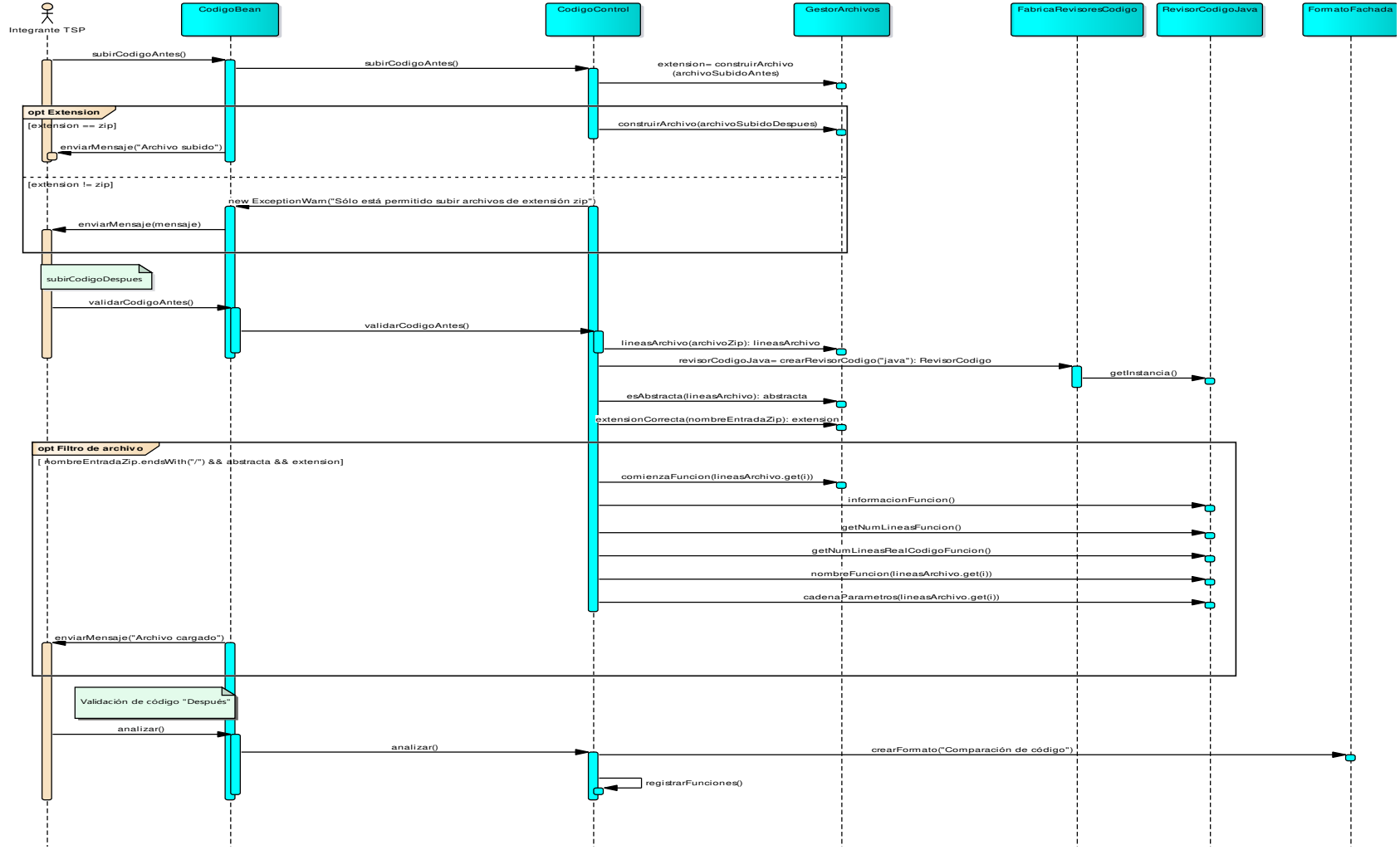
17. Registrar un defecto

DefectosControl se encarga de gestionar la inserción del nuevo defecto actualizando el formato LOGD respectivo con esta información.



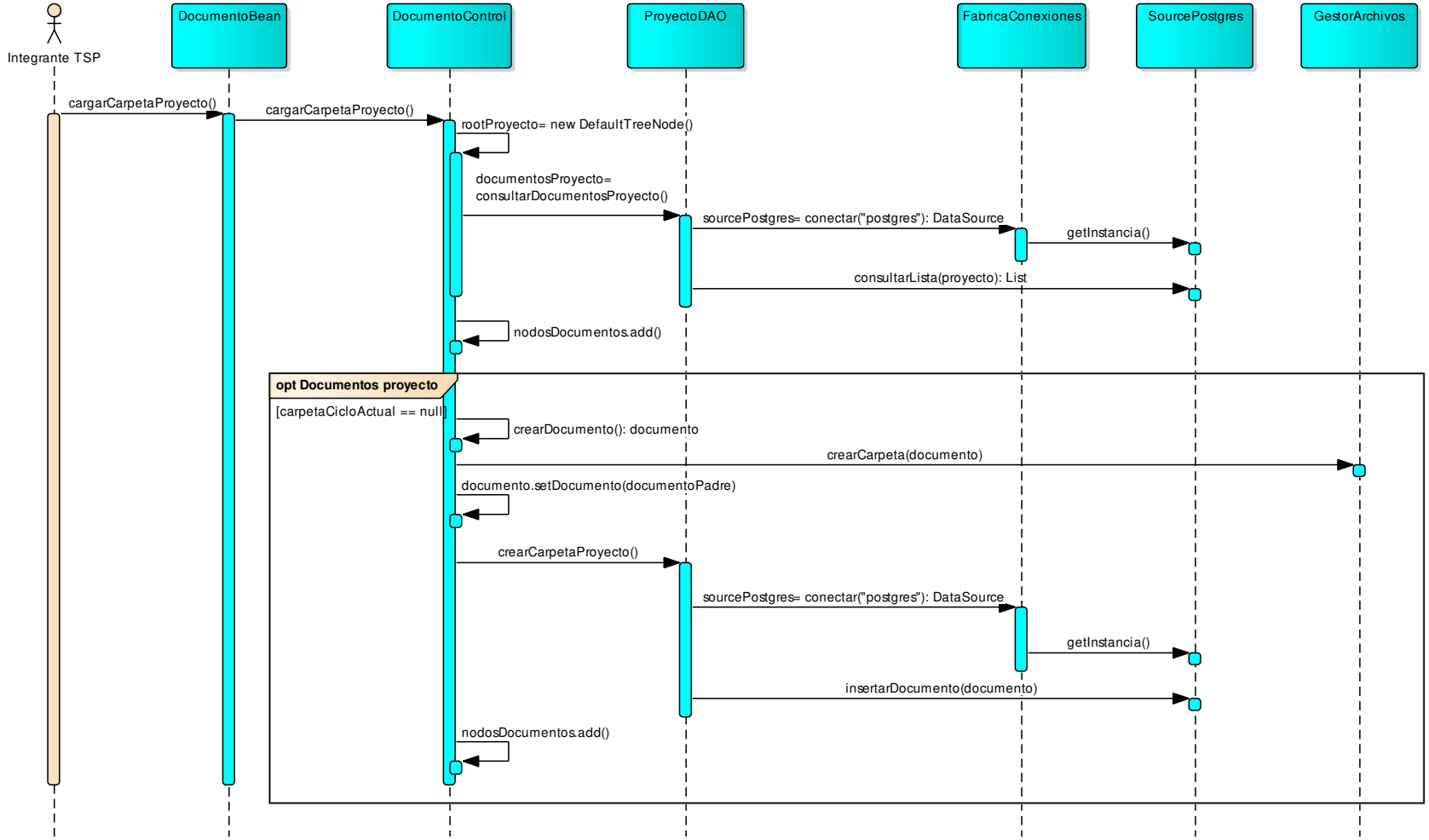
18. Revisión de código

La comparación de códigos en tamaño se gestiona por el controlador *CodigoControl* tal que el usuario sube el comprimido de sus dos versiones del proyecto, valida el contenido y los compara para finalmente mostrar el resultado al usuario.



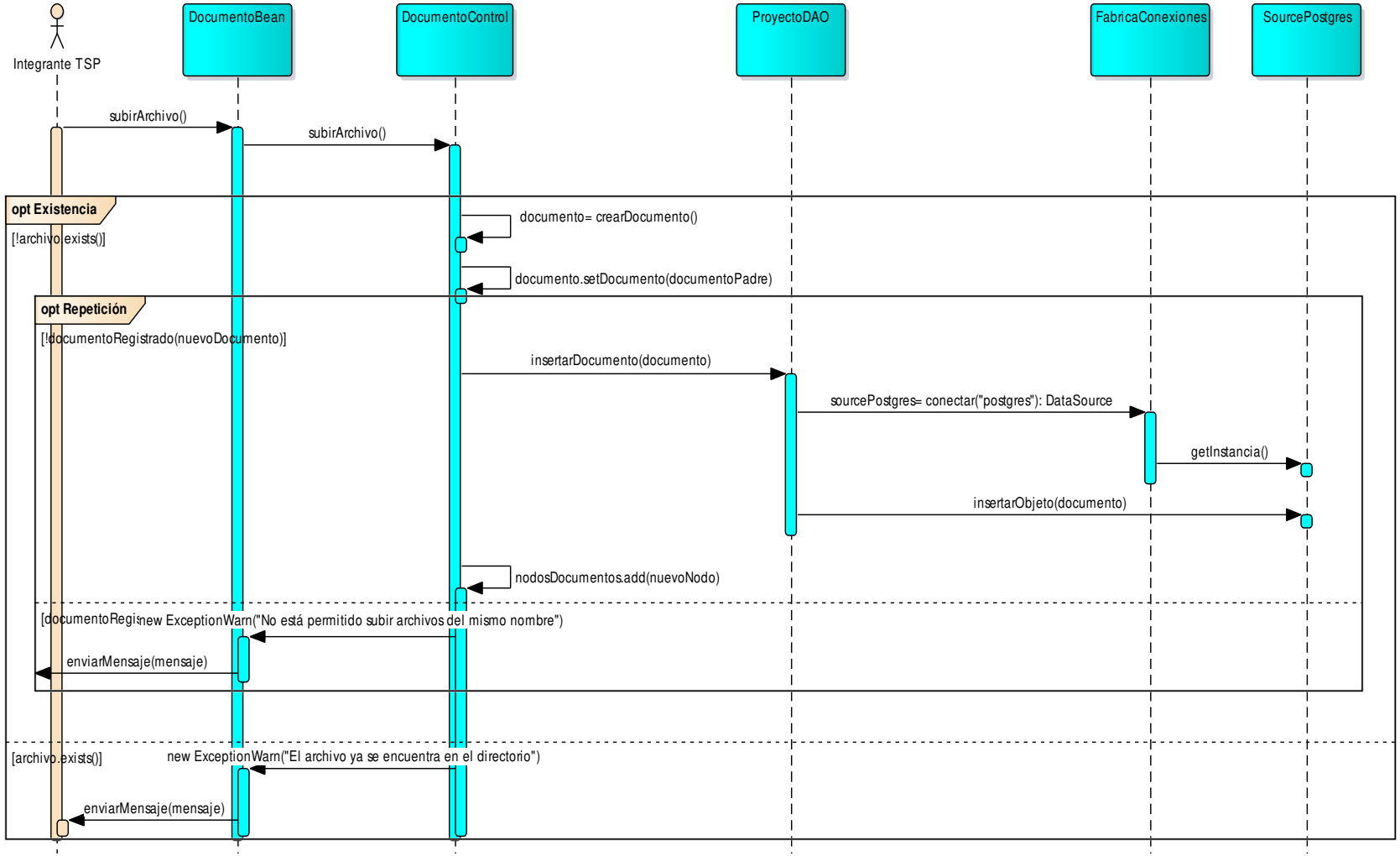
19. Gestión del cuaderno de proyecto.

DocumentoControl se encarga de brindar al usuario la carpeta del proyecto, tal que, de no poseer el directorio del ciclo actual, lo construye y registra en la base de datos.



20. Adjuntar documentos

El controlador *DocumentoControl* gestiona la carga de archivos al sistema, brindar su descarga y registrar su existencia en la base de datos.



ANEXO 3: Documentos del modelo no relacional

1. Formato LOGD

```
{
  "_id" : #,
  "atributos" : [
    { "atributo" : "parte",
      "valor" : "#"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "fecha", "valor" : "#" }, {"atributo" : "numero", "valor" : "#" },
                {"atributo" : "tipo", "valor" : "#" }, {"atributo" : "etapa inyección", "valor" : "#" },
                {"atributo" : "etapa remoción", "valor" : "#" }, {"atributo" : "tiempo arreglo", "valor" : "#" },
                {"atributo" : "defecto arreglado" }, {"atributo" : "descripción", "valor" : "#" },
                {"atributo" : "autor", "valor" : "#" } ] }[,...]
      ],
      "atributo" : "contenido iterable"
    }
  ],
  "nombre" : "logd",
  "proyecto" : "#",
  "ciclo" : #
}
```

2. Formato LOGT

```
{
  "_id" : #,
  "autor" : "#",
  "nombre" : "logt",
  "rol" : "#",
  "proyecto" : "#",
    "fecha" : "#",
  "ciclo" : #,
  "atributos" : [
    { "atributos" : [
          { "atributo" : "Fecha-hora inicio",
            "valor" : "#" },
          { "atributo" : "Fecha-hora finalización",
            "valor" : "#" },
          { "atributo" : "Tiempo dedicado",
            "valor" : "#" },
          { "atributo" : "Tiempo interrumpido",
            "valor" : "#" },
          { "atributo" : "Tarea",
            "valor" : "#" },
          { "atributo" : "Comentarios",
            "valor" : "#" }
        ]
    }[,...]
  ]
}
```

3. Formato PEER

```
{
   "_id" : #,
   "nombre" : "peer",
   "fecha" : "#",
   "proyecto" : "#",
   "ciclo" : #,
   "autor" : "#",
   "atributos" : [
      {
         "atributo" : "Trabajo y dificultad",
         "atributos" : [
            {
               "atributo" : "Líder de proyecto",
               "atributos" : [
                  {  "atributo" : "Trabajo requerido",
                     "valor" : "#"},
                  {  "atributo" : "Dificultad del rol",
                     "valor" : "#"}
               ]
            }[,...]
            {
               "atributo" : "Total",
               "atributos" : [
                  {  "atributo" : "Trabajo requerido",
                     "valor" : "#"},
                  {  "atributo" : "Dificultad del rol",
                     "valor" : "#"}
               ]
            }
         ]
      },
      {
         "atributo" : "Criterios equipo",
         "atributos" : [
            {  "atributo" : "Espíritu de equipo",
               "valor" : "#"}[,...]
         ]
      },
      {
         "atributo" : "Contribución general",
         "atributos" : [
            {  "atributo" : "Líder de proyecto",
               "valor" : "#"}[,...]
         ]
      },
      {
         "atributo" : "Ayuda y soporte",
         "atributos" : [
            {  "atributo" : "Líder de proyecto",
               "valor" : "#"}[,...]
         ]
      },
      {
         "atributo" : "Desempeño",
         "atributos" : [
            {  "atributo" : "Líder de proyecto",
               "valor" : "#"}[,...]
         ]
      }
   ]
}
```

4. Formato SCHEDULE

```
{
  "_id" : #,
  "nombre" : "schedule",
  "proyecto" : #,
  "fecha" : "#",
  "ciclo" : #,
  "atributos" : [
    { "atributos" : [
        { "atributo" : "N. Semana",
          "valor" : "#" },
        { "atributo" : "Fecha",
          "valor" : "#" },
        { "atributo" : "Plan",
          "atributos" : [ { "atributo" : "Horas directas", "valor" : "#" },
                {"atributo" : "Horas acumuladas", "valor" : "#" }, {"atributo" : "VP Acumulado", "valor" : "#" } ] },
        { "atributo" : "Real",
          "atributos" : [ { "atributo" : "Horas de equipo", "valor" : "#" }, {"atributo" : "Horas acumuladas", "valor" : "#" },
                {"atributo" : "VG Semanal", "valor" : "#" }, {"atributo" : "VG Acumulado", "valor" : "#" } ] }
      ]
    } [,...]
      ]
    }
  ]
}
```

5. Formato TASK

```
{
  "_id" : #,
  "nombre" : "task",
  "proyecto" : "#",
  "ciclo" : #,
  "atributos" : [
    { "atributos" : [
        { "atributo" : "Tarea",
          "atributos" : [ { "atributo" : "Etapa", "valor" : "#" }, {"atributo" : "Parte", "valor" : "#" },
                {"atributo" : "Nombre de la tarea", "valor" : "#" }, {"atributo" : "N. Integrts.", "valor" : "#" } ] },
        { "atributo" : "Horas del plan",
          "atributos" : [ { "atributo" : "Líder de proyecto", "valor" : "#" }, {"atributo" : "Gerente de desarrollo", "valor" : "#" },
                {"atributo" : "Gerente de planeación", "valor" : "#" },
                {"atributo" : "Gerente de calidad/proceso", "valor" : "#" }, {"atributo" : "Gerente de soporte", "valor" : "#" },
                {"atributo" : "Total horas", "valor" : "#" }, {"atributo" : "Acumulado horas", "valor" : "1.0" } ] },
        { "atributo" : "Tamaño del plan",
          "atributos" : [ { "atributo" : "Unidad de tamaño", "valor" : "#" }, {"atributo" : "Tamaño", "valor" : "#" },
                {"atributo" : "N. Semana", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" }, {"atributo" : "VP Acumulado", "valor" : "#" } ] },
        { "atributo" : "Real",
          "atributos" : [ { "atributo" : "Horas", "valor" : "#" }, {"atributo" : "Horas acumuladas", "valor" : "#" }, {"atributo" : "N. Semana", "valor" : "#" } ] }
      ]
    }[,...]
  ]
}
```

6. Formato SUMP


```
{
  "_id" : #,
  "atributos" : [
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Páginas de requerimientos (SRS)" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Código total (T)" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Intervalo de predicción (70%)" }
      ],
      "atributo" : "Tamaño del producto"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Gerencia y miscelanea" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Lanzamiento y estrategia" },...
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Total" }
      ],
      "atributo" : "Tiempo en fase (horas)"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Estrategia y planeación" },...
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Prueba de unidad" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Prueba de construcción e integración" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Prueba de sistema" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Total" }
      ],
      "atributo" : "Defectos inyectados"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Estrategia y planeación" },...
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" }, {"atributo" : "Real porcentaje", "valor" : "#" } ],
          "atributo" : "Total" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Prueba de aceptación" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Vida de producto" }
      ],
      "atributo" : "Defectos removidos"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Código/hora" },
        { "atributos" : [ { "atributo" : "Real", "valor" : "#" } ],
          "atributo" : "CPI (índice de costo-desempeño)" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Porcentaje reuso" }
      ],
      "atributo" : "Resumen"
    }
  ],
}
```

7. Formato SUMQ


```
{
  "_id" : #,
  "atributos" : [
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Real", "valor" : "#" } ],
          "atributo" : "En compilación" },...
      ],
      "atributo" : "Porcentaje libre de defectos"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Inspección de requerimientos" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Inspección de diseño de alto nivel" }
      ],
      "atributo" : "Defectos/página"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Revisión de diseño detallado" },...
          "atributo" : "Prueba de aceptación" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Vida de producto" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Total" }
      ],
      "atributo" : "Defectos/Código"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Revision diseño detallado/Diseño detallado" },
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Revisión de código/Compilación" }
      ],
      "atributo" : "Proporciones de defecto"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Inspección de requerimientos/requerimientos" },...

      ],
      "atributo" : "Proporciones de tiempo de desarrollo"
    },
    { "atributos" : [
        { "atributo" : "Plan",
          "valor" : "#" },
        { "atributo" : "Real",
          "valor" : "#" }
      ],
      "atributo" : "A/FR"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Estrategia y planeación" },...
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Prueba de aceptación" }
      ],
      "atributo" : "Rendimientos de fase"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Antes de compilación" },...
      ],
      "atributo" : "Rendimientos de proceso"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Estrategia y planeación" },...
      ],
      "atributo" : "Tasa de inyección de defectos"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Estrategia y planeación" },...
      ],
      "atributo" : "Tasa de remoción de defectos"
    },
    { "atributos" : [
        { "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ],
          "atributo" : "Inspección de requerimientos" },...
      ],
      "atributo" : "Inspección/Proporciones de revisión"
    }
  ],
}
```

8. Formato WEEK


```
{
  "_id" : #,
  "nombre" : "week",
  "proyecto" : "#",
  "fecha" : "#",
  "ciclo" : #,
  "atributos" : [
    { "atributo" : "semana",
      "valor" : "#"
    },
    { "atributo" : "Información semanal",
      "atributos" : [
        { "atributo" : "Horas Semana",
          "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ] },
        { "atributo" : "Horas Ciclo",
          "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ] },
        { "atributo" : "VG Semana",
          "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ] },
        { "atributo" : "VG Ciclo",
          "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ] },
        { "atributo" : "Total horas Fase Actual",
          "atributos" : [ { "atributo" : "Plan", "valor" : "#" }, {"atributo" : "Real", "valor" : "#" } ] }
      ]
    },
    { "atributo" : "Información semanal integrantes",
      "atributos" : [
        { "atributo" : "Líder de proyecto",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" } ] },
        { "atributo" : "Gerente de desarrollo",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" } ] },
        { "atributo" : "Gerente de planeación",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" } ] },
        { "atributo" : "Gerente de calidad/proceso",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" } ] },
        { "atributo" : "Gerente de soporte",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "VP", "valor" : "#" } ] },
        { "atributo" : "Totales integrantes",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" } ] }
      ]
    },
    { "atributo" : "Tareas completas",
      "atributos" : [
        { "atributo" : "Tarea",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "Semana planeada", "valor" : "#" }, {"atributo" : "Nombre tarea", "valor" : "#" } ] }[,...]
        { "atributo" : "Total",
          "atributos" : [ { "atributo" : "Horas planeadas", "valor" : "#" }, {"atributo" : "Horas reales", "valor" : "#" }, {"atributo" : "VG", "valor" : "#" }, {"atributo" : "Nombre tarea", "valor" : "#" } ] }
      ]
    },
    { "atributo" : "Comentarios",
      "valor" : "#"
    }
  ]
```

9. Formato CCR

```
{
  "_id" : #,
  "autor" : "#",
  "atributos" : [
    { "atributo" : "parte",
      "valor" : "#"
    },
    { "atributo" : "aprobación",
      "valor" : "#"
    },
    { "atributo" : "Fecha aprobación",
      "valor" : "#"
    },
    { "atributos" : [
        { "atributo" : "Razón del cambio",
          "valor" : "#" },
        { "atributo" : "Beneficios del cambio",
          "valor" : "#" },
        { "atributo" : "Impacto del cambio",
          "valor" : "#" },
        { "atributo" : "Descripción del cambio",
          "valor" : "#" }
      ],
      "atributo" : "Información del cambio"
    }
  ],
  "nombre" : "ccr",
  "fecha" : "#",
  "proyecto" : "#",
  "ciclo" : #
}
```

10. Formato STRAT

```
{
  "_id" : #,
  "atributos" : [
    {
      "atributos" : [
        {  "atributo" : "parte",
           "valor" : "#"},
        {  "atributo" : "funcion",
           "valor" : "#"},
        {  "atributos" : [
             {  "atributo" : "C1",
                "valor" : "#"}
           ],
           "atributo" : "TAM_Ciclo"
        },
        {
          "atributos" : [
            {  "atributo" : "C1",
               "valor" : "#"}
          ],
          "atributo" : "Horas_Ciclo"
        }
      ]
    }[...]
  ],
  "nombre" : "strat",
  "fecha" : "#",
  "proyecto" : "#",
  "ciclo" : #
}
```

11. Formato PIP

```
{
  "_id" : #,
  "nombre" : "pip",
  "fecha" : "#",
  "proyecto" : "#",
  "ciclo" : #,
  "autor" : "#",
  "atributos" : [
    {  "atributo" : "parte",
       "valor" : "#"},
    {  "atributo" : "Prioridad",
       "valor" : "#"},
    {  "atributo" : "Descripción del problema",
       "valor" : "#"},
    {  "atributo" : "Descripción de la propuesta",
       "valor" : "#"},
    {  "atributo" : "Fecha revision",
       "valor" : "#"},
    {  "atributo" : "Cambios",
       "valor" : "#"}
  ]
}
```

12. Formato ITL

```
{
  "_id" : #,
  "atributos" : [
    {
      "atributo" : "parte",
      "valor" : "#"
    },
    {
      "atributos" : [
        {
          "atributos" : [
            {  "atributo" : "fecha",
               "valor" : "#"},
            {  "atributo" : "riesgo-problema",
               "valor" : "#"},
            {  "atributo" : "numero",
               "valor" : "#"},
            {  "atributo" : "prioridad",
               "valor" : "#"},
            {  "atributo" : "autor",
               "valor" : "#"},
            {  "atributo" : "fecha seguimiento",
               "valor" : "#"},
            {  "atributo" : "fecha resolución",
               "valor" : "#"},
            {  "atributo" : "descripción",
               "valor" : "#"}
          ]
        }[,...]
      ],
      "atributo" : "contenido iterable"
    }
  ],
  "nombre" : "itl",
  "fecha" : "#",
  "proyecto" : "#",
  "ciclo" : #
}
```

ANEXO 4: Cartilla TSP

CARTILLA TSP

1

Visión general de TSP

El software industrial es desarrollado por equipos porque son más efectivos. Son capaces de llevar planes de proyecto, proceso común de trabajo y una efectiva comunicación que con llevan a productos de alta calidad.

1.1 ¿Qué es TSP?

TSP se define como una herramienta de trabajo o como un curso de alto nivel de licenciatura en la ingeniería de software de equipo. Se enfatiza en balancear el proceso, producto, trabajo en equipo y aprovecharse de la experiencia en la industria in la planeación y manejo de proyectos de software. TSP guía al equipo paso a paso, y le muestra cómo aplicar la ingeniería de software y sus principales procesos en el ambiente de trabajo de equipo.

Por qué los equipos de ingeniería necesitan un proceso

Reunir un grupo de personas no produce automáticamente un equipo. Los pasos requeridos para construir un equipo no son evidentes; requieren de mecanismos. Los integrantes de un equipo deben definir cómo trabajar juntos y para ello definir una estrategia de trabajo. Ellos deben asignar las tareas entre los miembros del equipo, coordinada cada una de éstas y reportar los progresos.

1.2 Principios de TSP

1. El aprendizaje es más efectivo cuando se sigue un proceso definido y se consigue una rápida retroalimentación. Los scripts y formularios de TSP proveen una definición moderada y un marco de referencia repetible para el trabajo de ingeniería en equipo. TSP provee un rápido rendimiento en la retroalimentación porque produce en forzosos ciclos cortos de desarrollo y resultados de evaluaciones por cada ciclo.
2. El productivo trabajo en equipo requiere una combinación de metas específicas, un en un ambiente de trabajo colaborativo.
3. Cuanto se ha luchado con los problemas de un actual proyecto y se ha seguido una guía efectiva de soluciones, se apreciarán los beneficios de las buenas prácticas de desarrollo.
4. La instrucción es más efectiva cuando se construye bajo un cuerpo de conocimiento previo. Se obtiene una experiencia más gratificante con equipos de software y cusos de equipos de software.

1.3 Diseño de TSP

1. Provee un simple marco de referencia que se basa en PSP.

2. Desarrollo de productos en varios ciclos.
3. Establece medidas estándar para la calidad y el desempeño.
4. Provee medidas precisas para equipos y estudiantes.
5. Hace uso de roles y equipos de evaluación.
6. Requiere procesos de disciplina.
7. Provee una guía para los trabajos de equipo.

1.4 Flujo y estructura de TSP

TSP cuenta con varios ciclos. El primer ciclo inicia con el lanzamiento, en el cual el instructor describe los objetivos generales del producto. Posteriormente, el equipo sigue a través del TSP en siete pasos: estrategia, planeación, requerimientos, diseño, implementación, pruebas y postmortem. En el ciclo 2 los ingenieros repiten los mismos pasos, esta vez orientados hacia la mejora del producto producido en el ciclo 1.

Estrategia de desarrollo cíclico

Cuando se inicia una estrategia de desarrollo cíclico, lo mejor es comenzar con la versión vital mínima de producto. Al decidir el tamaño y el contenido de cada ciclo, se deben considerar las siguientes limitaciones.

1. Cada ciclo debe producir una versión probable que es un subconjunto apropiado del producto final.
2. Cada ciclo debe ser lo suficientemente pequeño como para ser fácilmente desarrollo y probado en el tiempo disponible.
3. Cuando se combinan, los productos del ciclo deben producir el producto final deseado.

El TSP comienza por tener equipos que producen una estrategia de desarrollo. Para empezar, recogiendo la base más pequeña razonable para desarrollar durante el primer ciclo. Luego estimar los tamaños de las funciones que se planean añadir en cada ciclo siguiente. Este enfoque garantiza que se completan subconjuntos del trabajo inicial del producto.

1.5 El proceso de TSP

La secuencia de comandos de la tabla 1.1 muestra el flujo general de TSP. Cada paso de guion es adoptado por una o más detalladas secuencias de comandos. Todos los guiones de TSP se incluyen también en el apéndice D. Cada secuencia de comandos se inicia con una breve exposición de la finalidad general de la actividad (por ejemplo desarrollar un documento de requerimientos, producir un diseño o llevar a cabo una prueba). Cada script tiene también criterios de entrada y de salida.

1.6 La Estructura de libros de texto y de flujo

Es importante leer el libro y seguir los pasos especificados en el documento DEV de la tabla 1.1.

1.7 Resumen

Los principios básicos de TSP son:

1. El aprendizaje es más eficaz cuando los estudiantes siguen los pasos definidos y repetibles y obtener una rápida retroalimentación sobre su trabajo.

2. el trabajo en equipo productivo requiere un objetivo definido el equipo, un entorno de trabajo eficaz y capaz de entrenamiento y liderazgo.

3. Cuando los estudiantes están expuestos a los problemas de los proyectos de desarrollo realistas y luego guiados a soluciones eficaces, que obtienen una mejor apreciación del valor de la ingeniería de sonido.

4. La instrucción es la masa efectiva cuando se basa en el cuerpo a disposición de la ingeniería previa, científica, y la experiencia pedagógica.

A partir de estos cuatro principios, el diseño TSP implica siete opciones.

l. Proporcionar un marco simple que se basa en el fundamento de la PSP.

2. Desarrollo de productos en varios ciclos.

3. Establecer las medidas estándares de calidad y rendimiento.

4. Proporcionar las medidas precisas para que los equipos y los estudiantes.

S. Use papel y equipo de evaluaciones,

6. Requerir disciplina de proceso.

7. Proporcionar orientación sobre problemas de trabajo en equipo.

El proceso TSP sigue una estrategia de desarrollo cíclico. Al comenzar un pequeño conjunto de funciones iniciales, el equipo puede completar rápidamente la primera versión del trabajo para posteriormente mejorar la planificación y desarrollo durante el segundo ciclo. Si hay tiempo, un tercer ciclo permite que el aprendizaje se refuerce aún más. La estrategia de desarrollo cíclico es muy similar a los procesos seguidos por los grupos de desarrollo de software a gran escala con éxito.

2

La lógica del proceso de trabajo en equipo

2.1 Por qué los proyectos fallan

Cuando los proyectos fallan, generalmente es porque los problemas de trabajo en equipo y no por cuestiones técnicas.

2.2 Problemas comunes de equipo

Estudios demuestran que los más comunes problemas conciernen al liderazgo, la cooperación, participación, procastinación, calidad y evaluación.

2.3 ¿Qué es un equipo?

Un equipo consta de:

a. Al menos dos personas, que
b. Trabajan hacia un objetivo/meta/misión común, donde
c. A cada persona le ha sido asignada un rol específico o funciones a realizar, y donde
d. La finalización de la misión requiere de la dependencia entre los miembros del equipo.

Tamaño del equipo

Los equipos pueden ser de casi cualquier tamaño, partiendo de dos. Sin embargo, en situaciones prácticas, los equipos son más efectivos cuando desarrollan relaciones con casi todos los miembros. El tamaño del equipo lo suelen definir las industrias, es decir, la organización donde se construya; para proyectos grandes, se construyen equipos de hasta 20 personas, cada una de ellas trabajando bajo la supervisión de un gerente o supervisor.

En cuanto a los estudiantes, los equipos suelen tener entre cuatro y doce miembros, dependiendo la dificultad de los proyectos a desarrollar. La experiencia permite afirmar que los equipos entre cuatro y ocho miembros suelen ser más efectivos.

2.4 Construyendo equipos efectivos

Para construir equipos efectivos, se necesitan los tipos adecuados de tareas y condiciones de trabajo. El equipo debe tener un alto compromiso y un ambiente de trabajo que soporte dicho trabajo en equipo. Los miembros del equipo requieren de cohesión, metas, realimentación, y comúnmente, un marco de trabajo.

2.5 Cómo desarrollan los equipos

Algunos desarrollos de equipo pueden ocurrir por suerte, o pueden resultar de un conciso proceso de desarrollo en equipo. Este trabajo consiste en iniciar con tareas individuales, de tal modo que cada miembro tiene unas metas definidas. Como equipo, los integrantes aceptan las metas comunes con una especial significancia. La asignación de las metas individuales puede llevarse a cabo en forma aleatoria o de acuerdo a la propuesta del equipo mismo.

2.6 Cómo construir equipos TSP

Grupos pequeños puede conllevar a equipos más efectivos. Las técnicas de ayuda a equipos permiten construir relaciones interpersonales que ayudan al trabajo conjunto y al soporte mutuo. Las guías de TSP proporcionan os pasos de construcción de equipo para definir las metas, roles, establecer planes y mantener la comunicación entre los miembros.

2.7 Resumen

En este capítulo se presenta una visión general y la lógica de TSP. Cubre lo que son los equipos, lo que se hace en el trabajo en equipo y cómo se resuelven problemas en equipo. El TSP está diseñado para aprovechar las ventajas naturales de los equipos y reducir al mínimo la probabilidad o el impacto de las debilidades de los equipos.

Un problema significativo de las personas se refiere a la capacidad de manejar las presiones del cronograma. El TSP ayuda a que los equipos manejen dichas presiones mediante el desarrollo de planes detallados y proporcionando un proceso de desarrollo estructurado.

El secreto de los equipos exitosos, es la construcción de una misión clara, distinta y con alto control de tareas. El entorno también apoya al buen desempeño del equipo; a animar a los miembros a planificar el trabajo y permitirles mantener una disciplina personal.

Cuando los equipos se unen, pueden conllevar a resultados extraordinarios. Definen sus propios procesos, establecen y trabajan bajo un plan común. Mantener la energía y el entusiasmo también ayuda a tener equipos verdaderamente excepcionales.

3

Lanzamiento de un proyecto de equipo

3.1 ¿Por qué dirigir un lanzamiento de equipo?

Los equipos necesitan establecer sus relaciones de trabajo, determinar los roles, y añadir sus metas. El lanzamiento es el primer paso, y se estima un tiempo de una hora para lograrlo.

El instructor es quien toma la decisión final en la asignación de roles.

3.2 Metas de equipo

1. Producir un producto de calidad

1.1 % defectos encontrados dentro de la primera compilación 80%

1.2 Número de defectos encontrados en la prueba de sistema 0

1.3 Funciones de requerimientos para cuando el proyecto esté completo 100%

2. Correr un proyecto productivo y bien manejado

2.1 Error en el estimado del tamaño del producto < 20%

2.2 Error en las horas estimadas de desarrollo < 20%

2.3 % Información registrada en el cuaderno del proyecto 100%

3. Terminar a tiempo

3.1 Días antes o después de completar el desarrollo. < 4.

3.3 Metas de miembros-equipo

1. Ser un miembro efectivo y cooperativo

1.1 La evaluación PEER promedio del rol para ayuda y soporte debe ser mayor a 3.

1.2 La evaluación PEER promedio para la contribución general debe ser mayor a 3.

2. Hacer traabajo personal disciplinado de manera consitente.

2.1 % de información personal registrada y en el cuaderno del proyeto 100%

2.2 Porcentaje de semanas para completar WEEK 100%

3. Planear y rastrear todo el trabajo personal

3.1 Porcentaje de información peronal del proyecto registrada en SUMP y SUMQ 100%

3.2 Porcentaje de tareas de proyecto con información planeada y real completa en el formato TASK 100%

4. Producir productos de calidad (INS LOGTEST)

4.1 % promedio de defectos encontrados antes de la primera compilación > 70%

4.2 Densidad de defectos encontrada durante la compilación < 10KLOC

4.3 Densidad de defectos encontrada durante la prueba de unidad < 5KLOC

4.4 Densidad de defectos encontrada después de la prueba de unidad 0

3.4 Metas de rol

Líder de equipo

1. Construir y mantener un equipo efectivo

2. Motivar a todos los miembros del equipo a participar activamente en el proyecto

3. Resolver todos los problemas que presenten los integrantes del problema del equipo. Los que exponen ellos

4. Mantener al instructor informado acerca del progreso del equipo

5. Desempeñarse efectivamente como facilitador de las reuniones de equipo

Gerente de desarrollo (producir productos funcionales)

1. Producir producto superior

2. Utilizar por completo las habilidades de los integrantes

Gerente de planeación (Guiar al equipo en hacer un plan detallado)

1.Producir un plan completo, preciso y exacto, para el equipo y cada integrante del equipo

2. Reportar de manera exacta el estatus semanal del equipo

Gerente de calidad/proceso (asegurar que el equipo use TSP para producir un producto libre de defectos)

1. Todos los integrantes del equipo reportan de manera precisa y utilizan apropiadamente la información de TSP

2. El equipo sigue fielmente TSP y produce un producto de calidad.

3. Todas las inspecciones de equipo son apropiadamente reportadas y moderadas.

4. Todas las juntas de equipo se reportan de manera precisa y los reportes se almacenan en el cuaderno del proyecto.

Gerente de soporte (Asegurar eque el proyecto es apropiadamente controlado)

1. El equipo tiene herramientas adecuadas y métodos para soportar el trabajo.

2. No se hacen cambios no autorizados a los productos base

3. Todos los riesgos y problemas del equipo se registran en el lanzamiento de problemas y se reportan cada semana

4. El equipo cumple sus metas de reúso para el ciclo de desarrollo.

Responsabilidades	Líder de equipo	Gerente de desarrollo	Gerente de planeación	Gerente de proceso/calidad	Gerente de requerimientos/soporte
Construir y mantener un equipo eficaz	x				
Resolver los problemas entre los miembros del equipo	x				
Rastrear y reportar el progreso del equipo	x				
Actuar como facilitador de la reunión	x				
Interface con el instructor	x				
Mantener el cuaderno de proyectos	x				
Ayudar al equipo asignar tareas	x				
Conducir el trabajo de desarrollo		x			
Equipo de planificación de plomo y seguimiento de los progresos			x		
la planificación de entrega de calidad y seguimiento				x	
Proporcionar apoyo proceso de equipo				x	
Mantener los estándares del equipo y glosario				x	
Manejar el cumplimiento de la presentación de informes				x	
Alertar al equipo a problemas de calidad				x	
Obtener herramientas y el apoyo necesarios					x
gestión de la configuración mango					x
Dirigir la Junta de Control de Cambios					x
Actuar como defensor de equipo de reutilización					x
Manejo de emisión y seguimiento de riesgos					x
Mantener el sistema de glosario					x
Desarrollar el producto	x	x	x	x	x
Hacer planes personales	x	x	x	x	x
Realizar un seguimiento del trabajo personal	x	x	x	x	x
Producir productos de calidad	x	x	x	x	x
Siga las prácticas personales disciplinados	x	x	x	x	x

3.5 Lanzamiento de scripts TSP

El formato de lanzamiento se refiere a LAU1 y LAUn.

Criterio de entrada

1. Los estudiantes han completado el curso PSP

2. Los estudiantes han leído el libro de texto capítulos 1, 2,3 y Apéndice A

En la revisión general del curso, el instructor revisa los objetivos TSP del curso y discute lo que se espera lograr. Este es el momento de preguntar acerca de cómo se evaluará el trabajo y discutir las calificaciones individuales y en equipo. El objetivo general del curso es que los miembros del equipo trabajen cooperativamente para producir un efectivo resultado general.

Para ser más efectivo, se necesitan combinar los talentos y habilidades de los integrantes. Para proveer la información necesaria, se debe completar el formato INFO. Esta información se utiliza para asignar el rol y las tareas.

En el siguiente paso del proceso, el instructor describe el producto a construir y contesta las preguntas relacionadas al curso.

Durante la siguiente reunión, el instructor lleva a cabo las asignaciones de equipo y rol. Cuando es posible, el instructor mantiene al mismo equipo durante todo el curso.

Posteriormente, el instructor revisa el proceso de fijación de metas y las metas de TSP.

Después de que los equipos han establecido metas y roles, es tiempo de llevar a cabo la primera reunión de equipo. Le provee una oportunidad al equipo para discutir y ponerse de acuerdo en los objetivos para el ciclo de desarrollo. Para llevar a cabo la junta, debe seguirse el formato WEEK. Lo más importante de la reunión semanal es reunir y analizar la información del equipo de la semana anterior y para el ciclo de desarrollo hasta la fecha. Todos los integrantes entregan su formato WEEK al gerente de planeación y éste elabora el WEEK del equipo.

Una parte importante de la reunión semanal es que los miembros se pongan de acuerdo acerca de la información que se le entregará al gerente de planeación y cuándo se le entregará.

En este punto, se ha completado el proceso de lanzamiento y el equipo está listo para comenzar el proyecto, siguiendo el formato STRAT.

3.6 Resumen

En este capítulo se describen los pasos en el proceso de lanzamiento TSP. Durante el lanzamiento, el instructor lleva a los estudiantes a través del resumen del curso, la recopilación de información de los estudiantes, los objetivos del producto, trabajos en equipo, fijación de objetivos de equipo y la reunión de equipo. Luego los equipos mantienen su primera reunión semanal y están de acuerdo en cómo y cuándo van a proporcionar los datos semanales con el gerente de planificación. Tras el lanzamiento del equipo, los equipos siguen el proceso TSP para iniciar sus proyectos.

El proceso de lanzamiento se utiliza para formar y construir equipos, determinar los roles de los miembros del equipo, establecer los objetivos del equipo, y ayudar a los equipos a que decidan sus prácticas de trabajo. Definir y acordar de qué se encargará cada función es un primer paso esencial en la formación de equipos. El instructor hace que las asignaciones

iniciales de rol, que en posteriores ciclos se puede asignar de acuerdo a las preferencias de los estudiantes.

El establecimiento de objetivos se realiza al inicio de cada ciclo del proyecto TSP. Los objetivos del equipo establecen el marco para la estrategia y el plan. Estos, a su vez, proporcionan la base para el equipo posteriormente. Los equipos deben saber cómo medir los objetivos, establecer las metas no agresivas y proveer los esfuerzos para lograr dichos objetivos. La única manera de aprender a establecer objetivos útiles, es trabajar en ellos para satisfacerlos.

Para TSP, los tres objetivos básicos son: producir un producto de calidad, ejecutar un proyecto productivo y bien administrado, y acabar en el tiempo estimado. Las medidas de éxito son que cada equipo produzca un producto de calidad, con funciones previstas de acuerdo al calendario previsto.

4

Desarrollo de estrategia

4.1 Primera planeación

TSP requiere que se haga una planeación antes de producir los requerimientos. Existen tres motivos para planear antes de comenzar un proyecto:

1. Durante el proceso del desarrollo del plan, los equipos obtienen una apreciación común del trabajo que deben realizar.
2. El plan provee la base para la rastreabilidad del trabajo.
3. Si no se comienza realizando un plan y revisándolo con los gerentes o instructores, los equipos terminan comprometiéndose a fechas que tal vez no puedan cumplir.

4.2 ¿Qué es una estrategia?

La decisión estratégica que se ha tomado en TSP, es llevar a cabo el desarrollo del producto en un proceso cíclico. El número de ciclos es decidido por un instructor pero el contenido de cada uno depende de los integrantes de un equipo.

4.3 Diseño conceptual

Es el punto de inicio para la planeación de proyecto. La razón por la cual se necesita un diseño conceptual es que debe realizarse un plan para construir un producto. Para llevar a cabo un diseño conceptual, se tienen que realizar las siguientes preguntas:

1. ¿Cómo construiría este producto?
2. ¿Cuáles son los principales componentes que necesito para construir este producto?
3. ¿Qué funciones debe proveer el producto?
4. ¿Cuán grandes deben ser los componentes?

4.4 Gestión de riesgos

El motivo por el cual debe pensarse en riesgos en las primeras etapas de un proyecto es para que los mismos puedan evitarse o controlarse. Deben llevarse a cabo acciones mitigantes efectivas para esta gestión de riesgos.

4.5 Estrategia de reutilización

Uno de los riesgos clave en cualquier proyecto de software es el crecimiento del código. Una manera de reducir este riesgo es adoptando una estrategia de reutilización. Si se está desarrollando un código que cubre múltiples necesidades, se pueden añadir funciones con

poco o nada de código adicional. Esta estrategia puede reducir el esfuerzo requerido de desarrollo, a veces de manera sustancial.

4.6 Estrategia de scripts

Criterio de Entrada

1. Los estudiantes han leído el Capítulo 4
2. El instructor ha revisado y discutido el proceso TSP
3. El instructor ha descrito los objetivos generales del producto
4. Los equipos de desarrollo se han formado y se han asignado los roles
5. Los equipos se han puesto de acuerdo acerca de los objetivos de su trabajo

Establecer Criterios de Estrategia

Aunque pueden existir muchas estrategias viables para incluso los productos más simples, es importante primero establecer los criterios para evaluar las estrategias y luego utilizar estos criterios para revisar las propuestas. El principal objetivo de la estrategia es minimizar el riesgo de que se exceda el calendario de desarrollo disponible. Los criterios sugeridos para lograr este objetivo son los siguientes:

1. El producto del ciclo 1 provee un subconjunto de trabajo de función mínima del producto final.
2. El producto del ciclo 1 provee una base que puede mejorarse fácilmente.
3. Los productos del ciclo son de alta calidad y pueden probarse fácilmente.
4. El diseño del producto tiene estructura modular que permite a los miembros del equipo trabajar independientemente.

Producir el diseño conceptual

El gerente de desarrollo lidera al equipo en la producción del diseño conceptual. En el primer ciclo de desarrollo, este diseño debe incluir todas las funciones que se pretendan desarrollar en los tres ciclos de desarrollo. Como parte del proceso de estrategia, se va a decidir qué funciones a desarrollar en cada ciclo.

Seleccionar la Estrategia de Desarrollo

Con el criterio establecido, el siguiente paso es producir una estrategia de desarrollo propuesta. Una forma de hacer esto es examinar las funciones deseadas para el producto total y luego seleccionar un subconjunto mínimo de trabajo. El objetivo general es tener diversos elementos funcionales en desarrollo en cada ciclo.

Al desarrollar una estrategia, se debe reconocer que algunos elementos de producto muy probablemente serán más grandes que lo planeado. Entonces, si se pretende incluir mucha función en el primer ciclo, el trabajo puede tomar más de lo planeado y puede no tenerse el tiempo de completar los ciclos de desarrollo subsecuentes. La llave es producir un producto trabajable mínimo que se pueda probar y demostrar.

Aunque el equipo puede seleccionar cualquier método para desarrollar la estrategia, usualmente es una buena idea que el gerente de desarrollo proponga una estrategia inicial y el diseño conceptual. Luego debe revisarla el equipo. Si otros integrantes también desean hacer propuestas, deberían hacerlo. Luego hay que evaluar las propuestas realizadas y ya sea seleccionar una o combinarla y ajustarla para producir una que sea aceptable para el equipo.

Producir el Estimado Preliminar de Tamaño

Al producir la estrategia, se debe estimar cuántos LOCs va a requerir de manera probable cada función del diseño conceptual. Estos estimados son necesariamente hipotéticos pues aún no se ha decidido cómo construir el producto.

Después de postular las funciones y componentes probables, hay que considerar cada elemento del programa y juzgar cuántos LOCs nuevos y cambiados va a necesitar. Luego se deben registrar los estimados en el formato STRAT. Con cada ciclo subsecuente de desarrollo, hay que reevaluar esta estrategia y los estimados de tamaño y tiempo.

Producir el Estimado Preliminar de Tiempo

Usando el tamaño estimado del programa y la tasa de LOC por hora, se debe estimar cuánto tiempo tomará desarrollar cada función de manera probable.

Evaluar riesgos

Se deben evaluar los riesgos de acuerdo a su probabilidad de ocurrencia e impacto. Las clasificaciones son H (alto), M (medio) y L (bajo).

Documentar la estrategia

Al final del paso del desarrollo de la estrategia, hay que utilizar STRAT para documentarla.

Actualizar la Estrategia de Desarrollo

Hay que ingresar los nombres y tamaños de los componentes ya desarrollados y de las funciones que cubren. Luego hay que definir las funciones a añadir para el siguiente ciclo de desarrollo.

El proceso de manejo de configuración tiene diversas funciones clave. Debe registrar lo siguiente:

1. Copias de cada versión de cada elemento del producto
2. Un registro de todos los cambios hechos a cada línea de base
3. Quién hizo el cambio
4. Cuándo se hizo el cambio
5. Qué fue el cambio
6. Por qué se hizo el cambio

El gerente de soporte produce el plan de manejo de configuración y lo revisa con el equipo.

5

Plan de desarrollo

5.1 La necesidad de planificación

La planificación permite trabajar más efectivamente, ya que cuando no se planea, se tiene a hacer las cosas en un orden improductivo. Cuando se planea, existe una mayor probabilidad de cumplir compromisos.

Un punto clave en la planeación, es balancear la carga de trabajo de manera adecuada para así evitar problemas con el cumplimiento de tiempos acordados. Un plan detallado también ayuda a rastrear y gestionar el trabajo de una mejor manera, así como revisar el progreso realizado.

5.2 Proceso de planeación TSP

Los pasos para esto son los siguientes:

1. En la fase de estrategia se produce el diseño conceptual.
2. Durante la fase de estrategia, se estiman los tamaños de diferentes partes del diseño conceptual, se decide qué partes desarrollar en cada ciclo y se ingresa la información en el formato STRAT.
3. Durante la fase de planeación, en el formato SUMS se ingresan los nombres y tamaños estimados de los componentes que se van a desarrollar en el ciclo 1. También se identifican otros productos que se necesitan desarrollar en el ciclo 1.
4. Enlistar las tareas requeridas para construir el producto, estimar cuán larga será cada tarea e ingresar esta información en el formato TASK. También se debe estimar cuánto tiempo va a invertir el equipo completo en el proyecto y se ingresa esta información en el formato SCHEDULE.
5. Llevar a cabo un plan de calidad, SUMQ.
6. Hacer copias de los formatos TASK y SCHEDULE para cada ingeniero de manera que se llenen de manera individual.
7. Utilizar la herramienta TSP para generar formatos TASK y SCHEDULE que muestren el PV y tiempos de compleción para cada tarea personal.
8. Balancear la carga de trabajo con base en la información contenida en los formatos TASK y SCHEDULE.
9. Utilizar la herramienta TSP para producir el plan de equipo final.

5.3 Herramienta de soporte TSP

La herramienta contenida en este libro contiene los registros, formatos y plantillas necesarias para la planeación y rastreo de proyectos TSP. Cabe destacar que aunque la herramienta es un buen auxiliar, no automatiza el trabajo que conlleva esta metodología.

5.4 Desarrollo de scripts de plan

Criterio de Entrada

Hay que asegurarse de haber completado la estrategia de desarrollo y el diseño conceptual.

Planeación de Proyecto Paso 2.1

Enlistar los productos a producir en este ciclo de desarrollo y estimar sus tamaños. Aquí se nombran las partes del sistema y se estima cuán grandes serán. También se identifican otros productos y se estiman sus tamaños.

Planeación de Proyecto Paso 2.2

Registrar la información de producto y tamaño para este ciclo de desarrollo en el formato SUMS.

Planeación de Proyecto Paso 3.1

Producir una lista de tareas. Esta es la lista de los pasos principales en todos los scripts de desarrollo de TSP.

Planeación de Proyecto Paso 3.2

Producir los estimados de tarea para el equipo y de manera individual. Estimar cuánto tiempo tomará cada tarea de manera probable. El equipo completo debe hacer estos estimados.

1. Para cada tarea, ingresar en “Fase” la fase del proceso en la que se llevará a cabo.
2. En “Parte” ingresar el nombre y/o número de parte producida por la tarea (esto utilizando como base la información contenida en el SRS).
3. Ingresar el número de ingenieros que estarán involucrados en la tarea en el espacio marcado como “#Ingenieros” en el formato TASK
4. Ingresar las horas estimadas para cada ingeniero en la columna para ese rol determinado.
5. Si los estimados de tiempo para algunas de estas tareas son mayores a 10 horas, romper las tareas en subtareas más pequeñas y estimar el tiempo requerido para cada una.
6. Ingresar el tiempo total en horas para cada tarea.

Planeación de Proyecto Paso 4.1

Ingresar las horas semanales en el formato SCHEDULE. Estimar cuántas horas les tomará a ti y a tus compañeros dedicarse al proyecto cada semana. Mientras se planea el ciclo 1, incluir horas para ocho o más semanas, al menos hasta el primer intento del plan.

Planeación de Proyecto Paso 4.

Producir los formatos de equipo TASK y SCHEDULE. Después de ingresar las horas semanales en el formato SCHECDULE para el equipo, se usará la herramienta TSP para calcular el valor planeado y la fecha de compleción anticipada para cada tarea. Luego generar los formatos TASK y SCHEDULE completos.

Como paso final en la compleción del plan de equipo, hay que revisar que el plan incluya suficientes horas para producir el producto en el tiempo disponible. Seguir los siguientes pasos:

1. Totalizar las horas que los miembros planean invertir en el proyecto durante este ciclo.
2. Comparar el tiempo total desde el paso 1 con el tiempo total requerido para hacer todas las tareas del ciclo.
3. Si los tiempos de tarea exceden el tiempo de trabajo total planeado, no tiene punto realizar el siguiente paso.
4. Si el trabajo es muy grande, las alternativas son trabajar más o reducir el trabajo planeado. Si se deciden por más horas, ingresar las nuevas horas en el borrador de horario y ver si se ajusta al tiempo disponible.
5. Si se decide reducir la cantidad de trabajo, regresar al paso de estrategia y revisar las funciones a implementar durante este ciclo. Rehacer el plan.

No se deben reducir los estimados.

Después de haber completado los formatos TASK y SCHEDULE para el equipo, hay que completar las porciones de tamaño y tiempo en el formato SUMP.

Planeación de Proyecto Paso 5

Producir el plan de calidad. Para hacer el plan de calidad, se debe comenzar estimando los defectos a inyectar en cada fase. Se debe hacer esto para el producto completo. Luego ingresar la meta de rendimiento para cada fase de remoción de defectos. Finalmente, se genera un plan de calidad de prueba para saber si cumple con estos criterios. Si no, hacer ajustes y corree otro plan de prueba. Seguir estos pasos:

1. Estimar cuántos defectos el equipo inyectará en cada fase. Basar este estimado en el tiempo invertido por fase multiplicado por la tasa de inyección de defectos.
2. Estimar el rendimiento que el equipo alcanzará para cada fase de remoción de defectos. Basar este número en los criterios de rendimiento en la tabla 5.8. Cuando se ingresa el rendimiento, la herramienta calcula los defectos en el producto en esa fase y deduce el porcentaje de rendimiento.

3. Al estimar el rendimiento para las fases de prueba, nota que el rendimiento de prueba depende de la densidad de defectos ingresada en las fases de prueba. Comenzar con las metas de rendimiento para estas fases de prueba y ver si los valores de defectos/KLOC vienen en rangos específicos. Si se encuentran muchos defectos, reducir los rendimientos de fase de prueba hasta que los rendimientos y defectos/KLOC comparan con los criterios.
4. Utilizar la herramienta TSP para generar el plan de calidad de prueba.
5. Comparar los valores en este plan de calidad de prueba con la tabla 5.8. Si el total de defectos/KLOC no están dentro del rango sugerido para los ingenieros entrenados en PSP, hay que incrementar o disminuir los tiempos de desarrollo para el diseño y fases de código o ajustar los defectos inyectados en una o más fases. Continuar ajustando los tiempos de fase o tasas de inyección de defectos hasta que el total de defecto/KLOC cumpla con los criterios.
6. Si las tasas de remoción de defectos y tasas de revisión e inspección no están razonablemente cerca de los criterios, ajustar el tiempo invertido en las fases de remoción de defectos o cambiar los rendimientos de remoción de defectos hasta que estos valores estén más cerca de los criterios. Se ajusta el formato TASK.
7. Si las proporciones de defecto son bajas, reexaminar la información para asegurarse de que se cumplen los demás criterios.
8. Si los números de defectos/KLOC son muy altos en la compilación o prueba, revisar que nuevamente el valor total de defectos/KLOC esté dentro del rango.
9. Repetir pasos 4 a 8 hasta que el plan de calidad cumpla con los objetivos.
10. Utilizar herramienta TSP para generar el formato SUMQ final.

Planeación de Proyecto Paso 6

Hacer planes individuales. Deben llenar los formatos TASK y SCHEDULE de manera personal.

Planeación de Proyecto Paso 7

Balancear la carga de trabajo. Esto con base en los formatos TASK y SCHEDULE. Si se cambia algo en estos formatos, también cambian los formatos SUMP y SUMQ.

Paso Final de Planeación

Producir y distribuir los planes. Cuando los planes individuales estén balanceados, el paso final es producir el plan de equipo consolidado: SUMP, SUMQ, TASK y SCHEDULE.

Criterio de Salida

Se requieren los formatos TASK y SCHEDULE terminados para el equipo y de manera individual. También se requieren los formatos SUMS, SUMP y SUMQ.

5.5 Rastreo de trabajo

El motivo para rastrear el proyecto es para determinar el estatus y la calidad del trabajo. Esta práctica ayuda a gestionar el trabajo personal y permite al líder del equipo reportar el estatus semanal al instructor.

1. Registrar el tiempo en LOGT
2. Ingresar la semana en la que se completan las tareas
3. Generar la información de TASK y SCHEDULE
4. Ingresar los defectos en LOGD
5. Ingresar el tamaño del componente
6. Actualizar el formato SUMP
7. Generar el formato SUMQ
8. Producir el resumen del estatus consolidado del equipo

5.6 Plan de calidad

SUMQ brinda el rendimiento de calidad para cada ensamble del sistema.

PDF mide el porcentaje de los componentes de un producto que no tienen defectos en una fase dada.

Defectos por página: muestra el promedio de número de defectos removidos de cada página de los documentos de requerimientos y HLD.

Defectos/KLOC: número de defectos por KLOC encontrados.

Proporción de defectos: proveen introspección en las revisiones de diseño y código.

Proporciones de tiempo de desarrollo: tiempo invertido en cada fase de desarrollo.

A/FR: apreciación en relación a la proporción de fallas.

Tasas de revisión e inspección: para ello hay que reunir información de diferentes proyectos y utilizarla para establecer tasas objetivo.

Tasas de inyección de defectos: número de defectos inyectados por hora.

Tasas de remoción de defectos: defectos encontrados por hora.

Rendimiento de fase: porcentaje de defectos en un programa que fueron removidos en una fase dada.

Rendimiento de proceso: mide el porcentaje de defectos removidos antes de una fase dada.

6

Definición de requerimientos

6.1 ¿Qué son los requerimientos?

Se utiliza el documento SRS (que no es fundamental, pero es muy útil) para hacer una descripción clara y precisa del producto a crear. Los requerimientos proveen retroalimentación al cliente para garantizar el cumplimiento de las necesidades; el instructor puede actuar como cliente para asegurar que se esté logrando lo esperado.

En el proceso de requerimientos se deben plantear preguntas de parte de todos los integrantes, y que, para todas, debe existir alguna respuesta. Las adivinanzas también se recomiendan para garantizar que se entiende el problema del cliente, es decir, se valida si realmente se sabe lo que se quiere hacer.

Los requerimientos responden a dos criterios: declaración inicial de las necesidades del cliente y, explicar para qué se quiere hacer el proyecto. El instructor es la última palabra en la construcción de los requerimientos y así, del cumplimiento de estos criterios; tener acceso al usuario final otorga más seguridad.

En el ciclo 1, esta fase tarda una semana, en el ciclo 2 y 3, esta fase tarda media semana.

6.2 Por qué necesitamos los requerimientos

Los requerimientos se necesitan para identificar claramente las necesidades de los clientes. Cada uno revisa las necesidades del cliente. Para lograr entender estas necesidades se deben hacer discusiones en torno a preguntas planteadas, a fin de identificar cuáles de ellas se entienden, y con ello añadir información fundamental al proyecto.

Alcanzar el entendimiento común de lo que se va a construir; es esencial la participación de todos.

6.3 Cambios de los requerimientos

Como, hasta que el usuario prueba el sistema, es que sabe cómo va a cambiar su ambiente de acuerdo al producto que le proporcione el equipo de desarrollo, se recomienda un alto contacto con el cliente, a fin de aprender a detalle y con claridad lo que necesita.

Con el documento SRS se puede hacer más eficiente el ponerse de acuerdo con los usuarios, haciendo que los ingenieros expongan las funcionalidades ofrecidas por el producto, a los usuarios. Aun así, todos los cambios cuestan dinero y tiempo.

Pasos principales pasos para la obtención de requerimientos:

1. Evaluar la factibilidad del sistema.
2. Entender los problemas de organización.
3. Identificar los accionistas (stakeholdes) del sistema.
4. Registrar las fuentes de requerimientos.
5. Definir el ambiente operativo de sistema.
6. Evaluar los problemas de negocio.
7. Definir las restricciones de dominio.
8. Registrar la justificación de requerimientos.
9. Prototipar los requerimientos pobremente obtenidos.
10. Definir los escenarios de uso.
11. Definición del proceso operacional.

6.4 Especificación de requerimientos de software

Consiste en describir con propias palabras lo que se va a desarrollar y cómo se va a desenvolver el producto. Producir los requerimientos para permanentemente hacer el diseño. Se debe evitar el cómo y buscar el qué se va a hacer.

Temas principales a tratar:

- Requerimientos funcionales: entradas, salidas, cálculos y casos de uso.

-Interfaz externa de requerimientos: usuario, hardware, software, comunicación.

- Restricciones de diseño: formatos de archivo, lenguajes, estándares, compatibilidad.

- Atributos: disponibilidad, seguridad, mantenibilidad, conversión.

- Otros requerimientos: base de datos, instalación

Formato SRS (no debe por qué ser largo, debe ser específico)

1. Tabla de contenido
2. Introducción.
   - 2.1 Objetivo de SRS.
   - 2.2 Declaración del problema.
   - 2.3 Información del proyecto de equipo.
3. Declaración de requerimientos funcionales.
   - 3.1 Descripción de los requerimientos de la función del sistema.
   - 3.2 Requerimientos ciclo 1.
   - 3.3 Requerimientos ciclo 2.
   - 3.4 Estrategia de arriba-abajo.
4. Definición de reglas utilizadas en los requerimientos.
5. Requerimientos de interfaz externa.
6. Restricciones de diseño/implementación
7. Requerimientos especiales de sistema.
8. Referencias y fuentes de información.

Para hacer la rastreabilidad de requerimientos se puede utilizar una numeración por párrafos o líneas sobre el documento SRS (ver apéndice A).

6.5 Scripts de requerimientos TSP

Criterio de Entrada

- Se tiene una estrategia de desarrollo y un plan
- Se tiene el diseño conceptual.
- Todos los miembros del equipo han leído este capítulo y producido la declaración de necesidad del producto.

Revisión de declaración de necesidad

Se definen los requerimientos del producto. El gerente de desarrollo guía al equipo a lo largo de la revisión de la declaración de necesidad del producto y se asegura de que todas las preguntas e incertidumbres sean resueltas. En esta revisión, hay que concentrarse en las funciones se espera incluir durante el ciclo 1.

Aclaración de la Declaración de Necesidad

El gerente de desarrollo revisa la lista de preguntas del equipo para aclarar con el instructor.

Asignación de Tareas de Requerimientos

El gerente de desarrollo guía al equipo en la estructuración del SRS y los ayuda a identificar el trabajo requerido para producir el SRS. El equipo se compromete con fechas para completar el documento.

Documentación de Requerimientos

Al documentar el SRS, se hace una breve y clara declaración de lo que se pretende construir. Todos tienen que llegar a un acuerdo. Hay que asegurarse de registrar y rastrear la fuente de todas las declaraciones de requerimientos.

Plan de Prueba del Sistema

Se deben hacer pruebas bajo varias condiciones de error así como considerar usabilidad y problemas de recuperación.

Requerimientos e Inspección de Planeación de Prueba de Sistema

Inspección del SRS completo junto con el plan de prueba del sistema. En la inspección del SRS hay que identificar preguntas o desacuerdos y luego reunirse para la inspección, que debe ser liderada por el gerente de Calidad/Proceso que sigue el formato INS. El objetivo de la inspección es encontrar cualquier problema en el SRS así como inconsistencias antes de comenzar el trabajo de diseño.

Actualización de Requerimientos

Actualizar el SRS y el plan de prueba de sistema para corregir los problemas encontrados durante la inspección.

Revisión de usuario del SRS

El usuario final debe leer el SRS y estar de acuerdo en lo que describe.

Línea base de Requerimientos

El gerente de desarrollo hace un SRS oficial que ahora únicamente puede cambiarse utilizando el procedimiento de control de cambios (CCR).

Criterio de Salida

- SRS completo, actualizado e inspeccionado, plan de prueba del sistema bajo control de configuración.

- Toda la información de tiempo personal, defecto y de tamaño ingresada en formatos TSP y sistema de soporte.

- INS completo a partir de la inspección de requerimientos.

- Copias de todos los formatos, SRS y plan de prueba del sistema en el cuaderno del proyecto.

6.6 Resumen

En la fase de requerimientos, se produce la especificación de requisitos de software. El propósito es describir en propias palabras, aquellas funciones del producto a desarrollar. El SRS también debe proporcionar criterios claros e inequívocos para la evaluación del producto terminado, así como la orientación sobre la verificación de lo que se supone que debe hacer. Aunque el documento SRS es útil, no es lo más importante. Lo que realmente se quiere, es un acuerdo común de equipo de qué construir; es por esto que los requisitos de un proceso bien estructurado son importantes.

A medida que los usuarios aprenden más sobre el producto, ellos van a querer cambiar los requerimientos. Estos cambios a menudo son problemas serios, ya que cuestan tiempo y dinero.

Garantizar la trazabilidad funcional, el número de párrafos y las secciones del SRS, pueden ayudar a identificar el origen de cada declaración. Es necesario trazar el código desde el diseño y desde el diseño hasta el SRS y el estado de necesidad. Se debe etiquetar todos los requisitos de declaración de SRS como se explica en el apéndice A. Cuanto más sea posible, mantener todos los estados funcionales en oraciones separadas, breves y precisas.

7

Diseño con equipos

7.1 Principios de diseño

Se describe el proceso de diseño detallado en los formatos IMP1 e IMPn. DES1 y DESn menciona el proceso de diseño detallado.

7.2 Diseño en equipos

Hay la necesidad de definir la estructura total del sistema, y la forma de lograrlo, es trabajando todos juntos. Un diseño de alto nivel contiene interfaces, estructura lógica, funciones unitarias y generalidades del programa.

Un buen estudio de diseño puede proporcionarse gracias al desarrollo de prototipos. Si algunos integrantes del equipo son capaces de llevar a cabo el diseño, puede asignárseles la tarea, y uno o dos integrantes, encargarse del desarrollo del prototipo para posteriormente probarlo con los usuarios típicos del sistema. Este prototipo se asocia principalmente a la interfaz.

7.3 Estándares de diseño

Convenciones. Se debe definir el glosario del sistema, módulos, componentes, variables, archivos, cambios de nombre.

Formatos de interfaz. Incluye parámetros de variables, códigos de error, u otras condiciones.

Mensajes de sistema y error. Establece un estándar de formatos y mensajes entendibles. Ahorra el tiempo de desarrollo.

Estándar de defectos de TSP.

Hay que tener diseños precisos, y seguramente contar con los casos de uso nativos de PSP, buscando cómo es que el programa se va a utilizar, de modo que se identifique problemas de diseño y la usabilidad.

7.4 Diseño para la reutilización

Hacer un programa de reúso desde el principio del proyecto fomenta la producción de productos de alta calidad. Hay que definir funciones reutilizables autocontenidas, para así, contar con formas de comparar diseños con bajo acoplamiento.

Se debe garantizar el desarrollo de una buena documentación para evitar tener que mirar el código fuente. De igual manera, pruebas unitarias sobre código reusable permite visualizar la calidad de éstas.

Responder a las preguntas:

1. ¿El programa tiene las funciones correctas?
2. ¿La interfaz es adecuada?
3. ¿El rendimiento es el adecuado?
4. ¿El programa cumple con los estándares de calidad?

Si al responder a estas preguntas la mayoría de respuestas es “No”, entonces el reuso llevará más trabajo.

7.5 Diseño para la usabilidad

Consiste en la construcción de prototipos que son mostrados constantemente al cliente para identificar las interfaces.

7.6 Diseño para pruebas

Desarrollar código de pruebas permite probar componentes específicos del programa. Se realizan pruebas de caja negra y blanca. Las pruebas de caja blanca muestran los caminos que recorre la ejecución del programa, y las pruebas de caja negra, representa los caminos externos.

7.8 Scripts de diseño TSP

Criterio de Entrada

Haber completado la estrategia de desarrollo y el plan, haber documentado los estándares de diseño, y un SRS completo e inspeccionado.

Diseño de alto nivel

Se produce un diseño de alto nivel con los siguientes pasos: decidir la estructura general del producto, nombrar todos los componentes del producto, asignar funciones del producto a los componentes, producir las especificaciones externas del componente, asignar funciones de casos de uso a estos componentes, e identificar las tareas de diseño a completar. El gerente de desarrollo es quien guía al equipo en estos pasos.

Estándares de Diseño

Se producen los estándares de diseño y el glosario de nombres con ayuda del gerente de calidad/proceso.

Estructura general del producto

Primero se produce un diseño conceptual de alto nivel. Esto identifica las partes del componente, nombra sus funciones generales y decide cómo relacionarla. Existen muchas formas de hacer el diseño: diagramas de flujo, casos de uso u otras metodologías.

Para asignar las funciones del producto a los diferentes componentes, utilizar una tabla de rastreabilidad. Utilizar el formato STRAT para enlistar cada componente con sus funciones. Agrupar las funciones por componente, junto con las referencias a los párrafos de SRS para cada función. Luego etiquetar los componentes. La ventaja de esta tabla es que provee una forma rápida de ver lo que hace cada componente e identificar funciones omitidas o redundantes.

Asignación de Tareas de Diseño

El gerente de desarrollo guía al equipo en la creación del documento SDS. El equipo asigna estas secciones a los diferentes integrantes y se comprometen con fechas de entrega.

Especificación de Diseño

Mientras se produce el SDS, hay que definir las interfaces externas y especificaciones funcionales para cada componente. Como parte del SDS, hay que generar un conjunto completo de escenarios que cubran las funciones externas de todos los componentes.

El paso final en el diseño de alto nivel es producir varios documentos de diseño que especifiquen la lógica y estructura del programa principal del sistema. Después de producir el SDS, hay que revisar el trabajo y arreglar los problemas que se encuentren.

Cuando se termine de revisar el SDS personal, se le da al gerente de desarrollo y lo incorpora en el SDS general final.

Plan de Prueba de Integración

Es importante producirlo mientras se hacen las especificaciones de diseño. La razón es que las pruebas de integración se supone deben revisar y verificar todas las interfaces a lo largo de los componentes del sistema.

Inspección de diseño

Todo el equipo debe participar en la inspección, encontrando el número máximo de defectos de diseño y asegurándose de que todos los integrantes entienden el diseño.

Actualización del diseño

Después de la inspección de diseño, se deben arreglar los errores identificados en tu parte del SDS y/o plan de prueba de integración y hacer que uno o dos ingenieros revisen las correcciones si se necesita. Luego se le da el SDS corregido y revisado al gerente de desarrollo, que produce y distribuye los documentos completos.

Línea de base del Diseño

El gerente de soporte hace el SDS oficial y le da una copia a cada integrante del grupo.

Criterio de Salida

1. SDS completo, inspeccionado y corregido.
2. Plan de prueba de integración completo, inspeccionado y corregido.
3. Estándares de diseño y glosario de nombre.
4. INS completo.
5. Formatos SUMP y SUMQ actualizados.
6. Información de tiempo, tamaño y defectos de los ingenieros ingresada a la herramienta.
7. Copias de los materiales de diseño en el cuaderno de trabajo.

7.9 Resumen

El objetivo principal del proceso de diseño es asegurar que los ingenieros producen minuciosos diseños, de alta calidad. En el diseño de los equipos, primero producir una estructura global de diseño y luego dividir el producto en sus componentes principales. Los miembros del equipo, posteriormente diseñan por separado y ofrecen sus diseños para el gerente de desarrollo, que las combina en una única especificación de diseño de sistema. También es importante para producir e inspeccionar el plan de pruebas de integración, al mismo tiempo que se produce e inspecciona el SDS.

El diseño es el proceso creativo de decidir cómo se va a construir un producto. Su objetivo es proporcionar una base precisa, completa y de alta calidad para su implementación. Un diseño completo también debe definir cómo las partes del producto interactúan, cómo van a ser ensambladas, para producir el sistema acabado y terminado. A continuación, el diseño de alto nivel se hace una tarea completa y precisa, los ingenieros deben implementar componentes sin más orientación que el diseño. Esto requiere que los diseñadores utilizan un diseño preciso y con notaciones no ambiguas.

El proceso de TSP para producir el diseño de alto nivel, consta de cuatro pasos: decidir sobre la estructura general del producto, las funciones asignadas a los componentes del producto, la producción de las especificaciones externas de los componentes, y decidir cuáles componentes y funciones desarrollar en cada ciclo.

Uno de los problemas en el diseño de grandes productos de software, es que la estructura básica debe definirse antes de cualquier otra cosa especificar. Hasta que esta estructura se resuelve, se vuelve difícil dividir el trabajo entre los miembros del equipo. Esto hace que sea difícil de utilizar todas las energías y capacidades plenas de los miembros del equipo. Esto proporciona una excelente oportunidad para llevar a cabo los estudios de diseño, producción de interfaces de prototipo, desarrollo de normas de equipo, o desarrollar planes y especificaciones de reutilización.

Se deben establecer normas que tratan de convenciones de nomenclatura, formatos de interfaz, mensajes de sistema y de error, y las normas de tamaño. Tal vez la norma de diseño más importante se refiere a la representación de diseño. La reutilización es una forma potencialmente poderosa para mejorar la productividad de equipo. Las cuestiones clave son la reutilización de interfaces estándar, convenciones de llamada, estándares de documentación, calidad de productos y soporte de aplicaciones.

8

Implementación de producto

8.1 Diseño de criterios de complexión

Repetir los scripts DES1 y DES desde que no se alcance la especificación de alto nivel. Se pueden implementar algunos elementos del producto, mientras se diseñan otros; aunque es un proceso es irregular.

Todas las especificaciones de alto nivel deben estar completas e inspeccionadas para proceder con la implementación, sin embargo, el dividir el proyecto en varios componentes permite trabajarlos en forma independiente de tal manera que, se puede implementar parte del producto total.

8.2 Estándares de implementación

No debe tomar gran cantidad de tiempo su definición. Primero se debe estar de acuerdo de los estándares a construir, asignando a uno o dos miembros la construcción de borradores, para que sean discutidos por todos los miembros del equipo. El gerente de calidad/proceso es quién dirige esto.

Revisión de estándares, nombres de interfaz, estándares de tamaño, estándares de codificación, estándares de mensaje, estándares de defectos y estándares de prevención de defectos. Si un estándar propuesto se visualiza útil, se recomienda trabajarlo, sin que esto implique que el estándar sea perfecto; trabajar con un estándar que se ajuste será más sencillo, y cuyas mejoras será mejor registrarlas para futuros proyectos.

Revisar el nombre, interfaz y estándares desarrollados durante la fase de diseño para asegurar que son apropiados. Revisar que la lista de rutinas reutilizables esté completa y que todos los miembros del equipo la estén utilizando. Hay que revisar el glosario de nombres para que todos los estén utilizando. Revisar que las variables compartidas, nombres de archivos y parámetros son consistentes.

Las inspecciones de código serán más fáciles y efectivas al usar todos, las mismas notaciones. Hacer también buenas prácticas en comentarios. Si se usan las mismas convenciones, seguramente serán elementos a reusar a futuro.

Cada equipo tiene un estándar de conteo (que se define en la fase de diseño), como los LOC, documentos SRS y SDS. Se recomienda el conteo de páginas para definir el tamaño de estos documentos. Documentos: Requerimientos SRS, Diseño de alto nive SDS, diseño detallado, pantallas y reportes, bases de datos, scripts de prueba, mensajes y materiales de prueba. Para

requerimientos se puede usar el conteo de páginas de texto o párrafos. Para el diseño detallado se suele hacer conteo del número de líneas de texto, párrafos o número de casos de uso.

Estimar y rastrear el trabajo es lo que se está buscando. Para trabajos pequeños se recomienda sólo hacer un conteo de tiempo, no de tamaño. Asignar categorías de tamaño puede ayudar a determinar si un módulo conlleva más o menos tiempo.

Categorizar los defectos es una tarea difícil y requiere ser puntual para especificar claramente sin con ello tener una lista enorme de posibles defectos. Saber diferenciar entre tipos de defecto y causas de defecto (desconocimiento del lenguaje es una causa).

Categorías recomendadas para prevención de defectos:

1. Educación: conocimiento del lenguaje, ambiente o aplicación.
2. Comunicación: Arreglar el proceso.
3. Transcripción: Usar mejores herramientas.
4. Vista general: Refinar el proceso y utilizar mejores métodos.

Enfoque de prevención de defectos:

1. Elige los tipos de defectos que parezcan que causan el mayor problema.
2. Examina el número de defectos para identificar causas particulares de defectos y decidir a cuáles atender.
3. Cuando veas un defecto y veas que puedes prevenirlo, haz un cambio en el proceso para prevenirlo.
4. Comienza a ver por la siguiente categoría de defecto para manejarlo de la misma manera.

Buscar formas de cambiar el proceso para prevenir el defecto es la clave. Rastrear el rendimiento para ver si ha dado frutos, y notar si ha mejorado, sino, tratar de descubrir qué fue y registrarlo.

8.3 Estrategia de implementación

Debe estar conforme a la estrategia de diseño. En el diseño se trabaja de lo grande a lo pequeño (o específico), en la implementación es al revés. Se crean las funciones para asegurar que se hace lo que el programa espera. Se hacen los objetos atómicos para ir a un siguiente nivel, estando seguro de que cumplen con las especificaciones.

Utilizar los encabezados de comentarios para cada programa fuente, para que usuarios potenciales puedan observar rápidamente lo que hace el programa. Adicionalmente se recomienda colocar advertencias, con variables en mayúscula.

Cuando se hagan reuniones, o a diario informar, se debe preguntar qué y quién tiene elementos reutilizables, de modo que el líder de soporte los añada a la lista de elementos de reúso.

Antes de probar hay que revisar que el código cumple con las especificaciones.

8.4 Revisiones e inspecciones

Muchos defectos son aleatorios especialmente asociados al problema de tecleo. Una o dos semanas se utilizan para la detección de defectos. Es útil tener una lista de inspecciones para verificar y actualizar la lista con base en lo obtenido. El apéndice C trata éste tema.

8.5 Scripts de implementación

Se utiliza IMP1 e IMPn.

Criterio de entrada

1. Una estrategia y plan de desarrollo completos.
2. Especificaciones de SRS y SDS completas, revisadas y actualizadas.
3. Un estándar de codificación bien definido y documentado.
4. Copias disponibles de la lista de especificaciones de las rutinas funcionales, glosarios y todos los estándares adoptados por el equipo.

Implementación de la planeación

Como hay ingenieros mejores que otros, se reitera el enfoque de: algunos orientados al diseño y otros orientados a la implementación. El primer paso es revisar los trabajos a realizar y que dichas tareas estén distribuidas en varios miembros del equipo. Un enfoque es tener algunos ingenieros especializados en diseño y otros en implementación.

Después de asignar las tareas, cada ingeniero planea su trabajo de implementación; para estas tareas de pocas horas, un tiempo estimado es adecuado. Deben hacer un plan con los formatos SUMP y SUMQ, y para tareas no sustanciales utilizar el formato SUMTASK. Asumiendo que se tenga un estimado de LOG para cada módulo y has obtenido información de productividad personal, el trabajo de planeación no debería tomar mucho tiempo.

El siguiente paso es conducir una revisión personal de diseño que consta de revisar los Loops con tablas de rastreabilidad o máquinas de estado.

Después de fijar los problemas encontrados en la revisión hay que desarrollar cualquier código especial de prueba de unidad. Como el mayor error se encuentra en problemas de diseño, es necesario hacer el desarrollo de pruebas antes de la inspección de diseño detallado. Los planes de prueba revisan los caminos lógicos, Loops, variables, parámetros, límites y aseguramiento de la buena ejecución del programa.

Después del desarrollo de prueba, se lideran a uno o dos integrantes para el diseño detallado. Se ingresa esta información de inspección en el formato INS y los defectos más importantes en el LOGD. Hay que dar copia del INS al líder de calidad/proceso y al líder de equipo.

Después de la inspección de diseño se debe programar el módulo. Antes de que personalmente revises, mirar la historia de defectos y deducir cuántos defectos se puedan inyectar durante la programación. Con las propias tasas de defectos inyectados (mínimos y máximos) se puede hacer un estimado bueno. Es buena idea fijar un objetivo de tiempo; 50%

del 78% que se gasta. Una simple lectura del programa fuente puede tomar más del tiempo de crear el código. Hay que revisar la consistencia, igualdades, punteros, secuencias de llamado, etc; como hay mucho que ver, hay que garantizar el checklist aun cuando se tarde más del tiempo estimado. Otra buena técnica es que otro ingeniero revise tu programa (en caso de que uno no pueda encontrar más defectos) para que así pueda compilar el programa.

Después de la compilación compara el diseño, revisión de diseño, código y tiempos de revisión de código con el plan de calidad del equipo. Hay que revisar los niveles de defecto y tasas de defecto, si la información indica que hay problemas hay que revisar los resultados con el líder de calidad/proceso. Una forma de decir que se está cumpliendo (tabla 5.8), tiempo invertido en el diseño debe ser mayor que el tiempo de codificación; el tiempo invertido en la revisión de diseño debe ser mayor al 50% del tiempo de diseño.

Si la calidad del programa es pobre y si el rendimiento ha sido menor al 70%, el líder de calidad proceso que lleve a cabo la inspección de código. Entre más pobre sea la calidad del programa, y menor el rendimiento, mayores revisores se van a necesitar (apéndice C). Hay que utilizar clientes que no conocen el programa sin mencionar los defectos y de acuerdo a lo que ellos identificaron tomar acciones descritas en el apéndice C.

El siguiente paso son las pruebas de unidad. Se utilizan los materiales y el plan de prueba.

Siguiendo la inspección del código y las pruebas de unidad, el líder de calidad/proceso debe revisar la información del componente, para saber si es suficiente para incluirlo en el sistema base. El criterio para ello se usa el plan de calidad (formato SUMQ tabla 8.3 muestra que se cubren niveles de defectos, proporciones de defecto, proporciones de tiempo), el equipo entra en discusión y decidir qué hacer. Las opciones son poner el componente en el sistema base de todas formas para hacer más diseño, revisiones de código e inspecciones, o poner a un ingeniero retrabajando el componente antes de la integración; esto ahorra al equipo muchos días.

Se debe hacer una nueva revisión sobre el programa completo, reescribiendo cualquier código cuestionable o torpe. Hacer análisis de máquina de estado. También se pueden hacer mismos checkeos en forma individual para que cada uno descubra de forma independiente los defectos.

Finalmente hay que darle al líder de soporte la revisión de calidad.

Criterios de salida

1. Componentes completos, revisados e inspeccionados.
2. Los componentes ingresados en el sistema de configuración de manejo.
3. Formatos INS LOG completos para las revisiones de diseño, código e inspecciones.
4. Formatos SUMP, SUMQ y SUMS actualizados para el sistema y todos sus componentes.
5. Planes de prueba de unidad, soporte. Tamaño, tiempo e información de defectos.
6. Cuaderno actualizado.

8.6 Resumen

Los principales pasos en el proceso de implementación es la planeación de la implementación, diseño detallado, especificación detallada, revisión de código, prueba de unidad, revisión de calidad de componentes y liberación de componentes. Los estándares de implementación también se desarrollan para añadir y extender los estándares definidos en la fase de diseño.

La estrategia de implementación debe ser consistente con la estrategia de diseño, al hacer esto hay que considerar las pruebas, reúso e inspecciones. Para hacer las pruebas e inspecciones más eficientes la implementación debe comenzar con los detalles hasta lo global. Primero módulos más bajos y llegar a estructuras de alto nivel.

Tanto en diseño como en implementación, hay que considerar el reúso. En la paso de implementación, el líder del equipo guía al equipo en la asignación de tareas de implementación, luego cada ingeniero producen el diseño detallado para cada módulo y conducen una revisión de diseño personal, también desarrollan cualquier material de prueba requerido. Luego el líder de calidad/proceso lidera a los ingenieros a través de las inspecciones de diseño detallado, en la que cada ingeniero debe por lo menos completar una tabla de rastreo, tabla de ejecución o análisis de máquina de estado de cada loop o estructura de máquina de estado. Siguiendo la inspección de diseño, los ingenieros codifican los módulos y personalmente los revisan.

Ahora los ingenieros revisan la información de calidad con el líder de calidad/proceso, y utilizan esta información para guiar la inspección de código. Entre más pobre sea la calidad del programa se necesitarán más revisores. Hay que seguir el script INS para la inspección de código. En estas inspecciones varios defectos serán encontrados por uno o varios ingenieros; entre menos ingenieros los encuentren, serán menos los defectos reales que se encuentren en código. Siguiendo INS, el equipo estima el número probable de defectos que quedan y el rendimiento de inspección. Con el rendimiento de inspección uno o más ingenieros deberían hacer la inspección. Finalmente se hace la prueba de unidad, luego el líder de calidad/proceso valida que el componente vaya o no en el sistema base. Entonces los criterios se hacen por el plan de calidad, si el componente tiene problemas de calidad, las alternativas son ponerla en el base o poner otros ingenieros a retrabajarlo antes de la integración.

Después de pasar la revisión de calidad, los componentes completos se dan al líder de soporte para ingresarlo en el sistema base. Al final de esta fase, el equipo debe haber completado, revisado e inspeccionado los componentes que están bajo el manejo de configuración. Deben haber completado los formatos INS y LOGD para las inspecciones de diseño, código y reinspecciones. Formatos SUMP, SUMS Y SUMQ actualizados para el sistema y todos sus componentes. La información de proceso debe estar ingresada en la herramienta y el cuaderno de proyecto actualizado.

9

Integración y prueba de sistema

9.1 Principios para hacer pruebas

Evaluar el producto, no arreglarlo. Generalmente un producto de baja calidad demuestra problemas de calidad en las pruebas.

9.2 La estrategia para hacer pruebas en TSP

Durante las pruebas se verifica que los productos sean de alta calidad.

1. Utilizando partes desarrolladas y probadas por unidad hay que construir el sistema
2. Hacer una prueba de integración de sistema para verificar que todo está presente y que funcionan juntas.
3. Hacer pruebas al sistema de producto para validar que hace lo que los requerimientos requieren. Identificar módulos baja calidad y regresarlos al gerente de calidad/proceso para evaluación y limpieza.
4. Identificar componentes de baja calidad que sean problemáticos.

Construir, hacer prueba de integración y prueba de sistema.

Junta varias partes del sistema para probarlo (construcción de sistema). La prueba de integración identifica si las interfaces son compatibles trabajan juntos. Se verifican las funciones del sistema contra los requerimientos. Se solicita también pruebas de regresión.

9.3 Estrategia de construcción e integración

Asegurarse de que todas las partes están presentes para complementar un sistema de trabajo y proveerlo a la prueba de integración y de sistema. Se define el enfoque de la prueba de integración, que básicamente revisa que todos los componentes estén presentes y todos sus llamados funcionen. No se deben probar las funciones de componentes, eso se hace en la prueba de sistema. La estrategia de integración depende de la estrategia de desarrollo y de todo el trabajo.

Estrategia del Big Bang: poner todas las partes juntas haber si funcionan juntas o no. Requiere menor tiempo de pruebas pero pocas veces es efectiva, aunque sirve para sistemas de baja calidad.

Uno a la vez. Añadir partes a la vez. Te guía a las causas de los problemas, pero posiblemente requiera mayor trabajo de desarrollo de prueba.

Cluster. Añadir partes en cluster. No se puede sólo tomar las partes que sea, únicamente tomar y unir componentes relacionados.

Sistema plano. Integrar todas las partes de más alto nivel, y luego ir ahondando de manera sucesiva en las capas del sistema en paralelo. Se puede añadir un componente a la vez o los que uno quiera. Puede detectar problemas del sistema de manera temprana. Provee la flexibilidad de construir rápidamente un esqueleto total de todo el sistema. Su problema principal, es que requiere grandes números de stock o proveer números falsos.

El mejor enfoque es considerar todas las opciones y aplicar el que mejor se adapte al proyecto en particular.

9.4 Estrategia de prueba de sistema

Prueba de sistema:

¿El sistema desempeña apropiadamente las funciones que debe desempeñar?

¿El sistema cumple sus metas de calidad?

¿El sistema opera apropiadamente bajo condiciones normales?

¿El sistema opera de manera apropiada bajo condiciones adversas?

Hay que revisar el número de áreas clave. Por ejemplo, puede el sistema ser instalado? Inicia de manera correcta? Desempeña todas las funciones establecidas en los requerimientos? Se recuperará de fallas de hardware o software? Su rendimiento es adecuado? Evaluar el tiempo de respuesta sería para este caso.

Identificar los objetivos principales del sistema porque rara vez es suficiente alcanzar a recubrir todas las pruebas; un plan de prueba de sistema. Luego de ello divisar una estrategia para asegurar que todos los objetivos se cumplan.

Estrategias alternativas. Puedes probar los objetivos en serie (con un plan que captura lo más importante del proyecto), probando cada una de las funciones, operaciones bajo condiciones de estrés luego la usabilidad y luego el rendimiento. Este enfoque deja poco o casi nada de tiempo para pruebas reales de sistema.

Pruebas comprensivas. Con un plan de prueba cuidadosamente diseñado, se puede evaluar diversas características de producto.

Otra estrategia es de áreas funcionales. Cubre todos los aspectos de un área antes de pasar a otra. Elimina la duplicación de pruebas. El problema es que no se enfoca al comportamiento general del sistema.

Tercera estrategia: combinación de las anteriores. Funciones de bajo nivel para evaluar el comportamiento normal y anormal, luego se sube un nivel, y se prueban agregados funcionales para asegurar que trabajan juntos, y nuevamente se verifica el comportamiento

normal y anormal, se sube de nivel y así sucesivamente. Su desventaja es el tiempo que requiere tomar de forma progresiva las combinaciones funcionales para un sistema grande.

Cuarta estrategia: enfoque de reversa. Se parte con funciones de más alto nivel y se trabaja hacia abajo, haciendo pruebas de estrés y normales. Puedes hacer una prueba extensiva de varias combinaciones funcionales. Cubre problemas de sistema rápido, sólo funciona con productos de calidad (?). Con cada identificación de problemas de calidad, regresar, hablar con el gerente de calidad para que lo reemplace o repare.

9.5 Planeación de pruebas

Se hace en diversos lugares. En TSP, al hacer la prueba conceptual, se necesitan planes de pruebas para construcción e integración. Plan de prueba que deseas utilizar, te guía mejor.

Se debería tener a este punto (al final) una vista de todos los pasos de prueba a desarrollar, los materiales requeridos para cada prueba, los resultados que las pruebas producen, un estimado para el tiempo de corrida sin defectos, los defectos a encontrar, el tiempo para cada prueba, un estimado del trabajo requerido para desarrollar cada ítem en un plan de prueba. Se va a necesitar una lista de materiales de soporte requeridos y qué pruebas van a soportar, los objetivos para cada prueba, cuán grandes se espera que sean los materiales, quién va a desarrollar cada uno de los ítems de soporte de prueba y cuándo las pruebas de desarrollo se van a completar.

9.6 Rastreo y prueba de medida

Efectividad de prueba: cuántos defectos cada prueba descubre como una función de su tiempo de corrida. Se puede utilizar el número de defectos/hora como un modo de mérito en la selección de pruebas a incluir.

Además de LOGD y LOGT

¿Cuánto te tomó correr esta prueba?

¿Cuántos defectos encontraste?

¿Requiere intervención manual?

Registro de prueba!

Fecha de corrida de prueba, el nombre de la persona que la ejecuta, nombre y/o número de corrida de prueba, producto y configuración a probar, tiempo de inicio y fin de prueba, el número de defectos encontrados con las referencias LOGD y resultados de prueba. La forma más simple es el registro cronológico.

Módulos propensos a defecto. Defectos encontrados por clientes y por ingenieros. A mayores defectos que encuentres en prueba, mayor serán los que no encuentres.

Esto sugiere que se puede utilizar la información de prueba para evaluar el riesgo que el sistema pueda tener. Hay que sortear la información de defecto de módulo para encontrar qué módulos tienen los mayores defectos en cada prueba. Aún después de la prueba tienen defectos, el hecho de registrar, permite identificar más fácil el problema.

Hay dos formas de examinar la información dentro del módulo. SUMP da los valores de defectos removidos en cada fase, o SUMQ para las medidas de calidad de un módulo individual.

Rastreo de defectos. ¿Por qué y cómo hacer que estos defectos no vuelvan a pasar? ¿Cómo se pueden corregir estos defectos ahora?

9.7 Documentación

Parte del equipo construye la documentación de usuario mientras la otra parte se encarga de las pruebas. Con TSP se centra en la documentación básica de usuario. Todo esto depende de la calidad y contenido funcional del producto. La cantidad de gente para cada uno de los dos grupos depende el grado de calidad y contenido funcional del producto. Conforme pasan ciclos se recomienda aumentar el número de personas para desarrollar la documentación.

En muchas formas la documentación es más importante que el código del programa. Si se hace primero la documentación, se tendrá que hacer muchos cambios y al revés, el trabajo toma más tiempo. Como parte de la prueba de sistema, es buena idea revisar la documentación de usuario.

La guía de usuario debe tener una tabla de contenido y organizado y orientado a las necesidades del lector y no al producto. Debe incluir el "cómo comenzar", hacerlo fácil, explicando lo que el usuario puede hacer con su producto.

Hacer un glosario para palabras no estándar, presentar procedimientos de solución de problemas, incluir excepciones de mensajes de error, y procedimientos de recuperación, índice de claves. La documentación debe comenzar por un bosquejo de alto nivel.

Al revisar el documento hay que checkar:

1. Distribución
2. Terminología
3. Contenido del documento ¿Cubre todo el material requerido?
4. Decisión. Los métodos y procedimientos realmente funcionan?
5. El documento es fácil de leer?
6. Comprensión. Los lectores pueden entender lo que está escrito?

9.8 Scripts de prueba TSP

Criterio de entrada:

Se han completado e inspeccionado los SRS y SDS.

Se han implementado e inspeccionado y probado los componentes de la unidad bajo control de configuración

Se tiene la versión anterior del producto configurada y controlada

Desarrollo de pruebas:

1. Prueba de construcción total: Primer paso de la prueba de integración revisa que el sistema se haya construido y que todos los componentes planeados están incluidos. Es una revisión completa que verifica que cada componente esté presente.
2. Procedimientos de integración para probar interfaces. A partir de la prueba de completitud se conocen los componentes están presentes y ahora se asegura que se ejecuta, se empatan apropiadamente y se pueden llamar entre interfaces para suplir funciones apropiadamente.
3. Materiales de prueba de integración. Prueba de interfaces manuales y automáticas.
4. Materiales de prueba de sistema. Materiales que prueban el software mediante condiciones normales. Se visualiza cómo se va a utilizar el sistema. Deben considerar usabilidad, rendimiento y corregir cálculos matemáticos.
5. Declaración clara de los resultados esperados en cada prueba

Construcción

1. Revisar todos los componentes para asegurarse que están a la mano y cumplen con los requerimientos de dependencia (como bases de datos).
2. Revisar los ítems abastecidos, garantizando que no falte ninguna pieza.
3. Revisar la construcción propuesta para ver si hay consistencia y totalidad.
4. Compilar y enlazar componentes del sistema.

Llenar LOGD y LOGT para cada defecto encontrado y tarea realizada respectivamente. Dependiendo de los defectos, las acciones a tomar las decide el gerente de calidad.

Integración:

1. Revisar la totalidad del producto construido. Correr una prueba de totalidad
2. Correr pruebas de integración.
3. Registrar el plan de prueba en el LOGTEST. Nombre, fecha, tiempo, sistema a prueba, etc.
4. Si se encuentran defectos, el dueño del producto registra cada defecto. El gerente de calidad determina si se continúan con las pruebas. Hay que solucionar los problemas en forma paralela.

Prueba de sistema. Registro de defectos. Hay que hacer que el gerente de calidad proceso determine si toca continuar el proceso o cancelar.

Pruebas de regresión: hacer corridas sobre los ciclos de desarrollo.

Criterio de salida:

1. Una versión de producto completa, íntegra, con pruebas de sistema y pruebas de regresión

2. Registro de prueba completo en el registro de prueba
3. Completar las entradas LOGD y un análisis de defecto para cada defecto encontrado.
4. Documentación de usuario completa realizada
5. Formato SUMP y SUMQ actualizados por el sistema
6. Copia para la carpeta del proyecto

9.9 Resumen

Hay que identificar qué partes del sistema son propensas a defecto. Estos módulos deben ser sacados de las pruebas. Prueba e integración ensamblan el sistema, asegurando que las interfaces se unen correctamente y se encuentren todos los módulos. Deben garantizar que el sistema:

1. El sistema desempeña las funciones que debe desempeñar
2. El sistema cumple sus metas de calidad
3. El sistema operará apropiadamente bajo condiciones normales
4. El sistema operará apropiadamente bajo condiciones adversas
5. El sistema comienza de manera apropiada
6. El sistema desempeña todas las funciones dadas en los requerimientos
7. El sistema se recuperará de fallas de hardware o energía
8. El rendimiento del sistema es el adecuado
9. El sistema es utilizable

En la planeación de prueba se ordenan las pruebas y se ejecutan las pruebas planeadas, registrando los resultados y el tamaño de los módulos evaluados. Las necesidades del usuario es el enfoque de la documentación, a cargo de parte del equipo.

10

Postmortem

10.1 ¿Por qué se necesita postmortem?

El software es el núcleo de casi todos los productos y servicios de las industrias; es importante casi para cualquier industria.

10.2 Lo que un postmortem puede hacer por ti

Forma estructurada de aprender y mejorar. El trabajar más no es sinónimo de una mejora consistente; deben hacerse cambios de trabajo. En esta fase se visualizan los cambios de un ciclo a otro, determinar problemas, causas y divisas procesos de prevención. Hay que decidir dónde y cómo mejorar los cambios en los procesos personales.

10.3 Propuesta de mejora de proceso

La llave es enfocarse en cambios pequeños. Cada mejora ayuda un poco, con el tiempo se verá un cambio total significativo y mejorar el proceso personal. Cualquier idea de mejora es mejor anotarla en PIP para que no se olvide.

10.4 Scripts de postmortem

PM1 y PMn se muestran en las tablas 10.1 y 10.2; el 1 necesita al instructor. Se utiliza para curbir cualquier problema o lección para futuros postmortem, se evalúa la efectividad en discusiones de anteriores postmortem para decidir qué dejar y qué cambiar. Hay que concentrarse en temas relevantes o hecho de información para los reportes.

Criterio de entrada:

1. El equipo ha completado y probado el producto
2. Los ingenieros han reunido toda la información y formatos requeridos.

Revisión de los formatos del ciclo: SUMP y SUMQ. Se revisa la información de lo que los integrantes y equipo hicieron, identificar qué funcionó y qué no, comparar el rendimiento de metas y planes, identificar las áreas de problema y necesidades de mejorar, divisar mejoras de proceso (PIPs).

Revisión de calidad evaluando. Si tuvo uno o más defectos de sistema, evaluar qué mejorar en el proceso:

Hay que responder:

¿Cómo se compara el rendimiento real con el plan?

¿Qué lecciones puedes aprender de esta experiencia?

¿Deberías usar criterios personales y de equipo en el futuro?

¿En dónde ves oportunidades de mejora y por qué?

¿Dónde tuviste problemas que pueden corregirse la siguiente ocasión?

PM, MEJORAS:

¿En dónde es que el equipo hizo las cosas bien?

¿Cómo es que su rendimiento se compara con otros equipos?

¿Qué metas definir para futuros ciclos/proyectos?

¿Dónde deberían modificar los procesos que utilizaron?

Preparar todos los PIPs para sugerencias de mejora, evaluando el rendimiento, checando las metas, identificar áreas de prioridad para la mejora.

10.5 Evaluaciones de equipo

El líder del equipo examina cada rol.

1. ¿Dónde funcionó?

2. ¿En dónde están los problemas?

3. ¿Qué metas a mejorar para el desarrollo?

Facultad del instructor:

¿En qué podría ser más útil?

¿Problemas de instalación o herramientas?

Concentrarse en experiencias constructivas.

10.6 Preparar reporte de ciclo

Describe lo que se produjo, los proceso que se utilizaron y los roles llevados a cabo. Lo que se trabajó, lo que funcionó, lo que no, responsabilidades del equipo, como rol. Reporte breve, actual y enfatizar las lecciones aprendidas. Comparar el rendimiento con el ciclo anterior y destacar cualquier tendencia.

Formato del reporte de ciclo:

Tabla de contenido

Resumen

Reportes de rol

Líder de equipo

Desarrollo

Planeación

Proceso

Calidad

Soporte

Reportes de ingeniero

10.1 Reportes de rol

¿Cómo evaluar tu rendimiento de rol? ¿Cómo se desempeñó el equipo con el rol respectivo? Lo que funcionó y no y cómo mejorarse el rol para la siguiente ocasión. Es una guía para que la siguiente persona desempeñe mejor el rol.

Líder de equipo: Rendimiento del equipo desde el liderazgo, elementos de compromiso, áreas donde hubiera sido útil un instructor, prácticas que funcionaron y no.

Genere de desarrollo. Compara el contenido del procuto con los requerimientos y evalúa la efectividad, qué elementos fueron efectivos y que no, qué estrategia tomar a futuro. Describe los pasos de de diseño e implementación que se tomaron para la usabilidad, seguridad, rendimiento, compatbilidad o instalación.

Gerente de planeación. Describe cómo se comparó el rendimiento del equipo con el plan, las tendencias de valor ganado, valor ganado del equipo y reporte. Se comparan los ciclos anteriores y con otros proyectos.

Gerente de calidad/proceso. Utiliza información de calidad real para describir el rendimiento del equipo en términos de calidad. Ciclos de desarrollo completos. Se mide la disciplina.

Se debe incluir un resumen de todos los PIPs y cómo se manejaron. Se requiere una evaluación del rendimiento de inspección del equipo y de cada ingeniero.

El gerente de soporte describe las instalaciones de soporte y anota cualquier problema a mejor. Se comenta sobre la configuración, procedimientos de control de cambios, cómo se trabajó. La información debe incluir cambios de información de actividad, cambios tardíos, información cómo debo manejar mejor los cambios.

Gerente de soporte -> estrategia de reúso, porcentaje de reuso. Sugiere cómo mejorar las estrategias de reuso.

10.1 Reportes de ingeniero

Se debe reportar rendimiento personal incluida la cantidad de trabajo, cómo trabajar mejor a futuro.

10.1 Traduciendo el reporte

Definición de compromisos. El líder de equipo distribuye, el gerente de calidad proceso dirige el reporte borrador, luego los ingenieros hacen correcciones.

10.1 Evaluaciones de rol

Cada miembro del equipo prepara una copia personal de PEER, dando el punto de vista del rendimiento del equipo (porcentual).

10.1 Sugerencias de evaluación de rol

Actuar como si los comentarios fueran a la persona que llevaba el rol. Cada evaluación se hará pública. Tener cuidado de que las evaluaciones no avergüencen a nadie. Las evaluaciones son para ayuidar a la siguiente persona con el rol.

Criterio de salida:

1. El equipo ha producido un producto de alta calidad con la documentación requerida
2. El producto completo está bajo control de configuración
3. La información de proceso ha sido evaluada y los PIPs han sido sometidos y completos.
4. Las evaluaciones de rol están completas y sometidas.
5. El reporte de ciclo de desarrollo está completo y con copia para todos
6. Todos los formatos TSP se han completado.
7. Cuaderno de proyecto actualizado con todo

10.1 Resumen

El postmortem provee una forma estructurada de mejorar el proceso personal y de equipo. Se empieza con el líder para especificar lo que funcionó, lo que no y se compara lo planeado y lo real. Se identifica lo que se trabajó, qué problemas iba. Se revisa del líder a los demás. Concentrarse en áreas de mejora.

Mantener el reporte breve y actual. Hay que comparar el rendimiento con anteriores ciclos de desarrollo y marcar tendencias. Se tiene la opción de hacer comentarios adicionales del tema pero que sean de forma constructiva

ANEXO 5: Pruebas de software

1. Pruebas funcionales – (Caja blanca y gris)

1.1 Transición de estados

1.1.1 Transición de fases

La tabla 1 muestra los estados posibles para la implementación de la metodología TSP, lo que describe los estados desde la aparición del proyecto hasta su desarrollo fase a fase dentro de cada ciclo. El estado final de una fase es el estado inicial de la siguiente fase, y para pasar de una fase a otra, deben satisfacerse los criterios de salida específicos de la fase en la que el proyecto se encuentra actualmente. El resultado esperado evidencia el paso a la siguiente fase/ciclo, siendo la última columna la que evidencia que el logro o no de la transición, obteniendo un total del 100% de éxito en las salidas esperadas.

Tabla 1. Resultados de prueba de transición de estados.

Estado Inicial	Estado Final	Entradas	Resultado Esperado	Resultado Obtenido
Proyecto inexistente	Proyecto creado en fase de Lanzamiento y ciclo 1.	Nombre del proyecto, descripción, fecha de inicio, fecha estimada de finalización, instructor, criterios de proyecto y logo representativo.	Proyecto creado satisfactoriamente y cargado en sesión en fase de Lanzamiento y ciclo 1.	Resultado esperado.
Proyecto en ciclo n y fase de Lanzamiento	Proyecto en ciclo n y fase de Estrategia	Criterios de salida de la fase de Lanzamiento	Proyecto actualizado a fase de Estrategia y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Estrategia	Proyecto en ciclo n y fase de Planeación	Criterios de salida de la fase de Estrategia	Proyecto actualizado a fase de Planeación y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Planeación	Proyecto en ciclo n y fase de Requerimientos	Criterios de salida de la fase de Planeación	Proyecto actualizado a fase de Requerimientos y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Requerimientos	Proyecto en ciclo n y fase de Diseño	Criterios de salida de la fase de Requerimientos	Proyecto actualizado a fase de Diseño y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Diseño	Proyecto en ciclo n y fase de Implementación	Criterios de salida de la fase de Diseño	Proyecto actualizado a fase de Implementación y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Implementación	Proyecto en ciclo n y fase de Prueba	Criterios de salida de la fase de Implementación	Proyecto actualizado a fase de Prueba y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Prueba	Proyecto en ciclo n y fase de Postmortem	Criterios de salida de la fase de Prueba	Proyecto actualizado a fase de Postmortem y ciclo n.	Resultado esperado
Proyecto en ciclo n y fase de Postmortem	Proyecto en ciclo n + 1 y fase de Lanzamiento, o proyecto finalizado.	Criterios de salida de la fase de Postmortem	Proyecto actualizado a fase de Lanzamiento y ciclo n + 1. En caso de ser el último ciclo, proyecto finalizado.	Resultado esperado

Las siguientes figuras evidencian los criterios de salida de cada fase obtenidas de la prueba de datos reales. Cabe mencionar que algunos items de los criterios de salida se evalúan automáticamente con base en los datos actualmente almacenados en el sistema, lo que ofrece control y facilidad en la evaluación de determinados criterios del proyecto.

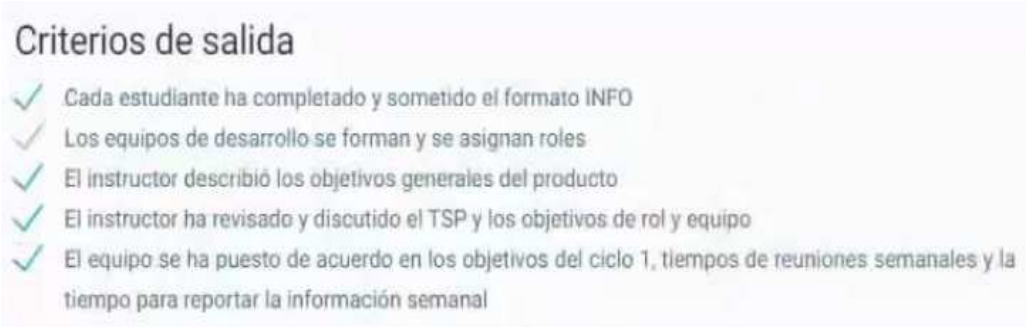


Figura 1. Criterios de salida de la fase de Lanzamiento



Figura 2. Criterios de salida de la fase de Estrategia



Figura 3. Criterios de salida de la fase de Planeación



Figura 4. Criterios de salida de la fase de Requerimientos

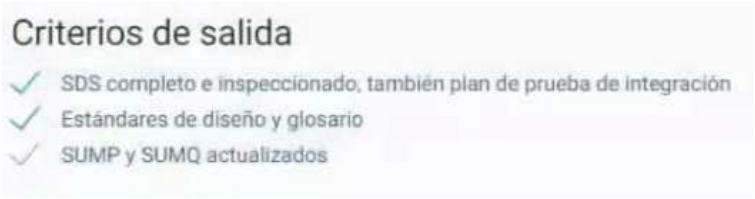


Figura 5. Criterios de salida de la fase de Diseño

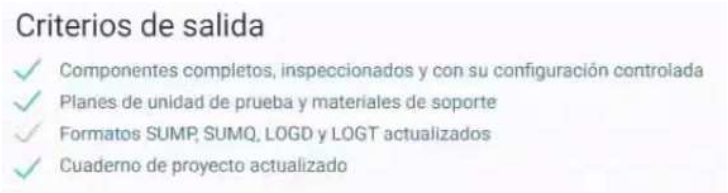


Figura 6. Criterios de salida de la fase de Implementación



Figura 7. Criterios de salida de la fase de Prueba



Figura 8. Criterios de salida de la fase de Postmortem

1.1.2 Disponibilidad de fases

Como parte de la buena experiencia del usuario cada fase posee un color establecido a fin de identificarlo rápidamente y un menú autodesplegable. La tabla 2 muestra el resultado de la revisión sobre el cumplimiento de estas dos características, de manera que se asigne el color de la fase sobre la que actualmente se encuentre el proyecto y se despliegue automáticamente el menú auxiliar a dicha fase. El porcentaje de aciertos sobre el cumplimiento de estos aspectos es del 100%.

Tabla 2. Revisión de disponibilidad de fases

Fase	Color	Acceso rápido
Lanzamiento	Correcto	Correcto
Estrategia	Correcto	Correcto
Planeación	Correcto	Correcto
Requerimientos	Correcto	Correcto
Diseño	Correcto	Correcto
Implementación	Correcto	Correcto
Prueba	Correcto	Correcto
Postmortem	Correcto	Correcto

1.1.3 Modificaciones sobre formatos

También se llevaron a cabo pruebas sobre la posibilidad de hacer modificaciones sobre formatos específicos de la metodología TSP con base en las fases y roles tal como se muestran en las tablas 3 y 4 respectivamente. Cada celda de estas tablas indica dos características sepadas por el carácter *slash* (/); la primera característica indica si en esa fase o ese rol respectivo tiene la posibilidad de hacer modificaciones sobre el formato, representando una S para *Sí* y una N para *No.* La segunda característica indica si al ejecutar la prueba el resultado es correcto (C) o incorrecto (I).

Tabla 3. Revisión sobre la modificación de formatos de acuerdo a la fase

Fase	CCR	PIP	PEER	SUMP	SUMQ	STRAT	LOGD	LOGT	WEEK	SCHEDULE	ITL	TASK
Lanzamiento	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C	N/C
Estrategia	N/C	N/C	N/C	N/C	N/C	S/C	S/C	N/C	N/C	N/C	S/C	N/C
Planeación	S/C	S/C	N/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	S/C
Requerimientos	S/C	S/C	N/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	N/C
Diseño	S/C	S/C	N/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	N/C
Implementación	S/C	S/C	N/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	N/C
Prueba	S/C	S/C	N/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	N/C
Postmortem	S/C	S/C	S/C	S/C	S/C	N/C	S/C	S/C	S/C	S/C	S/C	N/C

Tabla 4. Revisión sobre la modificación de formatos de acuerdo al rol

Rol	CCR	PIP	PEER	SUMP	SUMQ	STRAT	LOGD	LOGT	WEEK	SCHEDULE	ITL	TASK
Líder de equipo	S/C	S/C	S/C	S/C	S/C	N/I	S/C	S/C	S/C	S/C	S/C	S/C
Gerente de planeación	S/C	S/C	S/C	S/C	S/C	N/I	S/C	S/C	S/C	S/C	S/C	S/C
Gerente de soporte	S/C	S/C	S/C	S/C	S/C	N/I	S/C	S/C	S/C	S/C	S/C	S/C

Gerente de calidad/proceso	S/C	S/C	S/C	S/C	S/C	N/I	S/C	S/C	S/C	S/C	S/C	S/C
Gerente de desarrollo	S/C	S/C	S/C	S/C	S/C	S/C	S/C	S/C	S/C	S/C	S/C	S/C
Instructor	S/C	S/C	S/C	S/C	S/C	N/I	S/C	S/C	S/C	S/C	S/C	N/C

En síntesis sobre los resultados recolectados en las tablas X y X, todos los permisos sobre los formatos satisfacen las espectativas de la metodología TSP a excepción del formato STRAT, formato que sólo le corresponde al Gerente de desarrollo y que al momento de las pruebas, se le brindó a todos los roles de la metodología. Con respecto a esta característica evaluada, se obtuvo un acierto del 98.5%.

1.1.4 Rol dominante

Posteriormente se evaluó, lo que se denominó *Rol dominante*. Se probó que los formatos permitieran a los roles que tienen la mayor cantidad de deberes sobre un formato fueran los únicos con el privilegio de realizar modificaciones mayores sobre los mismos. Los resultados se muestran en la tabla 5, donde aquellos campos no diligenciados significan la no existencia de un rol dominante sobre el formato. El porcentaje de aciertos es de un total de 91.7% debido a que el formato STRAT poseía por roles dominantes a todos los integrantes del equipo y no únicamente al esperado Gerente de desarrollo.

Tabla 5. Revisión sobre los roles dominantes de los formatos

Formato	Rol dominante	Resultado
TASK	Gerente de planeación	Correcto
CCR	Gerente de planeación	Correcto
PIP	Gerente de calidad/proceso	Correcto
STRAT	Gerente de desarrollo	Incorrecto
PEER	-	Correcto
LOGD	-	Correcto
LOGT	-	Correcto
WEEK	-	Correcto
SCHEDULE	-	Correcto
ITL	-	Correcto
SUMP	-	Correcto
SUMQ	-	Correcto

1.1.5 Formatos transversales

En seguida se evaluaron los formatos de los que dispone todo integrante de un equipo TSP en todo momento, únicamente dependiendo del ciclo en el que se encuentre el proyecto actualmente. Esta prueba resumida en la tabla 6 indica la disponibilidad del formato en el menú general de acceso y en la página web enfocada en brindar información de la metodología. En todos los casos se obtuvieron los resultados esperados, sin embargo el formato IMP no se localizó en la sección de infomación lo que dio como resultado un 92.3% de acierto.

Tabla 6. Revisión sobre los formatos transversales

Formato	Ciclo	Menú	Información TSP
QUAL	-	Correcto	Correcto
LAU1	1	Correcto	Correcto
LAUN	N	Correcto	Correcto
STRAT1	1	Correcto	Correcto
STRATN	N	Correcto	Correcto
PLAN1	1	Correcto	Correcto
PLANN	N	Correcto	Correcto
REQ1	1	Correcto	Correcto
REQN	N	Correcto	Correcto
DES1	1	Correcto	Correcto
DESN	N	Correcto	Correcto
IMP1	1	Correcto	Incorrecto
IMPN	N	Correcto	Incorrecto

1.1.6 Cumplimiento de metas

En cuanto al automatizar y satisfacer la especificación del cumplimiento o no de las metas de cada rol, o del proyecto, se determinó si en la fase de postmortem se efectuaba esta transacción satisfactoriamente al tiempo que se daba disponibilidad a cada usuario únicamente sobre los roles que tiene a cargo en el ciclo actual del proyecto que está llevando a cabo. El resultado, con base en la tabla 7, fue un acierto del 100%.

Tabla 7. Revisión sobre el cumplimiento de metas

Meta	Postmortem	Disponibilidad
Metas de proyecto	Correcto	Correcto
Metas de Líder de equipo	Correcto	Correcto
Metas de Gerente de desarrollo	Correcto	Correcto
Metas de Gerente de calidad/proceso	Correcto	Correcto
Metas de Gerente de planeación	Correcto	Correcto
Metas de Gerente de soporte	Correcto	Correcto

1.2 Tablas de decisión

1.2.1 Usuario

Consiste en evaluar la gestión de los datos del usuario para interactuar con la aplicación y por ende, con sus servicios, tal como se muestra en la tabla 8. El total de aciertos sobre esta prueba fue del 100%.

Tabla 8. Revisión sobre la gestión de usuario

Acción	Condición	Sí	Resultado	No	Resultado
Inicio de sesión	¿Identificación y clave existen?	Cargar datos del usuario.	Correcto	Enviar mensaje de error	Correcto
Registro de usuario	¿Información completa?	Siguiente	Correcto	Enviar mensaje sobre los datos incompletos	Correcto
	¿Clave y confirmación de clave, iguales?	Siguiente	Correcto	Enviar mensaje de incompatibilidad de valores	Correcto
	¿Correo válido?	Siguiente	Correcto	Enviar mensaje de invalidez de correo electrónico	Correcto
	¿No existe el usuario?	Crear usuario e iniciar sesión	Correcto	Enviar mensaje indicado que el usuario ya se encuentra registrado	Correcto
Cierre de sesión	Cerrar sesión	Regreso al menú inicial	Correcto	Enviar alerta del error.	Correcto

1.2.2 Proyecto

Se hicieron pruebas para validar los datos de registro del proyecto y su gestión de fases/ciclo tal como se muestra en la tabla 9. El único error presentado fue la no validación de la existencia de un proyecto, de modo que al existir un proyecto del mismo nombre al proyecto a crear, no informaba claramente al usuario el motivo de la excepción, por tanto se obtuvo un porcentaje de acierto es del 94.4%.

Tabla 9. Revisión sobre la gestión del proyecto.

Acción	Condición	Sí	Resultado	No	Resultado
Crear proyecto	¿Datos completos?	Siguiente	Correcto	Enviar mensaje de datos incompletos	Correcto
	¿No existe el proyecto?	Siguiente	Correcto	Enviar mensaje indicando que el proyecto ya existe	Incorrecto
	¿Fecha de inicio de proyecto especificada?	Siguiente	Correcto	Asignar la fecha como la fecha actual	Correcto
	¿Fecha de inicio menor a la fecha actual?	Siguiente	Correcto	Enviar mensaje de error y asignar la fecha actual	Correcto

	¿Fecha estimada de finalización mínima requerida?	Siguiente	Correcto	Enviar mensaje de error y asigna la fecha mínima requerida.	Correcto
	¿Número de ciclos dentro del intervalo TSP?	Siguiente	Correcto	Enviar mensaje de error y asignar el número de ciclos máximos permitido	Correcto
	¿Identificación del instructor registrada?	Cargar la información del usuario instructor del proyecto	Correcto	Enviar mensaje de inexistencia del usuario y borrar datos ingresados del instructor	Correcto
	¿Logo del proyecto en formato de imagen?	Siguiente	Correcto	Enviar mensaje de error en el formato de imagen	Correcto
Siguiente fase/ciclo	¿El usuario posee el rol de Líder de equipo?	Validar criterios de salida de la fase actual y mostrar las opciones de validación restantes	Correcto	Enviar mensaje indicando el privilegio del rol	Correcto

1.2.3 Asignación de roles

Para la fase de lanzamiento, se hizo la prueba a la hora de asignar los roles del ciclo actual. El porcentaje de acierto fue del 100% tal que se valida la asignación de al menos un integrante por rol. Ver tabla 10.

Tabla 10. Revisión sobre la asignación de roles.

Acción	Condición	Sí	Resultado	No	Resultado
Asignar los roles	¿Todo los roles asignados?	Guardar asignación de roles	Correcto	Enviar mensaje solicitando la asignación de todos los roles	Correcto

1.2.4 Agregar meta

Se realizaron pruebas para validar la creación de metas para el proyecto en específico tal como se muestra en la tabla 11. Se obtuvo un acierto del 90% a causa de no controlar e informar al usuario la necesidad de diligenciar los datos sobre una meta, pues no se especificaba claramente la excepción ocurrida.

Tabla 11. Revisión sobre la adición de metas

Acción	Condición	Sí	Resultado	No	Resultado
Crear meta de proyecto	¿Nombre meta diligenciado?	Siguiente	Correcto	Enviar mensaje indicando que el nombre de la meta debe ser diligenciado	Correcto
	¿Valor de meta diligenciado?	Siguiente	Correcto	Enviar mensaje indicando que el valor de la meta debe ser diligenciado	Correcto
	¿Comparación seleccionada?	Siguiente	Correcto	Enviar mensaje indicando que la comparación de la meta debe seleccionarse	Correcto
	¿Unidad seleccionada?	Agregar la meta a la lista de metas del proyecto	Correcto	Enviar mensaje indicando que la unidad de la meta debe seleccionarse	Correcto
Crear meta de rol	¿Nombre meta diligenciada?	Agregar la meta a la lista de metas del respectivo rol	Correcto	Enviar mensaje indicando que el nombre de la meta debe ser diligenciado	Incorrecto

1.2.5 Formato STRAT

En la gestión del formato STRAT se realizaron las pruebas sobre los items mostrados en la tabla 12. Todas las validaciones de datos correspondientes dieron resultados positivos, sin embargo, se identificó la falta de control sobre el diligenciamiento del nombre de todas las funciones para cada parte del proyecto, obteniendo así un 95% de acierto.

Tabla 12. Revisión sobre el formato STRAT

Acción	Condición	Sí	Resultado	No	Resultado
Agregar parte	¿Nombre de la parte diligenciada?	Siguiente	Correcto	Indicar que el nombre de la parte no existe	Correcto
	¿Descripción de la parte diligenciada?	Agregar la parte a los nodos de la estrategia	Correcto	Indicar que la descripción es ogbligatoria	Correcto
Cambiar estimado de los puntos funcionales	¿Estimado de puntos funcionales >= 0?	Continuar	Correcto	Informar del control de valores positivos	Correcto
Cambiar estimado de las horas de ciclo	¿Estimado de horas de ciclo >= 0?	Continuar	Correcto	Informar del control de valores positivos	Correcto
Guardar formato STRAT	¿Nombres de las funciones diligenciadas?	Guardar formato STRAT	Correcto	Informar la falta de datos	Incorrecto

1.2.5 Formato ITL

La gestión del formato ITL controla principalmente las fechas de acción tal como se muestra en la tabla 13. Básicamente los errores encontrados corresponden a la falta de validación de las fechas ingresadas al momento de actualizar la información, obteniendo así un acierto del 75%.

Tabla 13. Revisión sobre el formato ITL

Acción	Condición	Sí	Resultado	No	Resultado
Agregar riesgo/problema	¿Todos los campos diligenciados?	Siguiente	Correcto	Informar acerca de la completitud de datos	Correcto
	¿Fecha de seguimiento > fecha de inicio del ciclo?	Siguiente	Correcto	Informar acerca de la restricción frente a la fecha de inicio del ciclo	Correcto
	¿Fecha de resolución > fecha de inicio del ciclo?	Siguiente	Correcto	Informar acerca de la restricción frente a la fecha de inicio del ciclo	Correcto
	¿Fecha de seguimiento <= fecha de resolución?	Continuar	Correcto	Informar acerca de la restricción entre seguimiento y resolución	Correcto
Actualizar fecha de seguimiento	¿Fecha de seguimiento > fecha de inicio del ciclo?	Siguiente	Correcto	Informar acerca de la restricción frente a la fecha de inicio del ciclo	Incorrecto
	¿Fecha de seguimiento <= fecha de resolución?	Continuar	Correcto	Informar acerca de la restricción entre seguimiento y resolución	Incorrecto
Actualizar fecha de resolución	¿Fecha de resolución > fecha de inicio del ciclo?	Siguiente	Correcto	Informar acerca de la restricción frente a la fecha de inicio del ciclo	Incorrecto
	¿Fecha de seguimiento <= fecha de resolución?	Continuar	Correcto	Informar acerca de la restricción entre seguimiento y resolución	Incorrecto

1.2.5 Formato TASK

Al probar las decisiones sobre la gestión del formato TASK se obtuvo un acierto del 100% tal que cada dato a registrar en éste se valida bajo las especificaciones de la metodología TSP. Ver tabla 14

Tabla 14. Revisión sobre el formato TASK

Acción	Condición	Sí	Resultado	No	Resultado
Cambiar número de horas estimadas en tarea por rol	¿Número de horas >= 0?	No hacer nada	Correcto	Actualizar número de horas a cero	Correcto
Cambiar tamaño de la tarea	¿Tamaño >= 0?	No hacer nada	Correcto	Actualizar tamaño a uno	Correcto
Cambiar número de semana de tarea	¿Número de semana < 0?	Hacer número de semana positivo	Correcto	Informar acerca de la restricción semanal	Correcto
	¿Número de semana = número de semana anterior + 1?	No hacer nada	Correcto	Informar acerca de la secuencialidad del plan	Correcto
Guardar formato TASK	¿Nombres de las tareas diligenciadas?	Siguiente	Correcto	Informar acerca de la completitud de las tareas	Correcto
	¿Tamaños de tarea > 0?	Siguiente	Correcto	Informar acerca del requisito del tamaño de las tareas	Correcto
	¿Partes seleccionadas?	Siguiente	Correcto	Informar acerca de la selección de los productos del software	Correcto
	¿Número de semanas secuenciales?	Siguiente	Correcto	Informar acerca de la secuencialidad el el número de semanas	Correcto
	¿Unidades de tamaño seleccionadas?	Guardar formato TASK	Correcto	Informar acerca de la selección de unidad de tamaño	Correcto

1.2.6 Formato LOGT

La gestión del formato LOGT se basa en la gestión de las actividades a registrar. Como se muestra en la tabla 15, se obtuvo un satisfactorio 100% de acierto en cada restricción estimada.

Tabla 15. Revisión sobre el formato LOGT

Acción	Condición	Sí	Resultado	No	Resultado
Iniciar actividad	¿No hay actividad iniciada?	Registrar fecha y hora de inicio de actividad seleccionada	Correcto	Ya se encuentra una actividad en ejecución	Correcto
Parar actividad	¿Existe actividad iniciada?	Siguiente	Correcto	No se puede parar una actividad no iniciada	Correcto
	¿Tarea terminada?	Registrar en formato TASK	Correcto	Debe especificar si ha terminado o no la tarea	Correcto
	¿Registro de tareas completado en TASK?	Llenar datos reales de tarea	Correcto	No hacer nada	Correcto
Marcar interrupción prevista	¿No existe otra interrupción activa?	Mostrar el tiempo marcado	Correcto	Ya se encuentra marcada otra interrupción	Correcto
Marcar interrupción imprevista	¿No existe otra interrupción activa?	Solicitar tiempo estimado de interrupción	Correcto	Ya se encuentra marcada otra interrupción	Correcto
Guardar interrupción prevista	¿Existe interrupción prevista activa?	Guardar interrupción	Correcto	Ya se encuentra marcada otra interrupción	Correcto
Guardar interrupción imprevista	¿Tiempo interrupción < tiempo real actual dedicado?	Guardar interrupción	Correcto	El tiempo de interupción no puede superar al tiempo total registrado	Correcto

1.2.7 Formato SCHEDULE

La gestión del formato SCHEDULE depende de los criterios mostrados en la tabla 16 para su generación y con base en ellos se determina un acierto del 100%.

Tabla 16. Revisión sobre el formato SCHEDULE

Acción	Condición	Sí	Resultado	No	Resultado
Actualizar formato SCHEDULE	¿Formato TASK registrado?	Siguiente	Correcto	Debe existir el formato TASK para obtener los datos	Correcto
	¿El día de hoy es igual al día seleccionado para revisión semanal?	Hacer resumen de valor ganado y planeado para todas las semanas no registradas hasta la fecha	Correcto	Sólo es posible generar el formato SCHEDULE el día especificado	Correcto

1.2.8 Formato WEEK

El formato WEEK, generado automáticamente, valida dos condiciones para llevarse a cabo tal como se muestra en la tabla 17, a partir de la cual se determina un acierto del 100%.

Tabla 17. Revisión sobre el formato WEEK

Acción	Condición	Sí	Resultado	No	Resultado
Generar formato WEEK	¿Formato SCHEDULE de la semana registrado?	Siguiente	Correcto	La semana actual ya se encuentra registrada	Correcto
	¿El día de hoy es igual al día seleccionado para revisión semanal?	Resumen de valor ganado, valor planeado y tiempos dedicados al proyecto	Correcto	El día no corresponde con el día de revisión semanal del proyecto	Correcto

1.2.9 Formato LOGD

La gestión del formato LOGD se fundamenta en el control de cada registro a ingresar sobre éste. Con base en las pruebas mostradas en la tabla 18, donde se valida la presencia de todos los datos de defecto, se determina un acierto satisfactorio del 100%.

Tabla 18. Revisión sobre el formato LOGD

Acción	Condición	Sí	Resultado	No	Resultado
Nuevo registro de defecto	¿Tipo de defecto seleccionado?	Siguiente	Correcto	Debe seleccionar el tipo de defecto	Correcto
	¿Etapa de inyección seleccionada?	Siguiente	Correcto	Debe seleccionar la etapa de inyección	Correcto
	¿Etapa de remoción seleccionada?	Siguiente	Correcto	Debe seleccionar la etapa de remoción	Correcto
	¿Tiempo de arreglo diligenciado?	Siguiente	Correcto	Debe diligenciar el tiempo de arreglo	Correcto
	¿Descripción diligenciada?	Nuevo defecto registrado	Correcto	Debe añadir una descripción	Correcto

1.2.10 Formato CCR

La gestión del formato CCR se centra en la completitud de los datos tal como se muestra en la tabla 19. Puesto que se obtuvieron los resultados esperados se considera un acierto del 100%.

Tabla 19. Revisión sobre el formato CCR

Acción	Condición	Sí	Resultado	No	Resultado
Guardar formato CCR	¿Datos completos?	Nuevo formato CCR creado	Correcto	Debe completar todos los datos solicitados	Correcto

1.2.11 Formato PIP

El formato PIP se gestiona con base en la completitud de los datos, única condición que se muestra en la tabla 20 y que al proporcionar resultados satisfactorios se determina un acierto del 100%.

Tabla 21. Revisión sobre el formato PIP

Acción	Condición	Sí	Resultado	No	Resultado
Guardar formato PIP	¿Datos completos?	Nuevo formato PIP creado	Correcto	Debe completar todos los datos solicitados	Correcto

1.2.12 Formato PEER

La gestión del formato PEER se basa en la completitud de los datos tal como se muestra en la tabla 22 en la que se obtuvo un acierto del 100%.

Tabla 22. Revisión sobre el formato PEER

Acción	Condición	Sí	Resultado	No	Resultado
Guardar formato PEER	¿Datos completos?	Formato PEER de ciclo actualizado	Correcto	Debe completar todos los datos solicitados	Correcto

1.2.13 Formato SUMP

A nivel de caja blanca el formato SUMP únicamente depende de la existencia del formato TASK, tal que, su generación automática valida esta condición. La tabla 23 muestra la validación de este criterio y por ende se deduce un acierto del 100% dado que las pruebas brindaron resultados satisfactorios.

Tabla 23. Revisión sobre el formato SUMP

Acción	Condición	Sí	Resultado	No	Resultado
Actualizar formato SUMP	¿Formato TASK generado?	Formato SUMP actualizado satisfactoriamente	Correcto	Debe existir el formato TASK para actualizar el formato	Correcto

1.2.14 Formato SUMQ

La gestión del formato SUMQ depende de la existencia del formato TASK como se muestra en la tabla 24. A partir de esta prueba se determinó un acierto del 100%.

Tabla 24. Revisión sobre el formato SUMQ

Acción	Condición	Sí	Resultado	No	Resultado
Actualizar formato SUMQ	¿Formato TASK generado?	Formato SUMQ actualizado satisfactoriamente	Correcto	Debe existir el formato TASK para actualizar el formato	Correcto

1.2.15 Confirmación

Como parte de los condicionales, se validó la presencia de mensajes de confirmación a las solicitudes de vitales cambios sobre las necesidades de los usuarios. La tabla 25 resume las pruebas sobre las confirmaciones tal que la columna de *Mensaje asociado* indica si el mensaje mostrado está relacionado con la solicitud del usuario y la columna *Solicitud* indica si dicha solicitud es correctamente llamada a partir de la confirmación.

Dado que al crear el usuario no se especifica el mensaje al respecto, se obtuvo un total de acierto del 91.6%.

Tabla 25. Revisión sobre los mensajes de confirmación.

Acción	Mensaje asociado	Solicitud
Crear usuario	Incorrecto	Correcto
Crear proyecto	Correcto	Correcto
Eliminar meta de proyecto	Correcto	Correcto
Iniciar actividad	Correcto	Correcto
Parar actividad	Correcto	Correcto

1.3 Comportamiento

1.3.1 Partición equivalente

A continuación se muestran los casos más importantes en el proceso de un proyecto de software bajo la metodología TSP, en forma secuencial; específicamente el proyecto aplicado *Caso de estudio cooperativa de egresados UPS* a cargo de la docente Alba Consuelo Nieto en un curso de orientación a TSP.

1.3.1.1 Registro de usuarios

El software brinda la interfaz mostrada en la figura 9 para el registro de usuarios, en este caso, el registro del estudiante Javier Enrique Mora. Los datos fueron validados satisfactoriamente, confirmando el registro e iniciando la sesión para dicho usuario.



Figura 9. Interfaz de registro de usuario. Fuente propia.

Este proceso tuvo que repetirse otras 5 veces para los demás integrantes del equipo, obteniendo los mismos resultados con base en los datos brindados por cada respectivo usuario.

1.3.1.2 Creación del proyecto

Se procede a crear el proyecto, llenando todos los datos correspondientes. El sistema validó satisfactoriamente los datos ingresados, creando el proyecto y actualizando la aplicación con el contenido correspondiente a dicho proyecto creado. En la figura 10 se pueden observar los parámetros del proyecto diligenciados en el formulario.

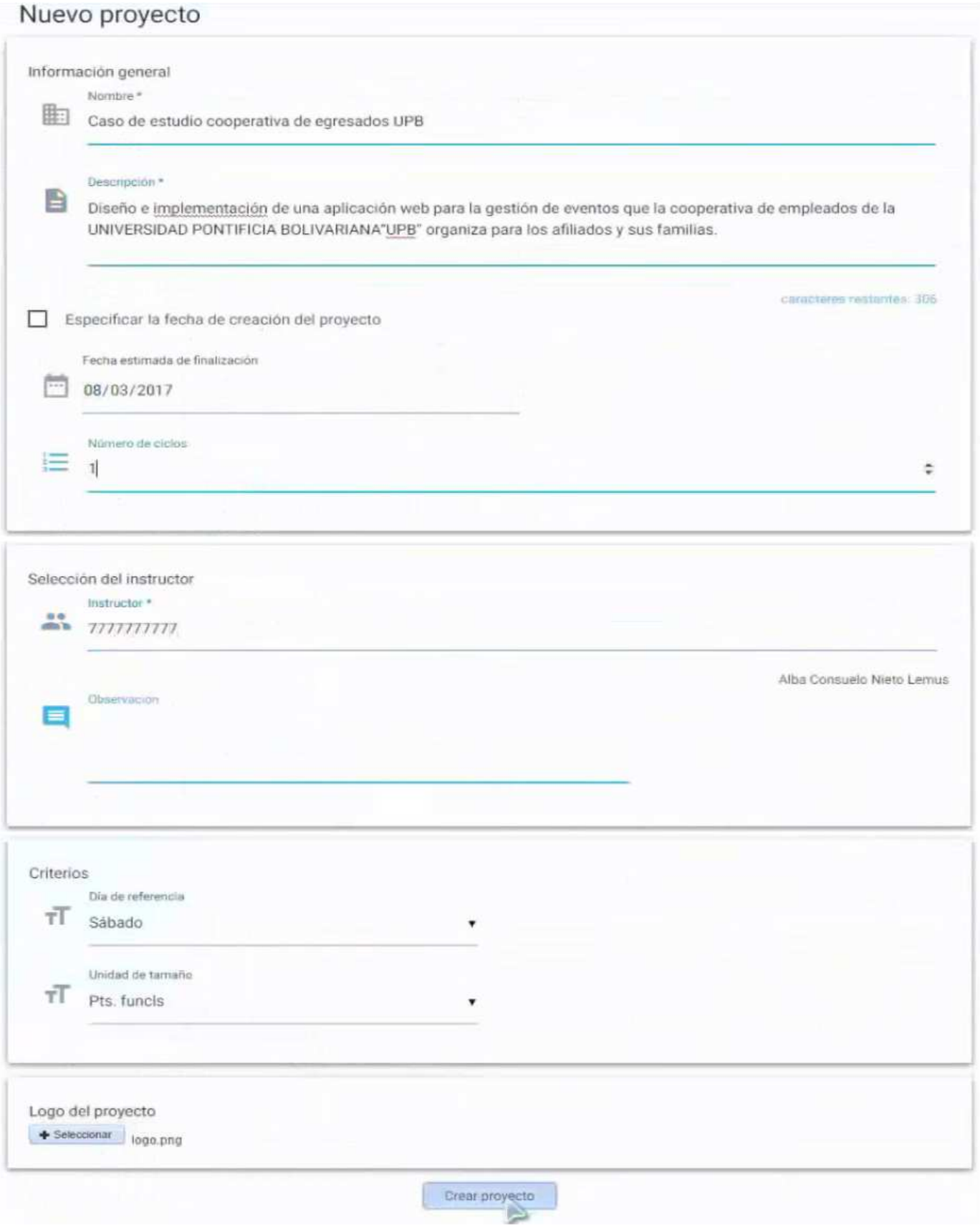


Figura 10. Interfaz para la creación de un proyecto. Fuente propia.

1.3.1.3 Asignación de roles

Como parte de la fase de la fase de Lanzamiento de la metodología TSP, se realiza el proceso de asignación de roles tal como se muestra en la figura 11. Cabe mencionar que, dado el tamaño del equipo TSP, se tuvo que asignar dos integrantes del equipo para el rol de Gerente de soporte.



Figura 11. Interfaz para la asignación de roles. Fuente propia.

1.3.1.4 Características del proyecto

Como parte de las características amigables, se identifica el logo de proyecto, roles, entre otras características del proyecto que se observan en la figura 12. De igual forma el usuario puede cambiar entre proyectos mediante la interfaz mostrada en la figura 13. En ambos casos, los datos fueron mostrados satisfactoriamente con cada proyecto seleccionado por el usuario.

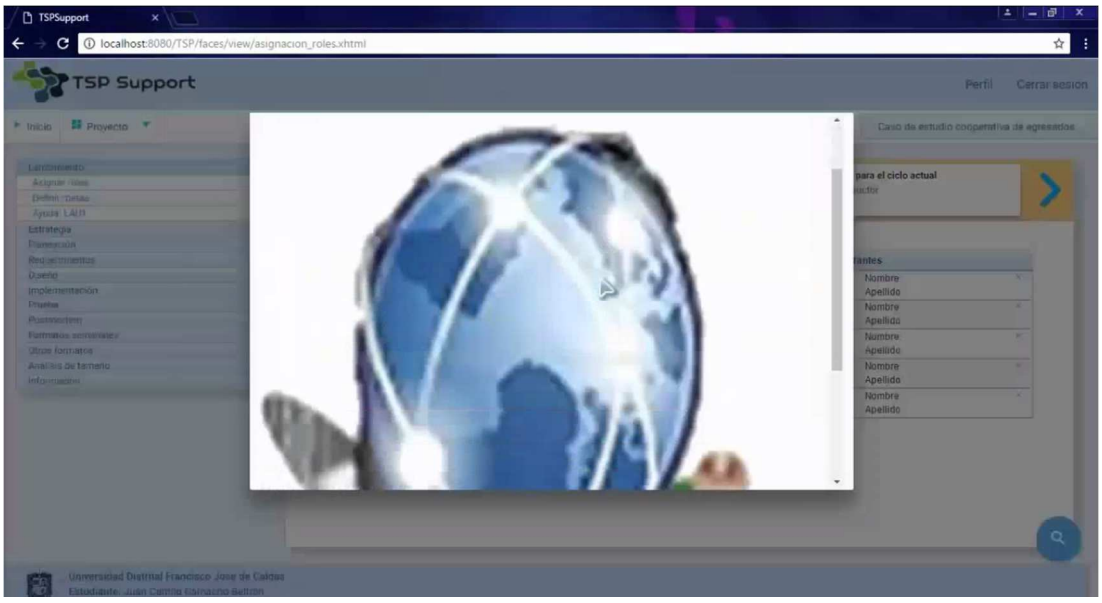


Figura 12. Interfaz para la identificación del logo del proyecto. Fuente propia.



Figura 13. Interfaz de selección de proyecto para trabajar. Fuente propia.

1.3.1.5 Definición de metas

Se definen las metas de ciclo y de cada rol como se muestra en las figuras 14, 15, 16, 17, 18, y 19. Con cada meta automáticamente calculable, los usuarios únicamente pueden cambiar su respectivo valor, en caso contrario, tienen la posibilidad de modificar la comparativa, el valor y la unidad de la meta. Al realizar las pruebas, se pudo gestionar satisfactoriamente la adición, eliminación y modificación de las metas específicas para el ciclo y los roles.



Figura 14. Gestión de las metas de proyecto. Fuente propia.

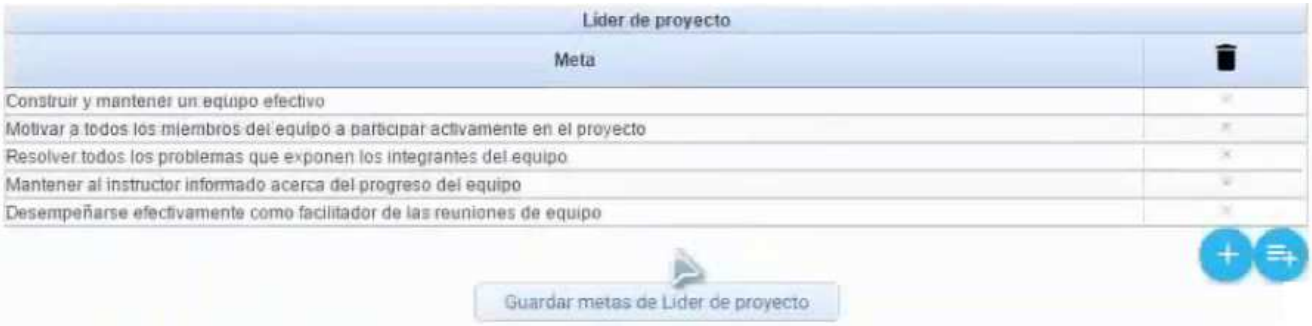


Figura 15. Gestión de las metas del líder de proyecto. Fuente propia.



Figura 16. Gestión de metas para el gerente de desarrollo. Fuente propia.



Figura 17. Gestión de metas para el gerente de planeación. Fuente propia.

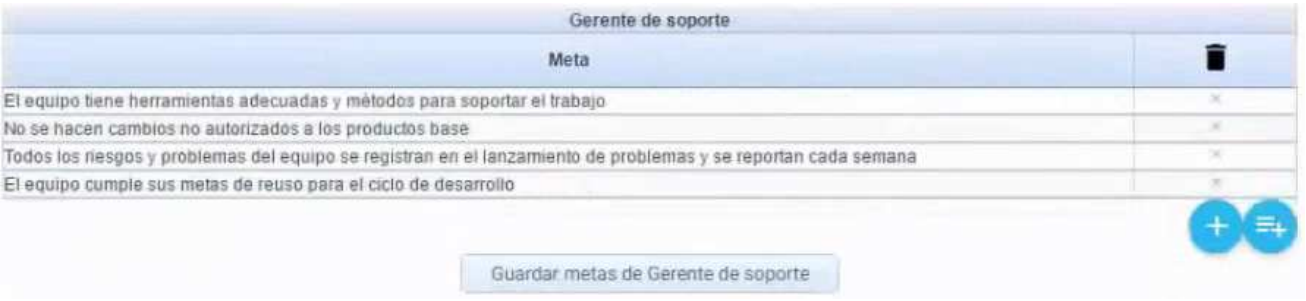


Figura 18. Gestión de las metas para el gerente de soporte. Fuente propia.

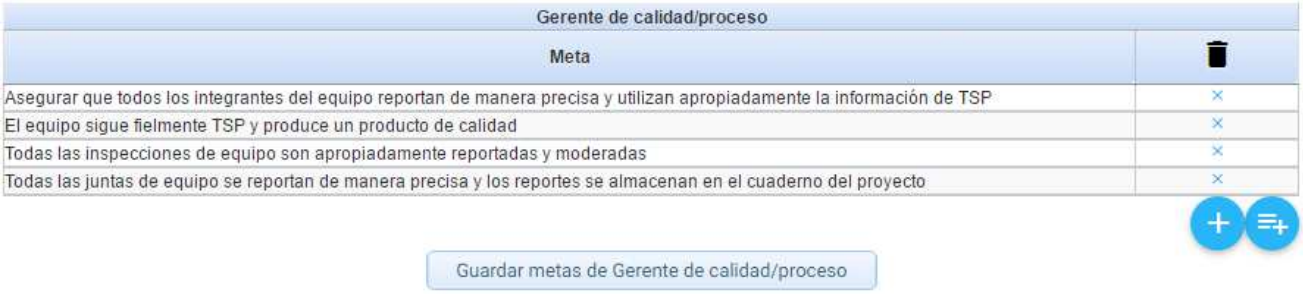


Figura 19. Gestión de las metas para el gerente de calidad/proceso. Fuente propia.

1.3.1.6 Cuaderno de proyecto

El cuaderno del proyecto, representado por un directorio compartido en el aplicativo, presentó los resultados esperados al generarse satisfactoriamente con el nombre del proyecto, el ciclo actual y los roles correspondientes. Cada rol tiene los privilegios de añadir y eliminar archivos y carpetas dentro de su dominio, y de descargar cualquier archivo alojado en cualquier carpea del dicho cuaderno. De igual manera, la herramienta provee todos los archivos del proyecto, desplegando únicamente las carpetas del ciclo actual, a fin de facilitar el contenido que dispone de cambios a la fecha. Cada una de estas características fue validada mediante la carga de documentación del proyecto real implementado a través de la interfaz mostrada en la figura 20, que ejecutada durante varias iteraciones, dio como resultado el directorio mostrador en la figura 21.



Figura 20. Gestión del directorio del proyecto. Fuente propia.

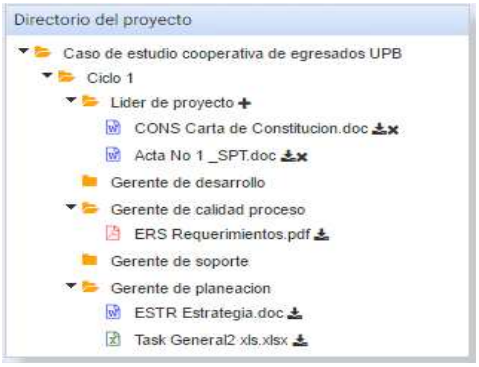


Figura 21. Interfaz del directorio del proyecto desde la perspectiva del rol líder de proyecto. Fuente propia.

También se revisó la asignación de íconos a cada tipo de archivo reconocido por el sistema, obteniendo los resultados esperados tanto en color como en diseño, teniendo en cuenta la gestión de carpetas al mostrar o ocultar su contenido.

1.3.1.7 Estrategia

Parte	Función	Pts. funcls	Horas		
		C1	C1		
Usuarios + ×	Crear usuario	125,0	3,0	⊗	+
	Modificar contraseña y datos de usuario - Activar/Desa	100,0	3,0	⊗	
Ventana Login + ×	Validar el usuario y el perfil en el sistema (funcionario	150,0	4,0	−	+
Inscripciones + ×	Crear inscripción de afiliado (con o sin grupo familiar)	100,0	2,0	⊗	+
	Modificar datos, cancelar inscripción	100,0	2,0	⊗	
	Detalle de la inscripción	75,0	2,0	⊗	
	Listar/Reportar inscripciones - filtros/opciones de cons	150,0	2,0	⊗	
Eventos + ×	Crear evento	125,0	3,0	⊗	+
	Modificar datos, abrir evento, cerrar evento, cancelar e	150,0	3,0	⊗	
	Detalle del evento - número de inscripciones, (total re	125,0	3,0	⊗	
	Listar eventos, filtro por estado	150,0	3,0	⊗	
Proveedores + ×	Crear proveedor	100,0	2,0	⊗	+
	Modificar datos	100,0	2,0	⊗	
	Ver detalle del proveedor	75,0	2,0	⊗	
	Listar proveedores - Opción de eliminar un proveedor	150,0	2,0	⊗	
Afiliados y familiares + ×	Crear afiliado, familiar	100,0	3,0	⊗	+
	Modificar datos del afiliado, activar o desactivar	100,0	3,0	⊗	
	Ver detalles	75,0	3,0	⊗	
	Listar/Reportar afiliados - Opción de eliminar un afilia	150,0	3,0	⊗	
Total		2200.0	50.0		

Figura 22. Interfaz de gestión de la estrategia. Fuente propia.

En lo que respecta al prototipo, la estrategia consiste en el diligenciamiento del formato STRAT, y por tanto la revisión se centró en la captura de sus respectivas mediciones. El resultado obtenido de la gestión de estrategia se observa en la figura 22, tal que, el sistema permitió agregar las partes del proyecto a diferentes niveles jerárquicos, sus respectivas funciones y estimados de tamaño y tiempo, controlando el rango de valores además de actualizaciones y remociones tanto de funciones como partes, efectuando satisfactoriamente el cálculo de los totales en tiempo real.

1.3.1.8 Estimado de riesgos y problemas

La gestión de riesgos y problemas decae sobre el formato ITL en el que se añadieron satisfactoriamente los riesgos considerados para algunos productos definidos en la fase de estrategia, permitiendo actualizar rápidamente los datos en caso de cometer algún error de diligenciamiento. En la figura 23 se muestra uno de los formatos ITL producidos, en este caso, para el producto Usuarios.



Figura 23. Interfaz de gestión de los riesgos y problemas. Fuente propia.

1.3.1.9 Gestión de tareas

Para la gestión de tareas, centrada en el diligenciamiento del formato TASK, se asignaron las tareas para cada etapa. De acuerdo a los datos del proyecto real, la clasificación determinó una tarea por etapa y dos para la etapa de *Estrategia y planeación,* diligenciando todos los valores respectivos a su vez que el sistema controló los valores para asegurar el correcto diligenciamiento. La figura 24 muestra el formato TASK (dividido verticalmente en dos partes) en el que cada integrante del equipo proporcionó los estimados de tiempo sobre las tareas diligenciadas por el gerente de planeación, actualizando inmediatamente los valores registrados, totalizando y determinando algunos valores automáticos.

La herramienta prototipo controló satisfactoriamente las modificaciones sobre el formato, en especial la gestión sobre el número de tareas por etapa y la secuencialidad del plan. De igual forma, el registro de valores reales, dependiente de la ejecución de las tareas (formato LOGT), se efectuó correctamente al hacer resumen y conversión de valores para especificar el logro de las tareas a lo largo del proyecto.

Tarea				Horas del plan			
Etapa	Producto	Nombre de la tarea	N. Integrts.	Líder de proyecto	Gerente de desarrollo	Gerente de planeación	Gerente de calidad/proceso
Gerencia y miscelanea	Usuarios	Gestión documental	6	2,5	2,5	2,5	2,5
Lanzamiento y estrategia	Usuarios	Lanzamiento	2	0	2,5	2,5	0,0
Estrategia y planeación	Usuarios	Planeación del tamaño	2	0	0	2,5	2,5
	Usuarios	Identificación de riesgos	3	0	0	2,5	0
Requerimientos	Usuarios	Administración de la configuración	4	0	0	2,5	2,5
Plan de prueba de sistema	Usuarios	Planeación general	6	2,5	2,5	1,5	2,5
Inspección de requerimientos	Inscripciones	Plan de calidad	3	2,5	0	2,0	2,5
Diseño de alto nivel	Eventos	M&M	4	2,0	3,0	0	0
Plan de prueba de integración	Eventos	Especificación de requerimientos (ERS)	5	3,0	2,0	0	2,0
Inspección de diseño de alto nivel (HLD)	Eventos	Inspección ERS	6	1,0	2,0	2,0	2,0
Diseño detallado	Eventos	Plan de diseño detallado	3	0	2,0	0	0
Revisión de diseño detallado (DLD)	Ventana Login	Gestión de revisión de diseño detallado	4	0	2,0	0	2,0
Desarrollo de prueba	Inscripciones	Plan de prueba de sistema	5	1,0	2,0	0	2,0
Inspección de diseño detallado (DLD)	Afiliados y familiares	Inspección plan de pruebas de sistema	4	1,0	2,0	0	0
Código	Ventana Login	Implementación	4	0	2,0	0	2,0
Revisión de código	Ventana Login	Revisión de código	5	1,0	1,0	0	2,0
Compilación	Inscripciones	Inspección de código	4	0,0	1,5	0	2,0
Inspección de código	Inscripciones	Plan de pruebas de unidad	4	1,0	2,0	0	0
Prueba de unidad	Ventana Login	Pruebas de unidad	4	0	1,5	2,0	0,0
Prueba de construcción e integración	Eventos	Construcción e integración	5	2,0	1,15	0	1,0
Prueba de sistema	Inscripciones	Pruebas de sistema	5	2,0	1,0	1,0	0
Documentación	Eventos	Documentación	5	2,0	2,0	2,0	0
Postmortem	Eventos	M&M Postmortem	6	1,0	1,0	1,0	1,3
			Total:	24.5	35.65	24.0	28.8

			Tamaño/Valor del plan					Real			
Gerente de soporte	Total horas	Acumulado horas	Unidad de tamaño	Tamaño	Número de semana	VP	VP Acumulado	Horas	Horas acumuladas	Número de semana	
5,0	15.0	15.0	Páginas	8,0	1	0,078145355	0,078145355	19.0529	19.0529	2	–
0	5.0	20.0	Páginas	4,0	1	0,02604845	0,10419381	5.6563	24.7092	2	–
0	5.0	25.0	Páginas	5,0	1	0,02604845	0,13024226	5.4895	30.1987	2	
5,0	7.5	32.5	Páginas	10,0	1	0,039072677	0,16931494	7.7541	37.9528	2	
5,0	10.0	42.5	Páginas	7,0	1	0,0520969	0,22141184	10.3554	48.3082	2	–
5,0	14.0	56.5	Páginas	7,0	1	0,07293566	0,2943475	15.5573	63.8655	2	–
0	7.0	63.5	Funciones	12,0	1	0,03646783	0,33081532	7.5016	71.3671	3	–
2,0	7.0	70.5	Páginas	15,0	1	0,03646783	0,36728314	7.2427	78.6098	2	–
4,0	11.0	81.5	Páginas	12,0	1	0,05730659	0,42458972	11.3301	89.9399	2	–
4,0	11.0	92.5	Páginas	14,0	2	0,05730659	0,4818963	11.75548	101.69538	3	–
4,0	6.0	98.5	Páginas	12,0	2	0,03125814	0,51315445	4.3332	106.02858	3	–
4,0	8.0	106.5	Páginas	7,0	2	0,04167752	0,554832	6.2182	112.24678	3	–
4,0	9.0	115.5	Líneas de código	6,0	2	0,04688721	0,6017192	10.8258	123.07258	3	–
4,0	7.0	122.5	Archivos	12,0	2	0,03646783	0,63818705	5.4761	128.54868	3	–
4,0	8.0	130.5	Líneas de código	1,0	2	0,04167752	0,6798646	8.2245	136.77318	3	–
4,0	8.0	138.5	Páginas	9,0	3	0,04167752	0,7215421	9.6064	146.37958	4	–
4,0	7.5	146.0	Líneas de código	11,0	3	0,039072677	0,7606148	5.2717	151.65128	4	–
4,0	7.0	153.0	Líneas de código	4,0	3	0,03646783	0,79708266	7.2329	158.88418	4	–
4,0	7.5	160.5	Archivos	12,0	3	0,039072677	0,83615535	9.573	168.45718	4	–
2,0	6.15	166.65	Funciones	1,0	3	0,032039594	0,86819494	6.4108	174.86798	4	–
3,0	7.0	173.65	Funciones	12,0	4	0,03646783	0,9046628	7.2895	182.15748	4	–
4,0	10.0	183.65	Funciones	8,0	4	0,0520969	0,9567597	8.2532	190.41068	5	–
4,0	8.3	191.95	Páginas	12,0	4	0,043240428	1,0000001	8.69858	199.10926	5	–
79.0											

Figura 24.Interfaz de gestión de las tareas. Fuente propia.

1.3.1.10Gestión del tiempo

Para el control de tareas, se validó y verificó la gestión de los tiempos para cada integrante en un rol determinado. La selección del rol brindó satisfactoriamente la lista de roles que el usuario en sesión tiene (figura 25). Una vez seleccionado el rol, el sistema cargó las tareas asignadas para éste en ciclo actual, tal como se esperaba (figura 26). Una vez el usuario seleccionó tanto el rol como la tarea, pudo dar comienzo a la misma, marcando las interrupciones deseadas y finalmente parando la actividad para diligenciar la información respectiva como se muestra en la figura 27.

Se comprobó que el sistema actualizara la información de la tarea con base en la finalización de éstas, haciendo síntesis del tiempo total dedicado a la tarea con base en cada registro efectuado en la gestión de los tiempos. De igual manera, se validaron los tiempos de registro, total de interrupciones y control de datos para ofrecer la integridad de los mismos.

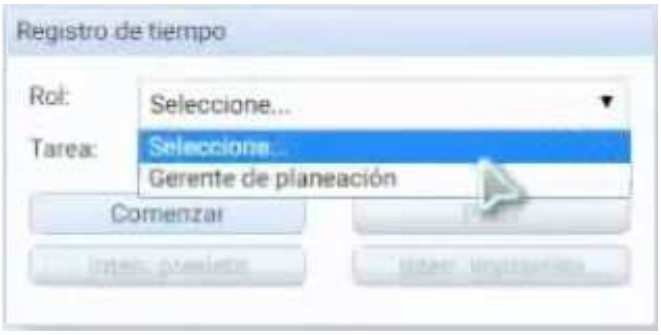


Figura 25. Interfaz de selección del rol para iniciar tarea. Fuente propia.

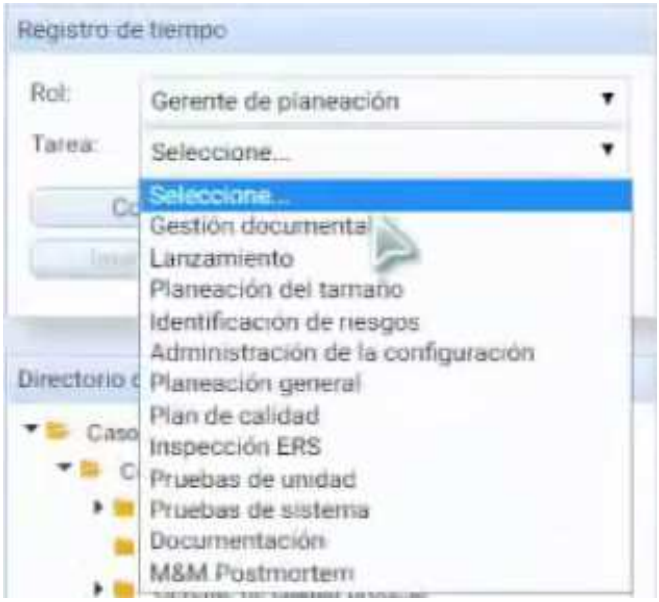


Figura 26. Interfaz de selección de la tarea para dar comienzo a la misma. Fuente propia.

En este punto en particular, se evaluó en mayor medida la característica responsiva debido al interés del sistema por colaborar en los procesos de ingeniería y por tanto no incomodar en los mismos. En la figura 28 se puede observar el sistema en su menor tamaño, proporcionando una fácil gestión en la ejecución de las tareas.

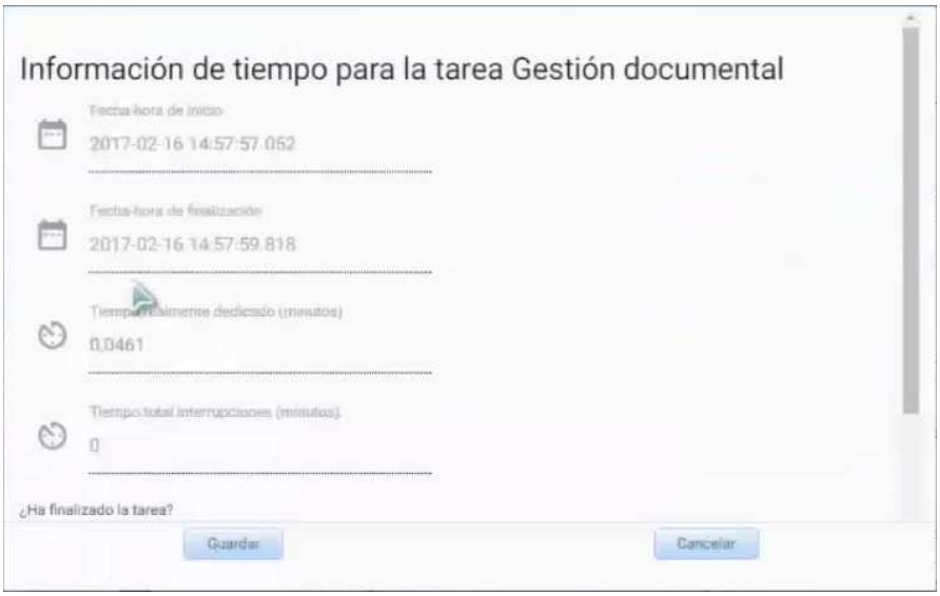


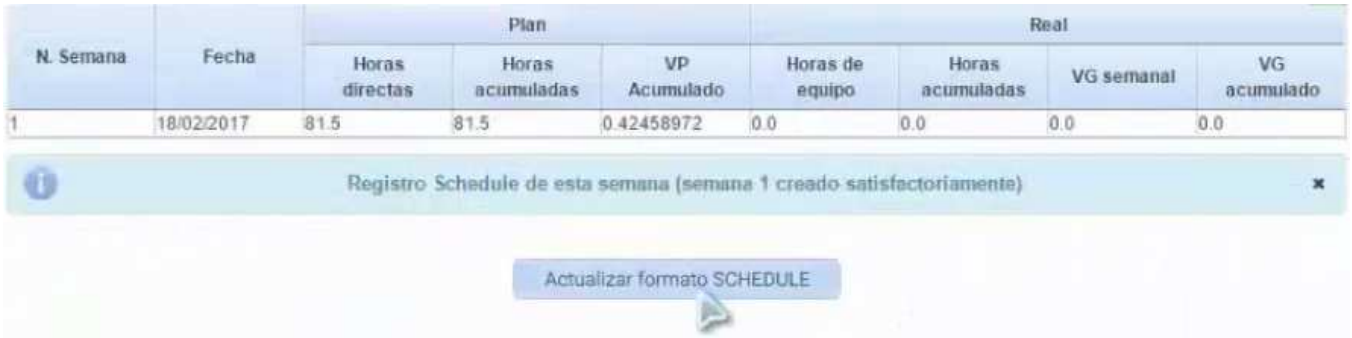
Figura 27. Interfaz de cierre de tarea. Fuente propia.



Figura 28. Interfaz general del prototipo en tamaño pequeño. Fuente propia.

1.3.1.11Progreso semanal

El progreso semanal se evaluó bajo la identificación de las tareas estimadas para la semana actual y las realmente terminadas, logrando con el sistema generar automáticamente esta información a través de los formatos SCHEDULE (figura 29) y WEEK (figura 30), corroborando los valores obtenidos y garantizando su generación con base en los criterios del proyecto y la secuencialidad de su ejecución.



N. Semana	Fecha	Plan			Real			
		Horas directas	Horas acumuladas	VP Acumulado	Horas de equipo	Horas acumuladas	VG semanal	VG acumulado
1	18/02/2017	81.5	81.5	0.42458972	0.0	0.0	0.0	0.0

Registro Schedule de esta semana (semana 1 creado satisfactoriamente)

Actualizar formato SCHEDULE

Figura 29. Interfaz de progreso semanal SCHEDULE. Fuente propia.



Formato WEEK (Semana 1)

Información semanal	Plan	Real
Horas del proyecto para esta semana	81.5	0.0
Horas del proyecto para este ciclo a la fecha	81.5	0.0
Valor ganado para esta semana	0.42458972	0.0
Valor ganado este ciclo a la fecha	0.42458972	0.0
Horas totales para las tareas completadas en esta fase hasta la fecha	81.5	0.0

Información semanal	Horas planeadas	Horas reales	Valor Ganado	Valor planeado
Líder de proyecto	12.5	0.0	0.0	0.06885473716666667
Gerente de desarrollo	12.5	0.0	0.0	0.06972301883333333
Gerente de planeación	16.0	0.0	0.0	0.10532256816666667
Gerente de calidad/proceso	14.5	0.0	0.0	0.08708865216666667
Gerente de soporte	26.0	0.0	0.0	0.09360076566666667
Total	81.5	0.0	0.424589742	

Tareas completadas	Horas planeadas	Horas realess	VValor Ganado	Semana planeada
Totales	0.0	0.0	0.0	

Generar Week

Figura 30. Interfaz de progreso semanal WEEK. Fuente propia.

1.3.1.12Indicadores y métricas

Con base en las mediciones tomadas, se verificaron los resultados obtenidos en los indicadores y métricas recolectados en los formatos SUMP y SUMQ, determinando que los resultados fueron los esperados y actualizados con base en los datos automáticos y semiautomáticos gestionados hasta su generación o actualización.

En las figuras 31 y 32 se puede observar el formato SUMP separado en dos partes, el cual evidencia el resumen de planeación contra el resumen de los datos reales capturados en la ejecución del ciclo. En las figuras 33 y 34 se puede observar el formato SUMQ separado en dos partes, el cual brinda indicadores de calidad y métricas de las etapas implementadas.

Formato SUMP			
Tamaño del producto		Plan	Real
Páginas de requerimientos (SRS)		5,0	10,0
Otras páginas de texto		3,0	4,0
Páginas de diseño de alto nivel (SDS)		7,0	10,0
Total de líneas de diseño		0	0
Código base (B)		1550,0	1600,0
Código eliminado (D)		0	0
Código modificado (M)		0	0
Código añadido (A)		0	0
Código reutilizado (R)		0	0
Código total nuevo y reutilizado (N)		0,0	0,0
Código total (T)		1550,0	1600,0
Intervalo de predicción (70%)		70,0	70,0
Tiempo en fase (horas)	Plan	Real	Real%
Gerencia y miscelanea Tiempo en fase (horas)	15.0	19.0529	9.569067757069663
Lanzamiento y estrategia	5.0	5.6563	2.8408020802246967
Estrategia y planeación	12.5	13.2436	6.651423444595193
Requerimientos	10.0	10.3554	5.20086308391684
Plan de prueba de sistema	14.0	15.5573	7.813448756727838
Inspección de requerimientos	7.0	7.5016	3.7675796695743835
Diseño de alto nivel	7.0	7.2427	3.637550558924281
Plan de prueba de integración	11.0	11.3301	5.690393304660969
Inspección de diseño de alto nivel (HLD)	11.0	11.75548	5.904034799787815
Diseño detallado	6.0	4.3332	2.176292554148411
Revisión de diseño detallado (DLD)	8.0	6.2182	3.123008944938071
Desarrollo de prueba	9.0	10.8258	5.437115280323979
Inspección de diseño detallado (DLD)	7.0	5.4761	2.7502990066860775
Código	8.0	8.2245	4.130646661034248
Revisión de código	8.0	9.6064	4.824687711661427
Compilación	7.5	5.2717	2.647641802294881
Inspección de código	7.0	7.2329	3.632628638165799
Prueba de unidad	7.5	9.573	4.807913002137621
Prueba de construcción e integración	6.15	6.4108	3.219739754946606
Prueba de sistema	7.0	7.2895	3.6610552417300934
Documentación	10.0	8.2532	4.145060857541231
Postmortem	8.3	8.69858	4.368747088909878
Total	191.95000000000002	199.10926	100.0

Figura 31. Interfaz de resumen SUMP parte 1. Fuente propia.

Defectos inyectados	Plan	Real	Real%
Estrategia y planeación	2.5	0	0.0
Requerimientos	2.0	0	0.0
Plan de prueba de sistema	2.8000000000000003	1	8.333333333333332
Inspección de requerimientos	1.4000000000000001	0	0.0
Diseño de alto nivel	1.4000000000000001	0	0.0
Plan de prueba de integración	2.2	0	0.0
Inspección de diseño de alto nivel (HLD)	2.2	0	0.0
Diseño detallado	1.2000000000000002	0	0.0
Revisión de diseño detallado (DLD)	1.6	0	0.0
Desarrollo de prueba	1.8	0	0.0
Inspección de diseño detallado (DLD)	1.4000000000000001	0	0.0
Código	1.6	0	0.0
Revisión de código	1.6	0	0.0
Compilación	1.5	0	0.0
Inspección de código	1.4000000000000001	0	0.0
Prueba de unidad	1.5	3	25.0
Prueba de construcción e integración	1.2300000000000002	1	8.333333333333332
Prueba de sistema	1.4000000000000001	7	58.333333333333336
Total	30.73	12	100.0
Defectos removidos	**Plan**	**Real**	**Real%**
Estrategia y planeación	1.75	0	0.0
Requerimientos	1.4	0	0.0
Plan de prueba de sistema	1.9600000000000002	0	0.0
Inspección de requerimientos	0.9800000000000001	0	0.0
Diseño de alto nivel	0.9800000000000001	0	0.0
Plan de prueba de integración	1.54	0	0.0
Inspección de diseño de alto nivel (HLD)	1.54	0	0.0
Diseño detallado	0.8400000000000002	0	0.0
Revisión de diseño detallado (DLD)	1.12	0	0.0
Desarrollo de prueba	1.26	0	0.0
Inspección de diseño detallado (DLD)	0.9800000000000001	0	0.0
Código	1.12	3	25.0
Revisión de código	1.12	0	0.0
Compilación	0.75	4	33.33333333333333
Inspección de código	0.9800000000000001	5	41.66666666666667
Prueba de unidad	1.35	0	0.0
Prueba de construcción e integración	0.9840000000000002	0	0.0
Prueba de sistema	1.12	0	0.0
Total	21.774400000000008	12	100.0
Prueba de aceptación	5.821399999999995	0.0	
Vida de producto	14.777399999999988	0.0	
Resumen		**Plan**	**Real**
Código/Hora		0.0	0.0
CPI (índice de costo_desempeño)			0.9640435608067651
Porcentaje de reuso		0.0	0.0

Figura 32. Interfaz de resumen SUMP parte 2. Fuente propia.

Formato SUMQ		
Porcentaje libre de defectos		**Real(%)**
En compilación		100.0
En prueba de unidad		0.0
En construcción e integración		0.0
En prueba de sistema		0.0
Defectos/Página	**Plan**	**Real**
Inspección de requerimientos	0.196	0.0
Inspección de diseño de alto nivel (HLD)	0.22	0.0
Defectos/Código	**Plan**	**Real**
Revisión de diseño detallado (DLD)	0	0
Inspección de diseño detallado (DLD)	0	0
Revisión de código	0	0
Compilación	0	0
Inspección de código	0	0
Prueba de unidad	0	0
Construcción e integración	0	0
Prueba de sistema	0	0
Desarollo total	0	0
Prueba de aceptación	0	0
Vida de producto	0	0
Total	0	0
Proporciones de defecto	**Plan**	**Real**
Revision diseño detallado/Diseño detallado	0.8296296296296296	0.0
Revisión de código/Compilación	1.4933333333333334	0.0
Proporciones de tiempo de desarrollo	**Plan**	**Real**
Inspección de requerimientos/Requerimientos	0.7	0.724414315236495
Inspección de diseño de alto nivel (HLD)/Diseño de alto nivel	1.5714285714285714	1.6230797906857939
Diseño detallado/Código	0.75	0.5268648550063832
Revisión diseño detallado (DLD)/Diseño detallado	1.3333333333333333	1.4350133850272317
Revisión de código/Código	1.0	1.1680223721806797
A/FR	**Plan**	**Real**
A/FR	23.466666666666665	21.787937869407937
Rendimientos de fase	**Plan (%)**	**Real (%)**
Estrategia y planeación	70,0	0.0
Requerimientos	70	0
Plan de prueba de sistema	70,0	0.0
Inspección de requerimientos	70	0.0
Diseño de alto nivel	70,0	0.0
Plan de prueba de integración	70,0	0.0
Inspección de diseño de alto nivel (HLD)	70	0.0
Diseño detallado	70,0	0.0
Revisión de diseño detallado (DLD)	70	0.0
Desarrollo de prueba	70,0	0.0
Inspección de diseño detallado (DLD)	70	0.0
Código	70,0	0.0
Revisión de código	70	-1.5
Compilación	50	-0.0
Inspección de código	70	-0.6666666666666666
Prueba de unidad	90	-0.625
Prueba de construcción e integración	80	-0.0
Prueba de sistema	80	0

Figura 33. Interfaz de resumen SUMQ Parte 1. Fuente propia.

Rendimientos de proceso	Plan	Real
Antes de compilación	0.7000000000000001	3.0
Antes de prueba de unidad	0.711904761904762	0.75
Antes de prueba de construcción e integración	0.7160045402951194	0.6
Antes de prueba de sistema	0.7197396963123647	0.5
Antes de prueba de aceptación	0.7085584119752687	1.0
Tasa de inyección de defectos	**Plan(.)**	**Real(.)**
Estrategia y planeación	0,2	0.0
Requerimientos	0,2	0.0
Plan de prueba de sistema	0,2	0.06427850591040862
Inspección de requerimientos	0,2	0.0
Diseño de alto nivel	0,2	0.0
Plan de prueba de integración	0,2	0.0
Inspección de diseño de alto nivel (HLD)	0,2	0.0
Diseño detallado	0,2	0.0
Revisión de diseño detallado (DLD)	0,2	0.0
Desarrollo de prueba	0,2	0.0
Inspección de diseño detallado (DLD)	0,2	0.0
Código	0,2	0.0
Revisión de código	0,2	0.0
Compilación	0,2	0.0
Inspección de código	0,2	0.0
Prueba de unidad	0,2	0.31338138514572234
Prueba de construcción e integración	0,2	0.15598677232170713
Prueba de sistema	0,2	0.9602853419301735
Tasa de remoción de defectos	**Plan(.)**	**Real(.)**
Estrategia y planeación	0.14	0.0
Requerimientos	0.13999999999999999	0.0
Plan de prueba de sistema	0.14	0.0
Inspección de requerimientos	0.14	0.0
Diseño de alto nivel	0.14	0.0
Plan de prueba de integración	0.14	0.0
Inspección de diseño de alto nivel (HLD)	0.14	0.0
Diseño detallado	0.14000000000000004	0.0
Revisión de diseño detallado (DLD)	0.14	0.0
Desarrollo de prueba	0.14	0.0
Inspección de diseño detallado (DLD)	0.14	0.0
Código	0.14	0.36476381542950936
Revisión de código	0.14	0.0
Compilación	0.1	0.7587685186941594
Inspección de código	0.14	0.6912856530575564
Prueba de unidad	0.18000000000000002	0.0
Prueba de construcción e integración	0.16000000000000003	0.0
Prueba de sistema	0.16	0.0
Inspección/Proporciones de revisión	**Plan**	**Real**
Inspección de requerimientos	0.7142857142857143	1.3330489495574278
Inspección de diseño de alto nivel (HLD)	0.6363636363636364	0.850667093134436
Revisión de diseño detallado (DLD)	0.0	0.0
Inspección de diseño detallado (DLD)	0.0	0.0
Revisión de código	0.0	0.0
Inspección de código	0.0	0.0

Figura 34. Interfaz de resumen SUMQ parte 2. Fuente propia.

1.3.1.13Registro de defectos

Para el registro de defectos se evaluó el control de tiempos y las dependencias entre éstos. El prototipo brindó cada formato de defectos (LOGD) por cada producto del sistema definido en la fase de estrategia, permitiendo actualizar esta información, de ser necesario. En la figura 35 se pueden observar los defectos sobre el producto *Usuarios.*



Figura 35. Interfaz de defectos LOGD. Fuente propia.

1.3.1.14Logro de metas

Para finalizar el ciclo, es importante aclarar el logro de metas, es decir, especificar el cumplimiento o no de las metas del ciclo. En la figura 36 se puede observar la interfaz que permite definir el logro o no de las metas. Las pruebas demostraron el registro correcto del logro en forma automática y manual.

Meta	Comparativa	Valor	Unidad de medida	Logro
Información registrada del ciclo	igual a	70.00	porcentaje	No lograda
Completar el desarrollo con un corto desfase de días	igual a	1.00	día(s)	Lograda
Defectos encontrados en compilación	igual a	80.00	porcentaje	No lograda
Defectos encontrados en la prueba de sistema	igual a	10.00	cantidad	No lograda
Requerimientos funcionales incluidos al terminar el proyecto 1era fase	igual a	60.00	porcentaje	Lograda
Requerimientos funcionales incluidos al terminar el proyecto 2da fase	igual a	100.00	porcentaje	Lograda
Error en el estimado de tamaño del producto	menor a	20.00	porcentaje	No lograda
Error en las horas estimadas de desarrollo	menor a	18.00	cantidad	Lograda

Líder de proyecto	
Meta	Logro
Construir y mantener un equipo efectivo	Lograda
Motivar a todos los miembros del equipo a participar activamente en el proyecto	No lograda
Resolver todos los problemas que exponen los integrantes del equipo	Lograda
Mantener al instructor informado acerca del progreso del equipo	Lograda
Desempeñarse efectivamente como facilitador de las reuniones de equipo	Lograda

Gerente de desarrollo	
Meta	Logro
Producir un producto superior	Lograda
Utilizar por completo las habilidades de los integrantes	No lograda

Gerente de calidad/proceso	
Meta	Logro
Asegurar que todos los integrantes del equipo reportan de manera precisa y utilizan apropiadamente la información de TSP	Lograda
El equipo sigue fielmente TSP y produce un producto de calidad	Lograda
Todas las inspecciones de equipo son apropiadamente reportadas y moderadas	No lograda
Todas las juntas de equipo se reportan de manera precisa y los reportes se almacenan en el cuaderno del proyecto	Lograda

Gerente de soporte	
Meta	Logro
El equipo tiene herramientas adecuadas y métodos para soportar el trabajo	Lograda
No se hacen cambios no autorizados a los productos base	No lograda
Todos los riesgos y problemas del equipo se registran en el lanzamiento de problemas y se reportan cada semana	Lograda
El equipo cumple sus metas de reuso para el ciclo de desarrollo	Lograda

Gerente de planeación	
Meta	Logro
Reportar de manera exacta el estatus semanal del equipo	Seleccione...
Producir un plan completo, preciso y exacto, para cada integrante y el equipo en general	Seleccione...

Figura 36. Interfaz de logro de metas. Fuente propia.

1.3.11.15 Diligenciamiento PEER

El prototipo logró gestionar correctamente el formato PEER, permitiendo diligenciar un solo formato por ciclo, validando la completitud de éste y la totalización de porcentajes correspondientes. En la figura 37 se puede observar un ejemplo del diligenciamiento del formato.

Formato PEER						
Trabajo y dificultad por cada rol						
Rol	Trabajo requerido		Dificultad			
Líder de proyecto	20,0		20,0			
Gerente de desarrollo	20,0		20,0			
Gerente de planeación	20,0		20,0			
Gerente de calidad/proceso	20,0		20,0			
Gerente de soporte	20,0		20,0			
Total	100.0		100.0			
Evaluación de criterios del equipo						
Espíritu de equipo	○ 1	○ 2	○ 3	○ 4	● 5	
Efectividad general	○ 1	○ 2	○ 3	○ 4	● 5	
Experiencia gratificante	○ 1	○ 2	○ 3	○ 4	● 5	
Productividad de equipo	○ 1	○ 2	○ 3	○ 4	● 5	
Calidad del proceso	○ 1	○ 2	○ 3	○ 4	● 5	
Calidad del producto	○ 1	○ 2	○ 3	○ 4	● 5	
Contribución general						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	
Ayuda y soporte						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	
Desempeño						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	

Figura 37. Interfaz de evaluación PEER. Fuente propia.

1.3.11.16 Gestión cualitativa de métricas

Las métricas gestionadas a partir de los datos capturados se representaron satisfactoriamente a través de distintas gráficas; algunas de ellas se muestran en las figuras 38, 39, 40, 41 y 42. Sin embargo, algunos de estos resultados cualitativos se sometieron a cambios debido a algunos nombres mal asignados o por representaciones mejor asociadas al tipo de información representada.

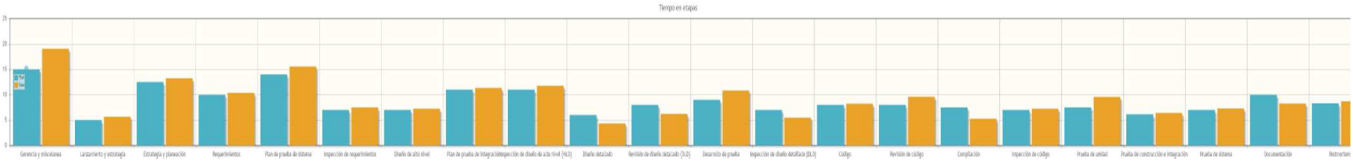


Figura 38. Comparación tiempo planeado y tiempo real por etapa. Fuente propia.



Figura 39. Porcentaje de defectos en compilación. Fuente propia.

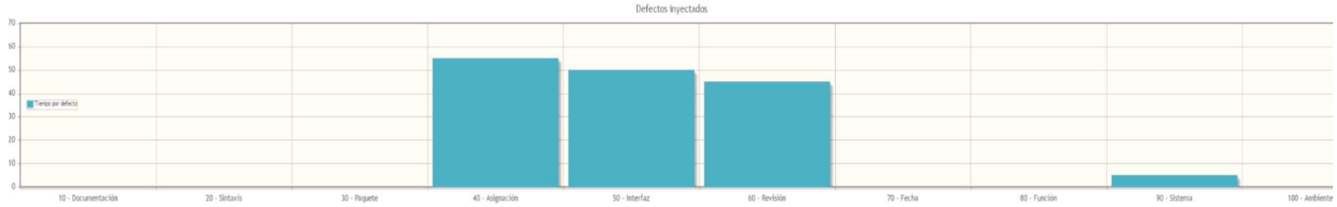


Figura 40. Número de defectos encontrados por etapa. Fuente propia.

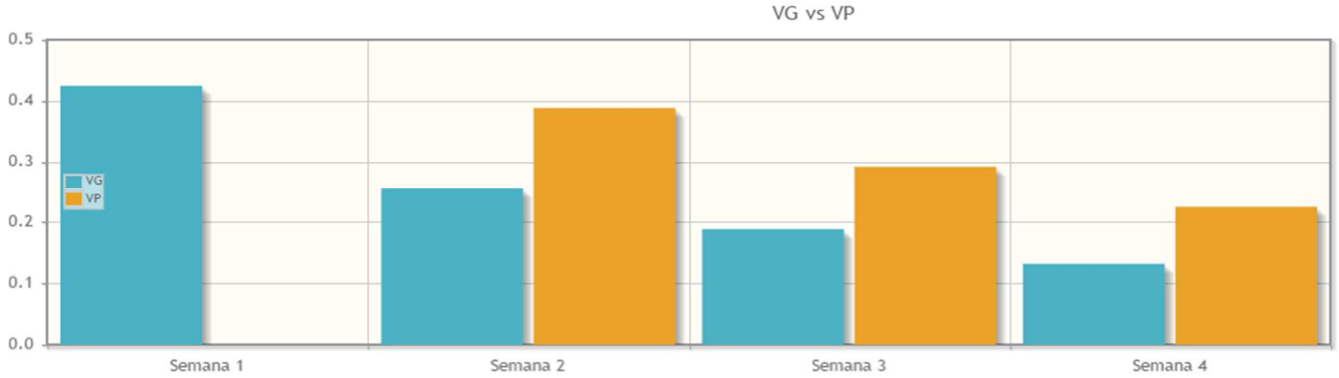


Figura 41. Valor ganado contra valor planeado. Fuente propia.

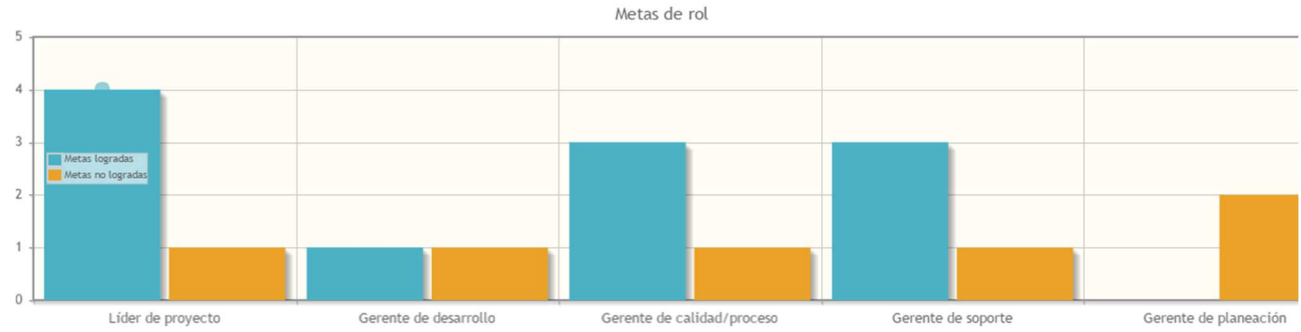


Figura 42. Cumplimiento de metas por rol. Fuente propia.

1.3.2 Límites

1.3.2.1 Gestión de proyectos

La creación de muchos proyectos no afecta significativamente el prototipo. En la figura 43 se puede observar el caso para 15 proyectos, los cuales se listan y se ajustan con la barra de desplazamiento del costado derecho.

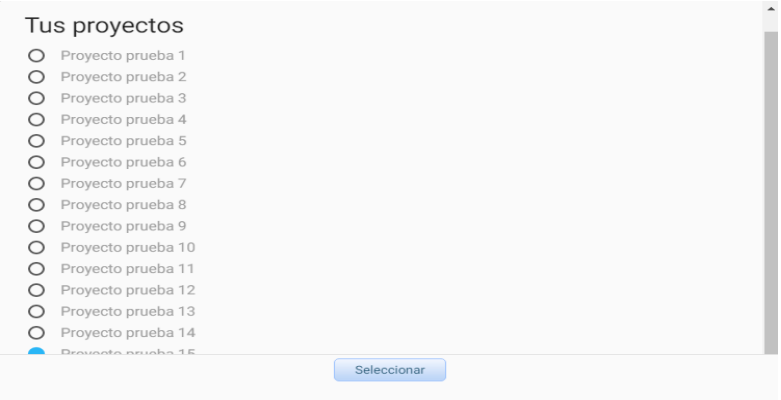


Figura 43. Prueba de selección de 15 proyectos. Fuente propia.

1.3.2.2 Adaptación cualitativa

Sin importar la cantidad de proyectos, los resultados cualitativos se ajustan perfectamente con la cantidad de éstos, brindando resultados coherentes para los usuarios tal como se muestra en la figura 44, un ejemplo sobre el tiempo dedicado en cada proyecto.

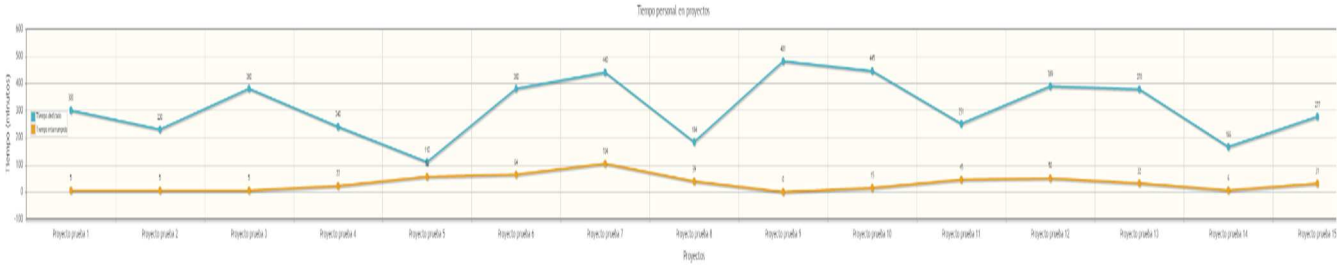


Figura 44. Prueba de gestión cualitativa para varios proyectos. Fuente propia.

2. PRUEBAS NO FUNCIONALES (Caja gris y negra)

2.1 Técnicas de prueba estructurales

2.1.1 Inyección de defectos

2.1.1.1 Login

En la prueba de login, se ingresaron datos no registrados y el sistema informó satisfactoriamente el resultado de la solicitud, tal como se muestra en la figura 45.



Figura 45. Defecto sobre intento de inicio de sesión. Fuente propia.

2.1.1.2 Registro de usuario

Al momento de registrar el usuario, se dejaron algunos campos sin diligenciar y el sistema informó el requerido para cada uno de ellos; un ejemplo de esto se encuentra en la figura 45.

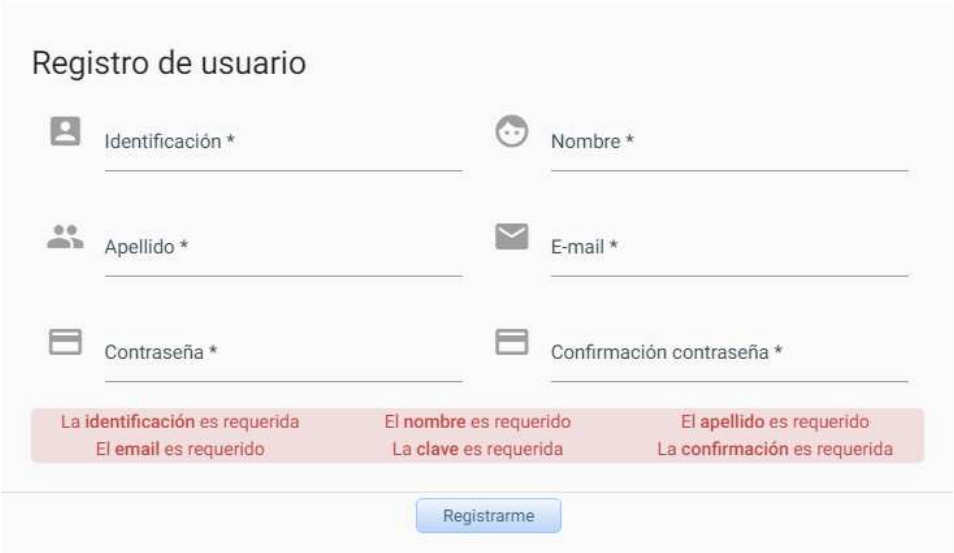


Figura 46. Prueba sobre el registro de usuario. Fuente propia.

En caso de que el usuario ya se encuentre registrado y todos los campos sean correctamente diligenciados, el sistema informa la imposibilidad de registrar un usuario que ya se encuentra en el sistema. En la figura 46 se puede observar un ejemplo de esto.



Figura 47. Defecto de existencia para registro de usuario. Fuente propia.

2.1.1.3 Schedule repetido

La actualización del formato SCHEDULE para la semana actual sobre la que se hace la solicitud, valida si el día de solicitud corresponde al día de la semana o en su defecto informa del error. Lo mismo sucede si la semana actual ya se encuentra registrada en el formato SCHEDULE; en ese caso el sistema informa la existencia para la semana actual como se muestra en la figura 47.



Figura 48. Defecto de generación SCHEDULE semanal. Fuente propia.

2.1.1.4 Creación de proyecto

Se inyectaron defectos sobre la creación de un proyecto bajo la metodología TSP en el prototipo. Sin importar el defecto almacenado (falta de datos o datos mal diligenciados), el sistema presentó un error catastrófico, pues toda la interfaz del usuario se bloqueó, como se muestra en la figura 48, debido a un problema de renderización Ajax mal referenciado.

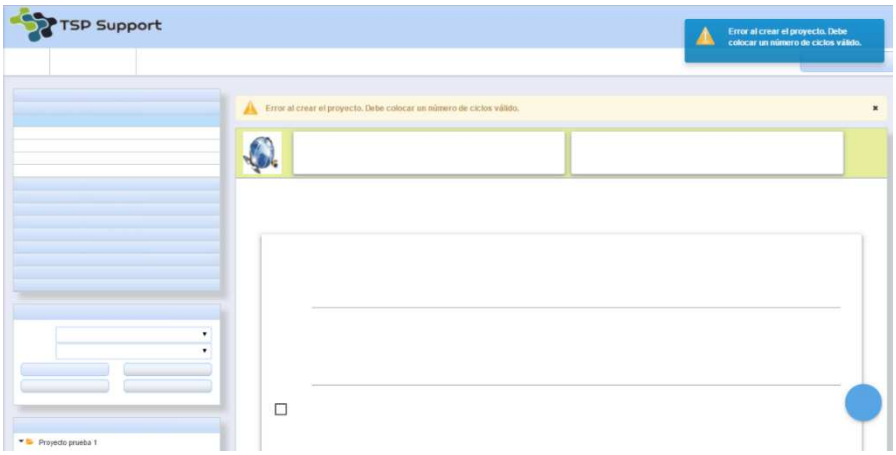


Figura 49. Defecto de datos en creación de proyecto. Fuente propia.

Una vez corregido el error catastrófico en la creación del proyecto, se obtuvo todos los resultados esperados al no diligenciar algunos datos del proyecto o diligenciar, por ejemplo, fechas fuera del rango permitido.

2.1.1.5 Asignación de roles

La principal validación sobre la asignación de roles consiste en no permitir asignar el mismo rol a un usuario más de una vez. En la figura 49 se puede observar cómo se controla la asignación de roles, informando al usuario al respecto.



Figura 50. Defectos sobre la reasignación de rol. Fuente propia.

2.1.1.6 Validación de metas

Al momento de crear una meta, el usuario debe diligenciar todo el formulario, y de faltar algún dato, el sistema debe informar del campo faltante. En la figura 50 se puede observar el ejemplo de inyección de error sobre una nueva meta de proyecto en la que se reconoce que el valor para la meta hace falta por ingresar.



Figura 51. Defecto sobre datos incompletos en meta. Fuente propia.

2.1.1.7 Siguiente fase/ciclo

El paso a la siguiente fase o ciclo depende del líder de equipo; otro integrante del proyecto no puede efectuar esta transacción. En la figura 51 se puede observar cómo el instructor trata de ir a la siguiente fase y el mensaje de excepción correspondiente.

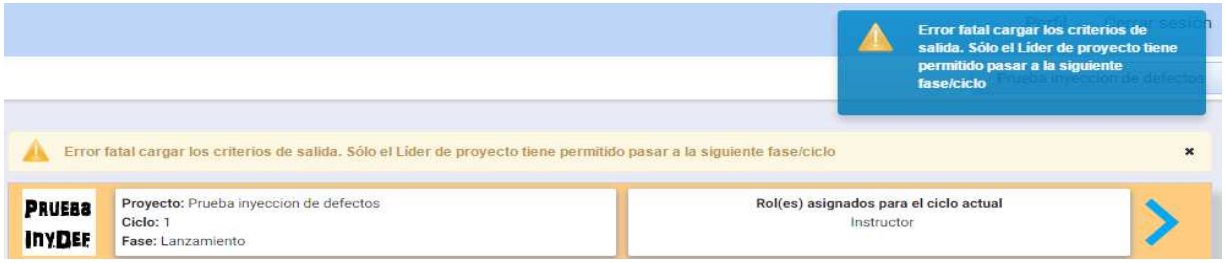


Figura 52. Defecto sobre el paso a la siguiente fase. Fuente propia.

2.1.1.8 Nuevo producto

Al crear un producto deben diligenciarse por lo menos el nombre y descripción del producto. En la figura 52 se puede observar el intento de ingreso de producto al sistema sin nombre ni descripción y la excepción que devuelve el sistema al respecto.

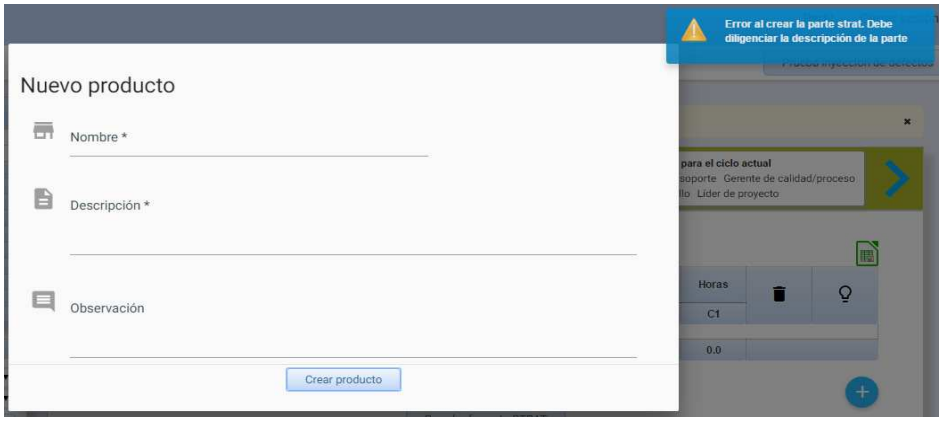


Figura 53. Defecto de datos incompletos de producto. Fuente propia.

2.1.1.9 Nuevo riesgo o problema

Cualquier riesgo o problema a registrar también es validado por el sistema, controlando las fechas de ingreso y los valores diligenciados. En la figura 53 se observa un error de fecha no muy claro para el usuario, debido a incompatibilidad con el navegador Mozilla, lo que se corrigió posteriormente y finalmente proveer información clara de la falta de valores en el formulario.

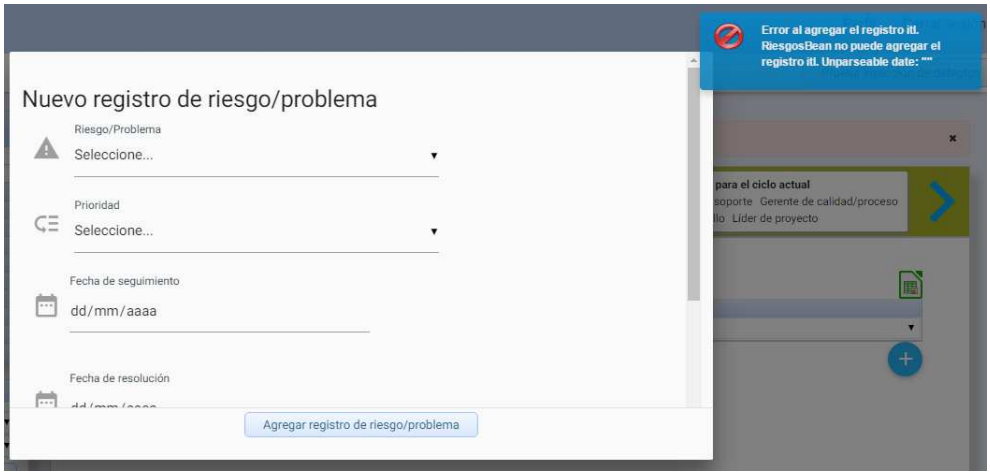


Figura 54. Defecto sobre registro de riesgo o problema. Fuente propia.

2.1.1.10 Diligenciamiento de tareas

Cada tarea del formato TASK debe tener todos los datos solicitados en el formulario con su respectivo contexto, el cual controla y valida el prototipo, enviando mensajes como el mostrado en la figura 54, en el que la parte del primer registro no fue asignado.

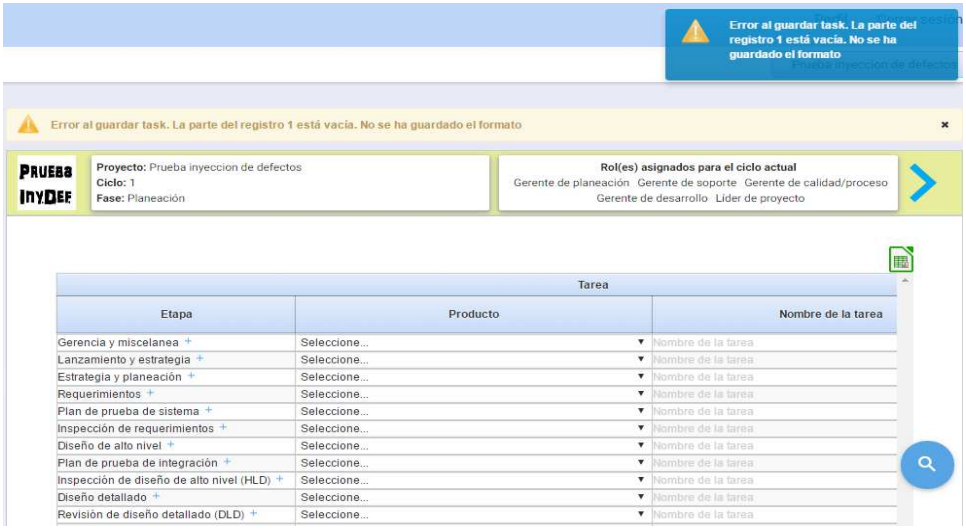


Figura 55. Defecto sobre el registro de las tareas. Fuente propia.

2.1.1.11 Otras validaciones

Validaciones semejantes a las ya mencionadas se efectuaron con los demás servicios del sistema, controlando la existencia de formatos dependientes, roles asignados, y valores dentro de los rangos permitidos. Además de las dos excepciones no controladas de los apartados 2.1.1.4 y 2.1.19, se obtuvieron todos los resultados esperados.

2.1.2 Estático

A continuación se muestra la revisión de los archivos de implementación, identificando, con base en su propósito, si su localización es la adecuada para el proyecto y si precisa las diferencias de funcionalidad. Para esta prueba se obtuvo un 100% de acierto frente a los resultados esperados.

Tabla 26. Evaluación estática del paquete de convertidores.

Paquete: a_web/a_converters	**Propósito: Mapear listas de objetos con la interfaz**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
ConvertidorProyecto.java	Mapea la lista de proyectos del usuario en sesión	Correcto	Correcto

Tabla 27. Evaluación estática sobre el paquete web.

Paquete: a_web/b_backing	**Propósito: Proveedor de servicios al cliente**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
AccesoBean.java	Fachada de control de acceso del usuario al sistema	Correcto	Correcto
CalidadBean.java	Fachada de gestión de métricas de calidad	Correcto	Correcto
CodigoBean.java	Fachada de análisis de código	Correcto	Correcto
CompBean.java	Fachada de gestión de formatos complementarios	Correcto	Correcto
DefectosBean.java	Fachada de gestión de defectos de proyecto	Correcto	Correcto
DocumentoBean.java	Fachada de gestión del cuaderno de proyecto	Correcto	Correcto
EstrategiaBean.java	Fachada de gestión del formato STRAT de TSP	Correcto	Correcto
GeneralBean.java	Fachada genérica para el reuso de funciones de gestión web.	Correcto	Correcto
GraficaBean.java	Fachada de gestión cualitativa de métricas	Correcto	Correcto
LanzamientoBean.java	Fachada de gestión de la fase de Lanzamiento del proyecto	Correcto	Correcto
PlaneacionBean.java	Fachada de gestión del formato TASK de TSP	Correcto	Correcto
ProyectoBean.java	Fachada de gestión del proyecto	Correcto	Correcto
ReportesBean.java	Fachada para la gestión de reportes	Correcto	Correcto
RiesgosBean.java	Fachada de gestión de riesgos del proyecto	Correcto	Correcto
SemanalBean.java	Fachada de gestión de métricas capturadas semanalmente	Correcto	Correcto
TiempoBean.java	Fachada de gestión de tiempo sobre las tareas; gestión del formato LOGT de TSP.	Correcto	Correcto
UsuarioBean.java	Fachada de gestión de los datos de usuario.	Correcto	Correcto

Tabla 28. Evaluación estática sobre la gestión de los datos.

Paquete: b_controlador/a_gestion	**Propósito: Procesar las solicitudes del cliente para la gestión del proyecto y sus respectivas métricas**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
AccesoControl.java	Controlador del acceso a la aplicación con base en la sesión.	Correcto	Correcto
CalidadControl.java	Controlador de métricas de calidad del proyecto	Correcto	Correcto
CodigoControl.java	Controlador de comparación de código.	Correcto	Correcto
CompControl.java	Controlador de formatos complementarios.	Correcto	Correcto
DefectosControl.java	Controlador de defectos del proyecto.	Correcto	Correcto
DocumentoControl.java	Controlador de la documentación del proyecto; cuaderno de proyecto.	Correcto	Correcto
EstrategiaControl.java	Controlador del formato STRAT de TSP.	Correcto	Correcto
Controlador.java	Generalizador de funciones de control.	Correcto	Correcto
GraficaControl.java	Controlador cualitativo de métricas.	Correcto	Correcto
LanzamientoControl.java	Controlador de procesos de la fase de Lanzamiento TSP.	Correcto	Correcto
PlaneacionControl.java	Controlador del formato TASK de TSP.	Correcto	Correcto
ProyectoControl.java	Controlador del proyecto respecto a ciclos, fases y roles.	Correcto	Correcto
ReportesControl.java	Controlador de reportes de resultados obtenidos.	Correcto	Correcto
RiesgosControl.java	Controlador de riesgos identificados en proyecto.	Correcto	Correcto
SemanalControl.java	Controlador de métricas semanales de proyecto.	Correcto	Correcto
TiempoControl.java	Controlador del tiempo dedicado a las tareas del proyecto.	Correcto	Correcto
UsuarioControl.java	Controlador de datos del usuario en sesión.	Correcto	Correcto

Tabla 29. Evaluación estática sobre la fachada de formatos.

Paquete: b_controlador/b_fachada	**Propósito: Ocultar la adquisición de datos o la ejecución de servicios**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FormatoFachada.java	Ocultar la obtención de formatos	Correcto	Correcto

Tabla 30. Evaluación estática sobre la gestión de formatos.

Paquete: b_controlador/c_fabricas/a_fabrica_formatos	**Propósito: Crear los formatos solicitados de acuerdo a sus especificaciones**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**

FabricaAbstractaFormatos.java	Posibilitar la construcción de diferentes fábricas de formatos.	Correcto	Correcto
FabricaFormatos.java	Construye los formatos TSP	Correcto	Correcto

Tabla 31. Evaluación estática sobre la gestión de atributos

Paquete: b_controlador/c_fabricas/b_fabrica_atributos	**Propósito: Crear los atributos de cada formato**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FabricaAbstractaAtributos.java	Posibilitar la construcción de diferentes fábricas de atributos.	Correcto	Correcto
FabricaAtributos.java	Construye los atributos de los formatos TSP.	Correcto	Correcto

Tabla 32. Evaluación estática sobre los revisores de código

Paquete: b_controlador/c_fabricas/c_fabrica_revisorescodigo	**Propósito: Crear los revisores de código solicitados**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FabricaAbstractaRevisCodigo.java	Posibilitar la construcción de diferentes fábricas de revisores de código.	Incorrecto	Correcto
FabricaRevisoresCodigo.java	Construye los revisores de código.	Correcto	Correcto

Tabla 33. Evaluación estática sobre la gestión cualitativa.

Paquete: b_controlador/c_fabricas/d_fabrica_graficas	**Propósito: Crear las gráficas requeridas**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FabricaAbstractaGraficas.java	Posibilitar la construcción de diferentes fábricas de revisores de código.	Incorrecto	Correcto
FabricaGraficas.java	Construye la fábrica de gráficas.	Correcto	Correcto

Tabla 34. Entidades del modelo relacional.

Paquete: c_negocio/a_relacional	**Propósito: Representación del modelo relacional en clases de negocio**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
Ciclo.java	Representar el ciclo de proyecto	Correcto	Correcto
CicloId.java	Representar la llave primaria de ciclo	Correcto	Correcto

Criterio.java	Representar los criterios TSP	Correcto	Correcto
Documento.java	Representar el documento de proyecto	Correcto	Correcto
Fase.java	Representar la fase de proyecto	Correcto	Correcto
FaseId.java	Representar la llave primaria de fase	Correcto	Correcto
Meta.java	Representar la meta de proyecto	Correcto	Correcto
MetaId.java	Representar la llave primaria de meta	Correcto	Correcto
Opciontsp.java	Representar las opciones nativas de TSP.	Correcto	Correcto
Parte.java	Representar la parte de proyecto	Correcto	Correcto
Proyecto.java	Representar al proyecto	Correcto	Correcto
PyCr.java	Representar los criterios de proyecto	Correcto	Correcto
PyCrId.java	Representar la llave primaria de los criterios del proyecto	Correcto	Correcto
RlCl.java	Representar el rol asignado en un ciclo de proyecto	Correcto	Correcto
RlClId.java	Representar la llave primaria del rol asignado en un ciclo de proyecto	Correcto	Correcto
Rol.java	Representar el rol de TSP	Correcto	Correcto
Usuario.java	Representar al usuario del sistema	Correcto	Correcto

Tabla 35. Entidades del modelo no relacional del atributo.

Paquete: c_negocio/b_no_relacional/atributo	**Propósito: Representación del atributo de formato del modelo no relacional en clases de negocio**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
AtributoCompuesto.java	Representación de un atributo de formato TSP	Correcto	Correcto
Atributo.java	Representación general de un atributo	Correcto	Correcto

Tabla 36. Entidades del modelo no relacional del formato.

Paquete: c_negocio/b_no_relacional/formato	**Propósito: Representación del formato del modelo no relacional en clases de negocio**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FormatoConcreto.java	Representación de un formato TSP	Correcto	Correcto
Formato.java	Representación general de un formato	Correcto	Correcto

Tabla 37. Gestión estática de otros elementos del prototipo.

Paquete: c_negocio/c_otros	**Propósito: Representación de clases con propósitos específicos en aplicación y no en persistencia**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
RevisorCodigo.java	Representación general de un revisor de código.	Correcto	Correcto
RevisorCodigoJava.java	Representación de un revisor de código en lenguaje Java.	Correcto	Correcto

Tabla 38. Gestión estática sobre la persistencia en el modelo relacional.

Paquete: d_datos/a_dao	**Propósito: Definición clara de las operaciones sobre la base de datos relacional dentro del contexto del negocio**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
ProyectoDAO.java	Ejecutar operaciones de persistencia con respecto a la ejecución del proyecto.	Correcto	Correcto
UsuarioDAO.java	Ejecutar operaciones de persistencia que tengan relación directa con los usuarios del sistema.	Correcto	Correcto

Tabla 39. Gestión estática sobre la gestión de conexiones a persistencia.

Paquete: d_datos/b_fabrica_conexiones	**Propósito: Obtener instancias de persistencia a fin de reducir costos de procesamiento**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
FabricaAbstConexiones.java	Posibilitar la construcción de diferentes fábricas de conexiones.	Correcto	Correcto
FabricaConexiones.java	Fabricación de conexiones a motores de bases de datos.	Correcto	Correcto

Tabla 40. Gestión estática sobre las transacciones.

Paquete: d_datos/c_datasource	**Propósito: Conexiones a motores de bases de datos**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**
DataSource.java	Definir una interfaz de acciones de persistencia	Correcto	Correcto
SourceMongo.java	Definir acciones de persistencia hacia el motor MongoDB.	Correcto	Correcto
SourcePostgres.java	Definir acciones de persistencia hacia el motor PostgreSQL.	Correcto	Correcto

Tabla 41. Gestión estática de elementos utilitarios.

Paquete: e_utilitaria	**Propósito: Proveer herramientas de utilidad fuera del modelo de negocio**		
Archivo	**Propósito del archivo**	**Localización**	**Funciones privadas/públicas/protegidas**

ExceptionFatal.java	Capturar errores en la aplicación	Correcto	Correcto
ExceptionWarn.java	Capturar excepciones por defectos de usuario	Correcto	Correcto
GestorArchivos.java	Manipular carpetas y archivos	Correcto	Correcto
Helper.java	Operar y transformar datos	Correcto	Correcto
NewHibernateUtil.java	Establecer la conexión al motor relacional mediante el filtro Hibernate.	Correcto	Correcto
Serial.java	Transformación de un objeto a json y viceversa.	Correcto	Correcto
TSP.java	Establecer valores de la metodología TSP.	Correcto	Correcto

2.2 Calidades sistémicas

2.2.1 Manifiestas

2.2.1.1 Rendimiento

La tabla 42 muestra el resultado de la prueba de rendimiento sobre la verificación obtenida en el escenario LAN de tres computadores y un router. En esta prueba se hizo una grabación local en cada equipo de los procesos efectuados, y a partir de ellos, se obtuvo un tiempo de inicio y de finalización de cada solicitud del usuario, del que se calculó la diferencia y se hizo un promedio. El menor tiempo de respuesta corresponde a la generación de la hoja de cálculo para el formato PEER y la de mayor tiempo, el guardado del formato TASK. En general, el tiempo de respuesta es bueno, variando la experiencia de usuario en el navegador, ya que, por ejemplo *Google Chrome* despliega los componentes conforme son cargados y *Mozilla Firefox* no.

Dado que los servidores actualmente responden a velocidades satisfactorias para los usuarios, se considera que la prueba en LAN proporciona una buena referencia para los resultados en producción, teniendo en cuenta que cada equipo, incluido el servidor, debía asignar parte de sus recursos para grabar los procesos ejecutados en el software.

Tabla 42. Tiempos de respuesta para cada solicitud en cada ordenador respectivo.

Tiempo inicio			Tiempo fin			Diferencia			Total segundos	Solicitud	PC	Prom.
MIN	SEG	30-DEC	MIN	SEG	30-DEC	MIN	SEG	30-DEC				
11	11	14	11	12	12	0	1	28	1,93333333	Actualizar formato SUMP	COMPAQ	
14	50	8	17	50	28	3	0	20	0,66666667	Actualizar formato SUMP	COMPUMAX	
10	54	4	10	55	5	0	1	1	1,03333333	Actualizar formato SUMQ	COMPAQ	
18	44	9	18	44	25	0	0	16	0,53333333	Actualizar formato SUMQ	COMPUMAX	1,04166667
10	36	20	10	37	12	0	1	22	1,73333333	Agregar riesgo	LENOVO	1,73333333
8	25	5	8	25	29	0	0	24	0,8	Asignar los roles	COMPUMAX	0,8
14	36	4	14	37	13	0	1	9	1,3	Ayuda y soporte	COMPAQ	
23	8	29	23	9	20	0	1	21	1,7	Ayuda y soporte	COMPUMAX	
23	42	29	23	43	26	0	1	27	1,9	Ayuda y soporte	COMPUMAX	
42	1	8	42	2	15	0	1	7	1,23333333	Ayuda y soporte	LENOVO	
42	58	6	42	58	23	0	0	17	0,56666667	Ayuda y soporte	LENOVO	
43	34	21	43	35	15	0	1	24	1,8	Ayuda y soporte	LENOVO	1,91666667
13	37	13	13	40	13	0	3	0	3	Cargar cumplimiento de metas	COMPAQ	3
8	32	0	8	35	7	0	3	7	3,23333333	Cargar definición de metas	COMPUMAX	3,23333333
12	35	29	12	41	6	0	6	7	6,23333333	Cargar formato PEER	COMPAQ	

33	47	15	33	50	4	0	3	19	3,63333333	Cargar formato PEER	LENOVO	
35	9	2	35	10	18	0	1	16	1,53333333	Cargar formato PEER	LENOVO	3,85833333
36	24	20	36	28	21	0	4	1	4,03333333	Cargar formato PEER	LENOVO	
2	41	16	2	45	15	0	4	29	4,96666667	Cargar formato STRAT	COMPAQ	4,96666667
11	1	24	11	7	15	0	6	21	6,7	Cargar formato SUMP	COMPAQ	
16	59	3	17	1	8	1	2	5	2,16666667	Cargar formato SUMP	COMPUMAX	4,31666667
10	44	5	10	50	19	0	6	14	6,46666667	Cargar formato SUMQ	COMPAQ	
13	55	2	17	59	10	4	4	8	4,26666667	Cargar formato SUMQ	COMPUMAX	5,36666667
4	33	19	4	38	13	0	5	24	5,8	Cargar formato TASK	COMPAQ	
10	10	14	10	15	2	0	5	18	5,6	Cargar formato TASK	COMPUMAX	
12	5	1	12	7	23	0	2	22	2,73333333	Cargar formato TASK	LENOVO	
18	36	8	18	41	1	0	5	23	5,76666667	Cargar formato TASK	LENOVO	6,375
1	54	22	1	59	10	0	5	18	5,6	Cargar metas	COMPAQ	
7	15	21	7	19	19	0	4	28	4,93333333	Cargar metas	LENOVO	
7	56	20	7	58	20	0	2	0	2	Cargar metas	LENOVO	
8	28	3	8	31	0	0	3	27	3,9	Cargar metas	LENOVO	4,50833333
14	25	5	14	26	23	0	1	18	1,6	Contribución general	COMPAQ	
23	6	19	23	7	6	0	1	17	1,56666667	Contribución general	COMPUMAX	
23	39	15	23	40	25	0	1	10	1,33333333	Contribución general	COMPUMAX	
41	56	2	41	58	1	0	2	29	2,96666667	Contribución general	LENOVO	
42	54	11	42	55	10	0	1	29	1,96666667	Contribución general	LENOVO	
43	29	13	43	30	1	0	1	18	1,6	Contribución general	LENOVO	1,83888889
3	5	25	3	6	16	0	1	21	1,7	Crear producto	COMPAQ	
9	13	6	9	13	27	0	0	21	0,7	Crear producto	LENOVO	1,2
38	7	3	38	10	14	0	3	11	3,36666667	Defectos inyectados	LENOVO	3,36666667
38	21	22	38	24	20	0	3	28	3,93333333	Defectos removidos	LENOVO	3,93333333
39	23	10	39	25	21	0	2	11	2,36666667	Defectos/Código	LENOVO	2,36666667
14	43	18	14	45	7	0	2	19	2,63333333	Desempeño	COMPAQ	
23	11	13	23	12	2	0	1	19	1,63333333	Desempeño	COMPUMAX	
23	46	21	23	47	25	0	1	4	1,13333333	Desempeño	COMPUMAX	
42	6	18	42	7	25	0	1	7	1,23333333	Desempeño	LENOVO	
43	2	15	43	2	28	0	0	13	0,43333333	Desempeño	LENOVO	
43	38	21	43	39	4	0	1	13	1,43333333	Desempeño	LENOVO	1,41666667
13	21	6	13	21	19	0	0	13	0,43333333	Exportar formato PEER	COMPAQ	0,43333333
3	48	10	3	49	17	0	1	7	1,23333333	Exportar hoja de cálculo formato STRAT	COMPAQ	
11	32	2	11	32	22	0	0	20	0,66666667	Exportar hoja de cálculo formato SUMQ	COMPAQ	
11	4	17	12	5	17	1	1	0	1	Exportar hoja de cálculo formato SUMP	COMPAQ	
20	14	25	20	15	26	0	1	1	1,03333333	Exportar hoja de cálculo formato LOGD (2 defectos)	COMPUMAX	
19	23	0	19	23	21	0	0	21	0,7	Exportar hoja de cálculo formato TASK	LENOVO	1,38
16	37	27	16	39	5	0	2	8	2,26666667	Generar formato WEEK	COMPUMAX	2,26666667

15	46	23	15	47	20	0	1	27	1,9	Guardar actividad	COMPUMAX	1,71666667
24	3	17	24	4	3	0	1	16	1,53333333	Guardar actividad	LENOVO	
8	44	6	8	44	28	0	0	22	0,73333333	Guardar asignación de metas de proyecto	COMPUMAX	0,73333333
19	55	29	19	56	28	0	1	29	1,96666667	Guardar defecto	COMPUMAX	1,6
25	32	5	25	33	12	0	1	7	1,23333333	Guardar defecto	LENOVO	
13	15	16	13	16	2	0	1	16	1,53333333	Guardar formato PEER	COMPAQ	1,08333333
21	28	6	21	28	21	0	0	15	0,5	Guardar formato PEER	COMPUMAX	
22	31	16	22	31	21	0	0	5	0,16666667	Guardar formato PEER	COMPUMAX	
34	41	13	34	42	2	0	1	19	1,63333333	Guardar formato PEER	LENOVO	
35	48	17	35	49	17	0	1	0	1	Guardar formato PEER	LENOVO	
37	5	12	37	6	2	0	1	20	1,66666667	Guardar formato PEER	LENOVO	
3	37	5	3	38	13	0	1	8	1,26666667	Guardar formato STRAT	COMPAQ	1,26666667
9	44	4	9	45	12	0	1	8	1,26666667	Guardar formato STRAT	LENOVO	
5	47	16	5	55	23	0	8	7	8,23333333	Guardar formato TASK	COMPAQ	5,8952381
12	35	12	12	37	29	0	2	17	2,56666667	Guardar formato TASK	COMPUMAX	
17	59	14	18	3	0	1	4	16	4,53333333	Guardar formato TASK	LENOVO	
19	1	11	19	8	28	0	7	17	7,56666667	Guardar formato TASK	LENOVO	
5	47	16	5	55	23	0	8	7	8,23333333	Guardar formato TASK	COMPAQ	
12	35	12	12	37	29	0	2	17	2,56666667	Guardar formato TASK	COMPUMAX	
19	1	11	19	8	28	0	7	17	7,56666667	Guardar formato TASK	LENOVO	
13	52	6	13	52	22	0	0	16	0,53333333	Guardar logros de metas	COMPAQ	0,88888889
21	39	0	21	40	2	0	1	2	1,06666667	Guardar logros de metas	COMPUMAX	
22	41	16	22	41	26	0	0	10	0,33333333	Guardar logros de metas	COMPUMAX	
34	56	17	34	56	28	0	0	11	0,36666667	Guardar logros de metas	LENOVO	
36	1	24	36	2	10	0	1	16	1,53333333	Guardar logros de metas	LENOVO	
37	21	24	37	22	9	0	1	15	1,5	Guardar logros de metas	LENOVO	
2	2	13	2	4	1	0	2	18	2,6	Guardar metas	COMPAQ	1,78333333
9	32	26	9	33	21	0	1	25	1,83333333	Guardar metas	COMPUMAX	
7	22	14	7	23	17	0	1	3	1,1	Guardar metas	LENOVO	
8	0	23	8	1	11	0	1	18	1,6	Guardar metas	LENOVO	
6	33	14	6	34	12	0	1	28	1,93333333	Iniciar actividad	COMPAQ	1,75
20	2	16	20	3	3	0	1	17	1,56666667	Iniciar actividad	LENOVO	
8	50	5	8	52	28	0	2	23	2,76666667	Ir a la siguiente fase/ciclo	LENOVO	2,15238095
10	59	3	11	0	26	1	1	23	1,76666667	Ir a la siguiente fase/ciclo	LENOVO	
25	53	28	25	56	3	0	3	5	3,16666667	Ir a la siguiente fase/ciclo	LENOVO	
26	2	19	26	4	24	0	2	5	2,16666667	Ir a la siguiente fase/ciclo	LENOVO	
26	11	0	26	12	4	0	1	4	1,13333333	Ir a la siguiente fase/ciclo	LENOVO	
26	18	23	26	20	13	0	2	20	2,66666667	Ir a la siguiente fase/ciclo	LENOVO	
26	26	16	26	27	28	0	1	12	1,4	Ir a la siguiente fase/ciclo	LENOVO	
39	13	4	39	15	14	0	2	10	2,33333333	Logro de metas	LENOVO	2,33333333
15	42	19	15	43	7	0	1	18	1,6	Parar actividad	COMPUMAX	1,58333333
23	56	14	23	57	1	0	1	17	1,56666667	Parar actividad	LENOVO	

38	34	4	38	35	24	0	1	20	1,66666667	PDF en compilación	LENOVO	
38	38	26	38	39	23	0	1	27	1,9	PDF en prueba de unidad	LENOVO	
38	42	22	38	43	11	0	1	19	1,63333333	PDF construcción e integración	LENOVO	
38	45	19	33	46	9	-5	1	20	1,66666667	PDF en prueba de sistema	LENOVO	1,71666667
39	35	25	39	36	9	0	1	14	1,46666667	Rendimientos de fase	LENOVO	1,46666667
21	36	15	21	37	21	0	1	6	1,2	Solicitar interrupción prevista	LENOVO	1,2
37	55	18	37	58	8	0	3	20	3,66666667	Tamaño del producto	LENOVO	3,66666667
39	49	19	39	51	18	0	2	29	2,96666667	Tasa de inyección de defectos	LENOVO	
40	0	23	40	3	18	0	3	25	3,83333333	Tasa de remoción de defectos	LENOVO	3,4
38	49	13	38	51	18	0	2	5	2,16666667	Tiempo de arreglo de defectos	LENOVO	2,16666667
37	32	2	37	34	24	0	2	22	2,73333333	Tiempo en etapas	LENOVO	2,73333333
14	13	3	14	25	24	0	12	21	12,7	Tiempo en proyectos	COMPAQ	
22	59	4	22	59	21	0	0	17	0,56666667	Tiempo en proyectos	COMPUMAX	
23	28	23	23	29	1	0	1	8	1,26666667	Tiempo en proyectos	COMPUMAX	
41	43	6	41	44	28	0	1	22	1,73333333	Tiempo en proyectos	LENOVO	
42	45	8	42	46	18	0	1	10	1,33333333	Tiempo en proyectos	LENOVO	
43	18	3	43	19	15	0	1	12	1,4	Tiempo en proyectos	LENOVO	3,16666667
14	18	21	14	20	3	0	2	12	2,4	Trabajo y dificultad	COMPAQ	
23	1	28	23	2	14	0	1	16	1,53333333	Trabajo y dificultad	COMPUMAX	
23	32	18	23	32	29	0	0	11	0,36666667	Trabajo y dificultad	COMPUMAX	
41	49	23	41	51	16	0	2	23	2,76666667	Trabajo y dificultad	LENOVO	
42	49	19	42	50	13	0	1	24	1,8	Trabajo y dificultad	LENOVO	
43	21	19	43	22	16	0	1	27	1,9	Trabajo y dificultad	LENOVO	1,79444444
39	3	13	39	5	12	0	2	29	2,96666667	Valor planeado contra valor ganado	LENOVO	2,96666667

2.2.1.2 Usabilidad

Para medir la usabilidad se persiguieron los parámetros correspondientes al artículo de Solarte[1]. De acuerdo a dicho documento, se construyó la tabla 43 donde se muestra la descripción de los parámetros, el valor asignado para cada uno de ellos, el puntaje para el prototipo y el motivo de dicho puntaje.

Tabla 43. Puntaje obtenido en cada criterio de usabilidad basado en el artículo de Solarte.

Nombre	Descripción	Impacto (%)	Puntaje (%)	Motivo
Usabilidad	Realiza tareas de una forma fácil y eficiente	10	9	El prototipo resuelve gran parte de las necesidades de un equipo de desarrollo de software orientado con la metodología TSP.
Aspecto psicológico	Es atractivo para los clientes evitando aspectos técnicos y generar interés en el mismo.	8	7	Conserva colores *suaves* y tendencias dentro del software, sin embargo el interés puede no ser el esperado.
Accesibilidad	Poder ser usado por el mayor número de personas, indiferentemente de las limitaciones del individuo	10	9	Permite el ingreso a todas las personas sin una jerarquía establecida en el software, sin embargo, puede ser limitado para clientes sin la tecnología suficiente para ellos. Ver apartado 2.2.2.2 de compatibilidad.

[1] Solarte, G., Ocampo, CA. (2010). Usabilidad en Espacio Web. Universidad Tecnológica de Pereira. Scientia et Technica. pp. 251-256

Aspectos técnicos y estáticos	Sin exceso de texto, destacar títulos y organizar la información mediante tablas.	9	9	Se muestra exactamente lo que el usuario necesita, con la ayuda suficiente, sin exceder las medidas de control.
Facilidad de uso e instalación	Agradable de usar, fácil de usar, autoexplicativo.	12	12	Es muy amigable e intuitivo.
Calidad y estructura de los contenidos	Ortografía y frases concretas; no discriminativas	32	32	Palabras correctamente escritas, respetando mayúsculas y sin hacer descripciones asociadas a la discriminación.
Eficiencia y tiempo de desgaste	Eficiencia y tiempo de descarga	10	7	Responde de acuerdo a las necesidades de los usuarios, sin embargo en algunos casos puede tardar un poco. Ver apartado 2.2.1.1 de rendimiento.
Seguridad	Seguridad	9	7	Brinda la seguridad ofrecida por la tecnología, así como encripción MD5. Sin embargo el alcance el proyecto no se centra en garantizar esta característica.
	Total	100	92	

2.2.2 Operacionales

2.2.2.1 Compatibilidad

El sistema es compatible sobre cualquier sistema operativo al soportarse en Java, una tecnología multiplataforma, sin embargo, el desarrollo se llevó a cabo sobre JDK 1.8.0_91, tecnología cuyo desempeño es el adecuado a partir de los sistemas operativos mencionados en la tabla 44. En dicha tabla también se mencionan los navegadores con los que se tiene compatibilidad, pues *Mozilla Firefox* y *Google Chrome* fueron los principales ambientes de prueba, aunque *Internet Explorer* también fue probado y a pesar de brindar todas las funcionalidades esperadas, los íconos del front-end no eran mostrados por razones inconclusas.

Un elemento adicional, igualmente mostrado en la tabla 44, es el plugin Adobe Flash Player, que en caso de que el usuario no disponga de esta tecnología, la cartilla de TSP es sustituida por un link para su descarga.

Tabla 44. Compatibilidad de la herramienta prototipo.

Criterio de compatibilidad	**Tecnología**
Sistema operativo	Windows XP Professional SP3
	Windows Vista SP2 (Home Premium en adelante)
	Windows 7 (Home Premium en adelante)
	Centos 5
	Fedora 17
	Ubuntu 12,04
	Mac OS X 10,7
	Android 4,4,2
Navegador	Google Chrome
	Mozilla Firefox
Plugins	Adobe Flash Player

Con base en los resultados obtenidos, y el impacto de los tres criterios de compatibilidad en la actualidad, se considera un acierto del 84% del total esperado.

2.2.2.1.1 Internacionalización

La tabla 45 muestra los archivos que deben ser modificados para cambiar el software de idioma y el costo estimado de modificación entre 1 y 10. El total de archivos es de 33, de los que 30 provienen de los mensajes de excepción, cuyo costo no es el más elevado puesto que la identificación de este contenido es rápidamente modificable, sin traer implicaciones mayores al software.

Con base en los resultados obtenidos en la prueba de internacionalización, se considera un logro del 90% por la facilidad de cambio, pero con cierto grado de dificultad, dado el número de archivos involucrados en la operación.

Tabla 45. Modificación de archivos para cambio de idioma.

Tipo de modificación	Número de archivos	Costo	Descripción
Validaciones de datos en el cliente	1	3	Únicamente el archivo script.js que posee cadenas simples que controlan el diligenciamiento de datos.
Mensajes de excepción	30	2	Identificar rápidamente todas las excepciones ExceptionWarn y ExceptionFatal del sistema y sustituir por el idioma requerido.
Texto en el cliente	1	1	Únicamente el archivo de mensajes.properties.
Datos de sistema	1	1	Archivo TSP.java con los elementos definidos formalmente en la metodología.

MANUAL DE INSTALACIÓN DE LA APLICACIÓN INFORMÁTICA "TSPSUPPORT", PROTOTIPO WEB PARA LA GESTIÓN DE MÉTRICAS COMO SOPORTE A LA METODOLOGÍA DE DESARROLLO TSP (TEAM SOFTWARE PROCESS ®)

Presentado por

JUAN CAMILO CAMACHO BELTRÁN



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2017

Tabla de contenido

1. REQUERIMIENTOS TECNOLÓGICOS 264

1.1 Requerimientos del cliente 264

1.1.1 Requerimientos software 264

1.1.2 Requerimientos hardware 264

1.2.1 Requerimientos software 265

1.2.2 Requerimientos hardware 265

2. SCRIPTS DEL MODELO RELACIONAL 266

4. EJEMPLO DE INSTALACIÓN 266

4.1 Creación del usuario en el motor relacional 267

4.2 Creación de tablas y relaciones. 267

4.3 Ejecución del script de triggers 268

4.4 Privilegios al usuario de conexión de la aplicación. 268

4.5 Despliegue del servidor. 269

AUDIENCIA

El siguiente documento va dirigido a instructores y estudiantes de la metodología TSP, así como a cualquier equipo de desarrollo de software interesado tanto en la metodología como en el uso de la herramienta TSPSupport.

1. REQUERIMIENTOS TECNOLÓGICOS

TSPSupport es una aplicación web desarrollada en Java bajo el framework JSF en colaboración con Primefaces, Hibernate, Jongo, del lado del servidor, y Materialize, Jquery y CSS del lado del cliente.

Con base en estas tecnologías software mencionadas, se determinaron tanto los requerimientos del cliente como del servidor, en hardware y software, los cuales serán mencionados a continuación.

1.1 Requerimientos del cliente

1.1.1 Requerimientos software

El usuario que desee hacer uso de la aplicación necesita, a nivel de software, de un sistema operativo capaz de soportar navegadores web, suficientes para ejecutar la librería JQuery en su versión 1.11.3, entre los cuales se recomienda Google Chrome y Mozilla Firefox.

A la fecha, Google Chrome se encuentra en su versión 58.0.3029.110 (a 64 bits), la que ha brindado una mejor experiencia de usuario. Por su parte, Mozilla Firefox en su actual versión 53.0.2 (a 32 bits) también mejoró la experiencia frente a versiones anteriores. Estas dos versiones son las recomendadas para el uso de la aplicación, sin embargo versiones anteriores también brindan la posibilidad de ejecutar proyectos sin diferencias significativas.

Asimismo no se recomienda hacer uso del navegador Internet Explorer dado que los íconos no se despliegan, sin embargo todas las funcionalidades de la aplicación TSPSupport se ejecutan correctamente.

1.1.2 Requerimientos hardware

Para el buen desempeño del software mencionado, el usuario debe disponer de una máquina tecnológica cuyos requisitos mínimos satisfagan las necesidades mínimas de oficina.

Con las pruebas realizadas, se demostró que un ordenador de procesador Intel ATOM puede suplir las necesidades, pues todos los servicios están a disposición del cliente. Sin

embargo, el desempeño decae significativamente, ya que, aunque los datos son entregados al navegador, el sistema tarda en hacer un renderizado de los componentes y su información contenida.

Por tanto, se afirma que el cliente debe disponer, como mínimo, de un equipo con un procesador equivalente a AMD E1 1500 APU o Intel Celeron N2830.

1.2 Requerimientos del servidor

1.2.1 Requerimientos software

A continuación se listan las tecnologías software requeridas para el correcto funcionamiento del prototipo.

1.1 Motor de base de datos documental Mongo. Probado en las versiones 2.6.1 y 3.2.9, siendo esta última versión la utilizada para el despliegue de la aplicación en el servidor de la comunidad educativa.

1.2 Paquete de desarrollo Java JDK. Desarrollado sobre la versión 1.8.101. Se hicieron pruebas sobre versiones anteriores que dieron resultados negativos. Se deduce que con versiones posteriores (1.8.x), todos los servicios de la aplicación seguirán en funcionamiento.

1.3 Motor de base de datos relacional, PostgreSQL. Desarrollado sobre la versión 9.3 y desplegado sobre la versión 9.1.

1.4 Apache Tomcat. En su versión 8.5.12, en la que se desplegó al servicio de la comunidad.

1.5 Sistema operativo Windows. La aplicación está configurada para desplegarse en un sistema operativo Windows, que lógicamente corresponde a una distribución server, por ejemplo 2008 o 2012.

1.2.2 Requerimientos hardware

El prototipo no es muy exigente con este requerimiento. La capacidad del ordenador depende de a quién va dirigida la herramienta y cuántos usuarios realizarán transacciones a diario sobre ésta. Por tanto, se afirma que un procesador core i7 o un Xeon de cualquier referencia, puede ser suficiente para una modesta comunidad de usuarios.

2. SCRIPTS DEL MODELO RELACIONAL

En el motor de base de datos relacional se debe crear la base de datos tsp con el usuario de la aplicación usuario_tsp de contraseña TSPSupport2017**. Posteriormente se deben ejecutar los scripts del modelo, triggers y privilegios (anexos al documento) para crear las tablas (y sus respectivas relaciones), algunas reglas adicionales de integridad y establecer los privilegios del usuario para la aplicación.

3. INTERNACIONALIZACIÓN

Para cambiar la aplicación de idioma, se deben cambiar tres archivos de la aplicación, o hacer referencia a ellos.

- mensajes_en.properties. Archivo de propiedades con todo el diccionario del front-end de la aplicación.
- TSP.java. Archivo de servidor con todo el diccionario del back-end de la aplicación..
- script.js. Lógica del lado del cliente con algunos mensajes en un lenguaje determinado.

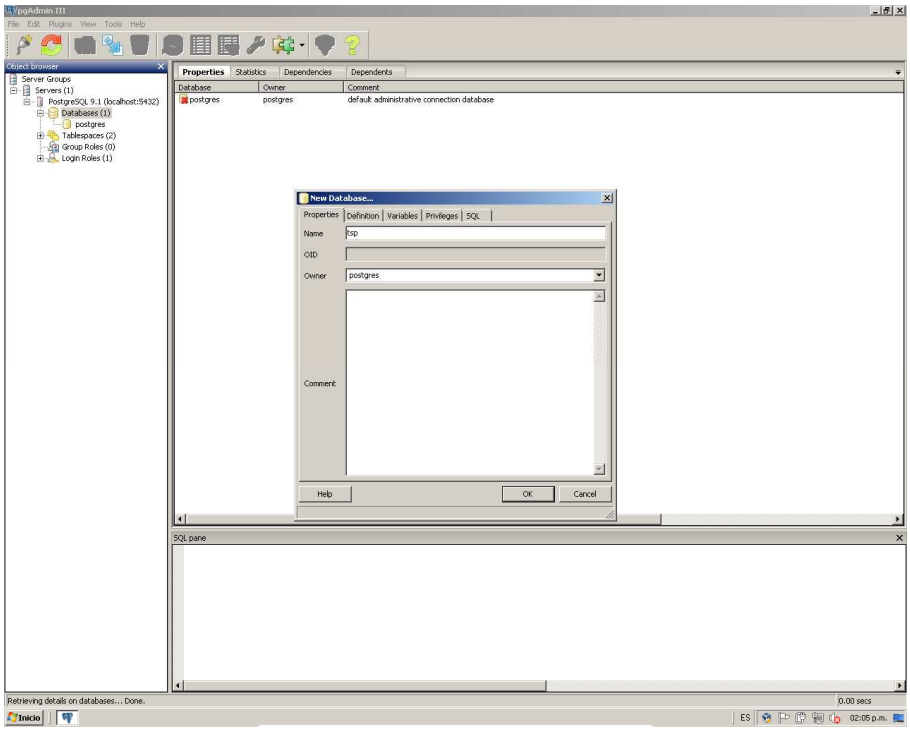
4. VERSIONES

La última versión programa para este prototipo corresponde a la 10, la cual compone todos los formatos automatizados de TSP, la gestión del proyecto, recolección de indicadores, métricas y graficación de resultados, todo ello bajo las correcciones obtenidas a partir de las diferentes pruebas realizadas.

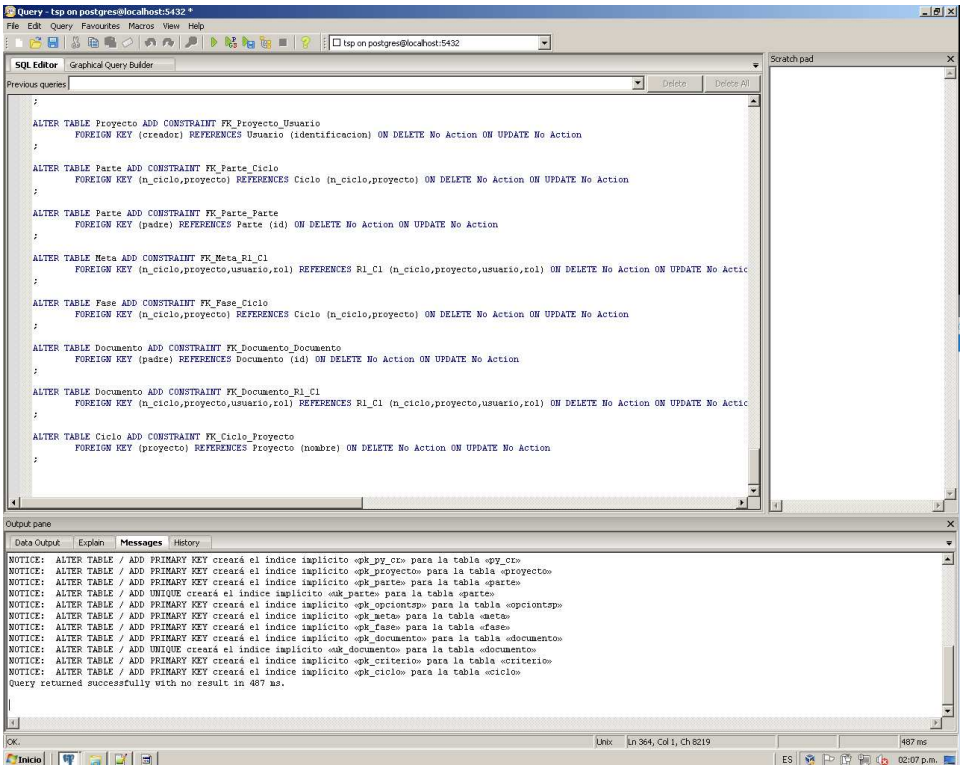
5. EJEMPLO DE INSTALACIÓN

Con base en las descripciones de los puntos anteriores, se ejecutan las siguientes acciones, una vez se dispone de las tecnologías requeridas en el servidor.

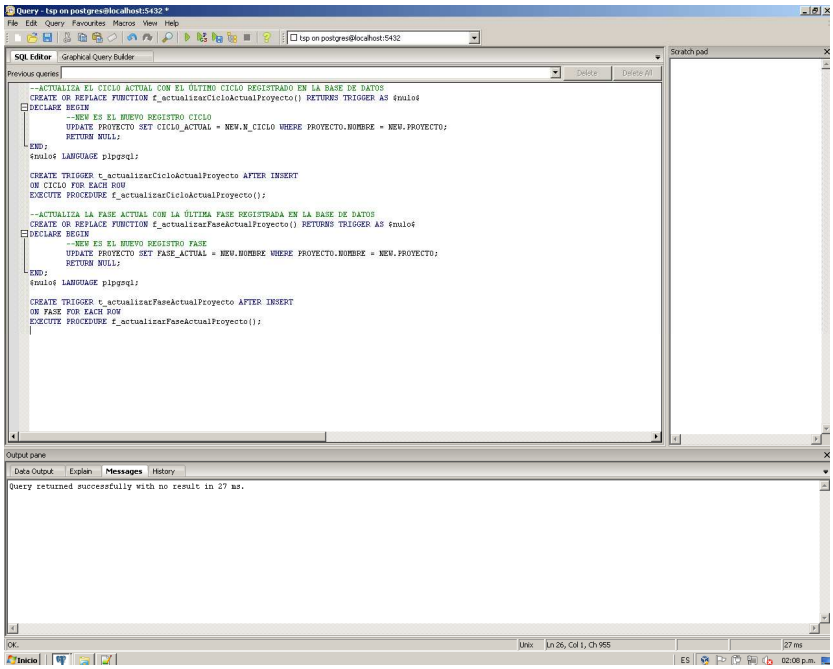
5.1 Creación del usuario en el motor relacional



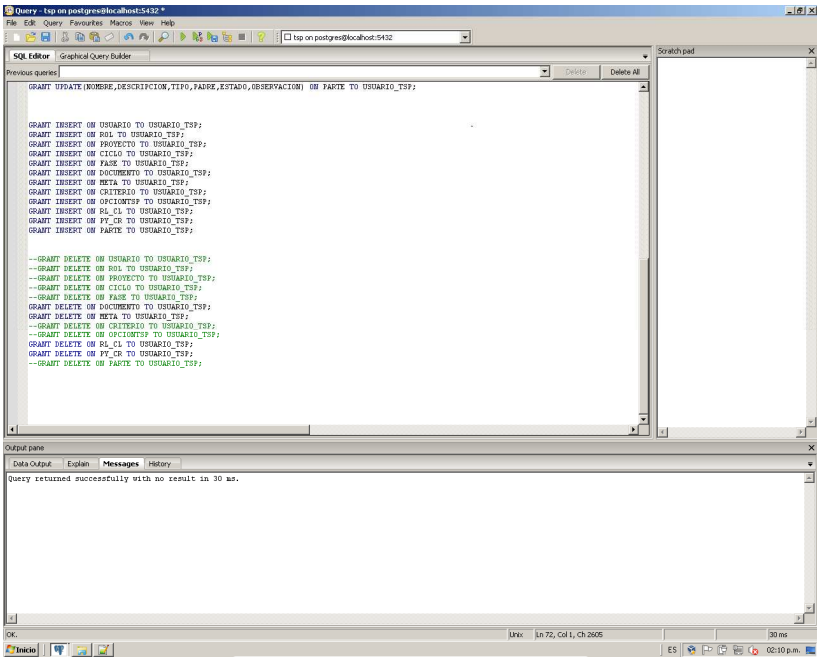
5.2 Creación de tablas y relaciones.



5.3 Ejecución del script de triggers



5.4 Privilegios al usuario de conexión de la aplicación.

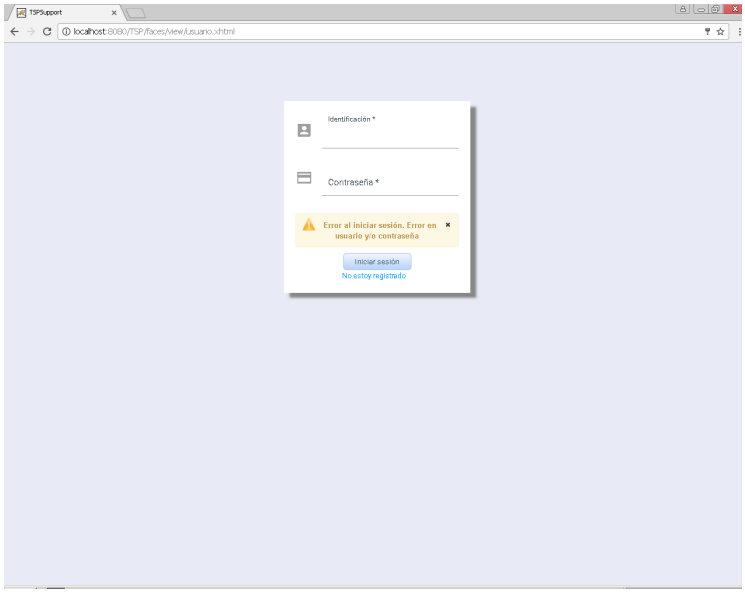


Si este script falla, se debe verificar la existencia de los roles en el motor de base de datos Postgres, pues la configuración del script considera que existe un usuario *admin_tsp*.

De igual forma, el fallo no esperado sobre los scripts requiere una evaluación de su contenido y la ejecución completa de la transacción para asegurar la correcta implantación del modelo de datos. De persistir la falla, repetir todos los pasos, eliminando la base de datos y todas las características establecidas hasta el momento del error.

5.5 Despliegue del servidor.

6. Finalmente desplegar el aplicativo TSP en el servidor Tomcat instalado en el servidor (copiando y pegando el archivo TSP.war en el directorio correspondiente) e iniciando el servicio web. Para comprobar el correcto despliegue de la aplicación se accede a la dirección web http://localhost:[puerto-servidor]/TSP



Información de contacto

alconi@gmail.com

jccamachob94@gmail.com



MANUAL DE USUARIO DE LA APLICACIÓN INFORMÁTICA "TSPSUPPORT", PROTOTIPO WEB PARA LA GESTIÓN DE MÉTRICAS COMO SOPORTE A LA METODOLOGÍA DE DESARROLLO TSP (TEAM SOFTWARE PROCESS ®)

Presentado por

JUAN CAMILO CAMACHO BELTRÁN



UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2017

Tabla de contenido

1. PROPÓSITO 274

2. DESCRIPCIÓN GENERAL 275

2.1 Primera interfaz 275

2.2 Interfaz general 276

2.3 Colores 276

2.3.1 Mensajes 276

2.3.2 Etapas TSP 277

2.4 Características responsivas 277

2.5 Acceso rápido a los formatos 278

3. DESARROLLO DE LAS FUNCIONALIDADES 279

3.1 Creación de proyecto 279

3.2 Asignación de roles 282

3.3 Definición de metas 282

3.4 Estrategia 284

3.5 Planeación 286

3.6 Registro de tiempo 289

3.7 Directorio del proyecto 294

3.8 Reporte semanal 294

3.9 Formatos de resumen 295

3.10 Otros formatos 298

3.10.1 Registro de defectos 298

3.10.2 Formato PEER 299

3.10.3 Formato CCR 299

3.10.4 Formato PIP 300

3.11 Comparador de código 301
3.12 Exportador de datos 301
3.13 Criterios de salida de fase 301
3.13.1 Fase de Lanzamiento 302
3.13.2 Fase de Estrategia 302
3.13.3 Fase de Planeación 302
3.13.4 Fase de Requerimientos 302
3.13.5 Fase de Diseño 302
3.13.6 Fase de Implementación 303
3.13.7 Fase de Pruebas 303
3.13.8 Fase de Postmortem 303
3.14 Resultados 303
3.14.1 Logro de metas 303
3.14.2 Resultados cualitativos 304

AUDIENCIA

El siguiente documento va dirigido a instructores y estudiantes de la metodología TSP, así como a cualquier equipo de desarrollo de software interesado tanto en la metodología como en el uso de la herramienta TSPSupport.

1. PROPÓSITO

Facilitar la ejecución de proyectos de desarrollo de software a través de la metodología TSP, recolectando métricas automáticamente, sin requerir un gran esfuerzo de los ingenieros por gestionar dicha información.

Con la automatización del proceso, el equipo TSP podrá retroalimentar su propio proceso de desarrollo, todo esto gracias a la información precisa que recolecta la herramienta de forma amigable e intuitiva.

Con este documento se busca guiar a los usuarios en el proceso a través del software, para que conozcan las funcionalidades que ofrece, y por tanto, puedan ejecutar las tareas de la metodología TSP con ayuda de la herramienta.

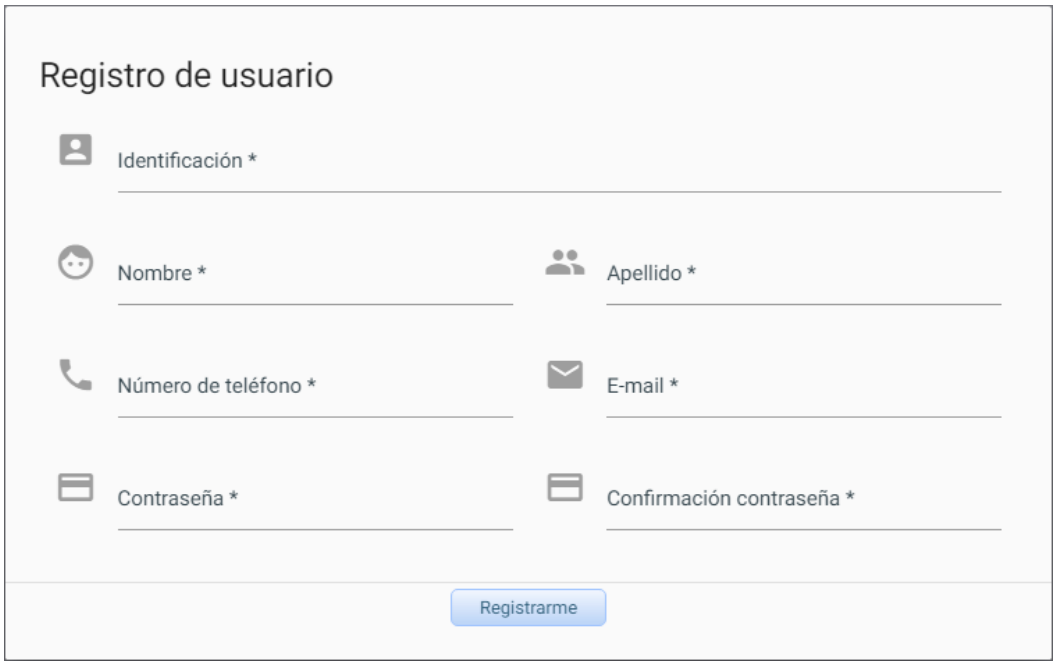
2. DESCRIPCIÓN GENERAL

2.1 Primera interfaz

Al entrar a la aplicación por primera vez (actualmente disponible en http://45.55.74.36:8080/tsp), ésta mostrará un formulario de inicio de sesión, solicitando identificación y contraseña del usuario, a fin de acceder a los servicios de la herramienta. A continuación se muestra dicha interfaz.



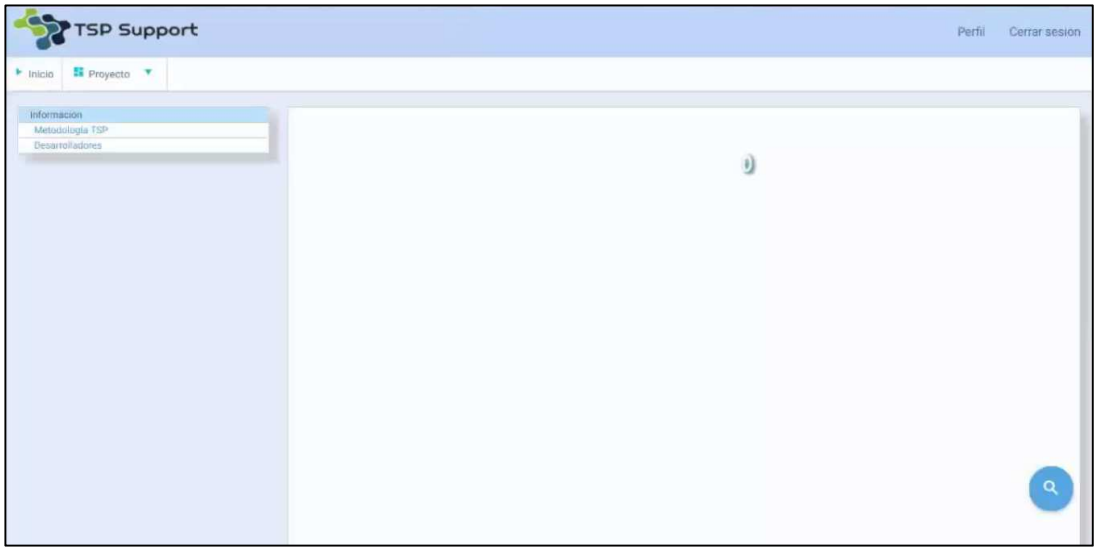
En caso de no estar registrado en el sistema, se puede proceder al respectivo registro dando click en el link *No estoy registrado,* lo que mostrará la siguiente ventana, donde se deben diligenciar todos los datos y dar click al botón *Registrarme*.



2.2 Interfaz general

Todas las interfaces de la aplicación se fundamentan en la estructura mostrada en la siguiente imagen. Se muestran dos paneles superiores: el primer panel brinda acceso a servicios relacionados con la cuenta del usuario, en este caso: *Perfil* y *Cerrar Sesión.* El segundo panel brinda los servicios generales como *Inicio* y *Proyecto.*

Luego de los paneles, se localizan dos secciones. La primera sección corresponde al menú específico de un proyecto en particular (creado en la aplicación) y la segunda sección muestra el contenido de la página, por ejemplo formatos TSP, el formulario de creación de proyecto, selección de roles, entre otros.



Con la creación de un proyecto de desarrollo de software en la herramienta, la sección de contenido mostrará un panel superior que indicará los detalles actuales del proyecto y los roles del usuario en sesión, para el ciclo actual.

2.3 Colores

3. *2.3.1 Mensajes*

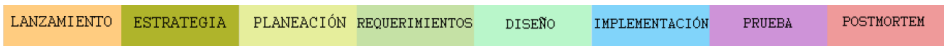
La aplicación cuenta con tres tipos de mensajes para dar a conocer al usuario el resultado de las solicitudes. Estos mensajes se localizan en la parte superior de la aplicación y se muestran cada vez que la aplicación da respuesta a una necesidad. La siguiente figura muestra un ejemplo del mensaje, en caso de ser exitosa la ejecución de la solicitud.



Por tanto, el color azul se asocia al tipo de mensaje exitoso. Los otros dos tipos de mensajes corresponden al color amarillo y rojo, donde el primero indica un error pequeño en algún dato o privilegio, en cambio el color rojo evidencia un error fatal que no era esperado o que irrumpe la integridad de la información de la aplicación.

4. *2.3.2 Etapas TSP*

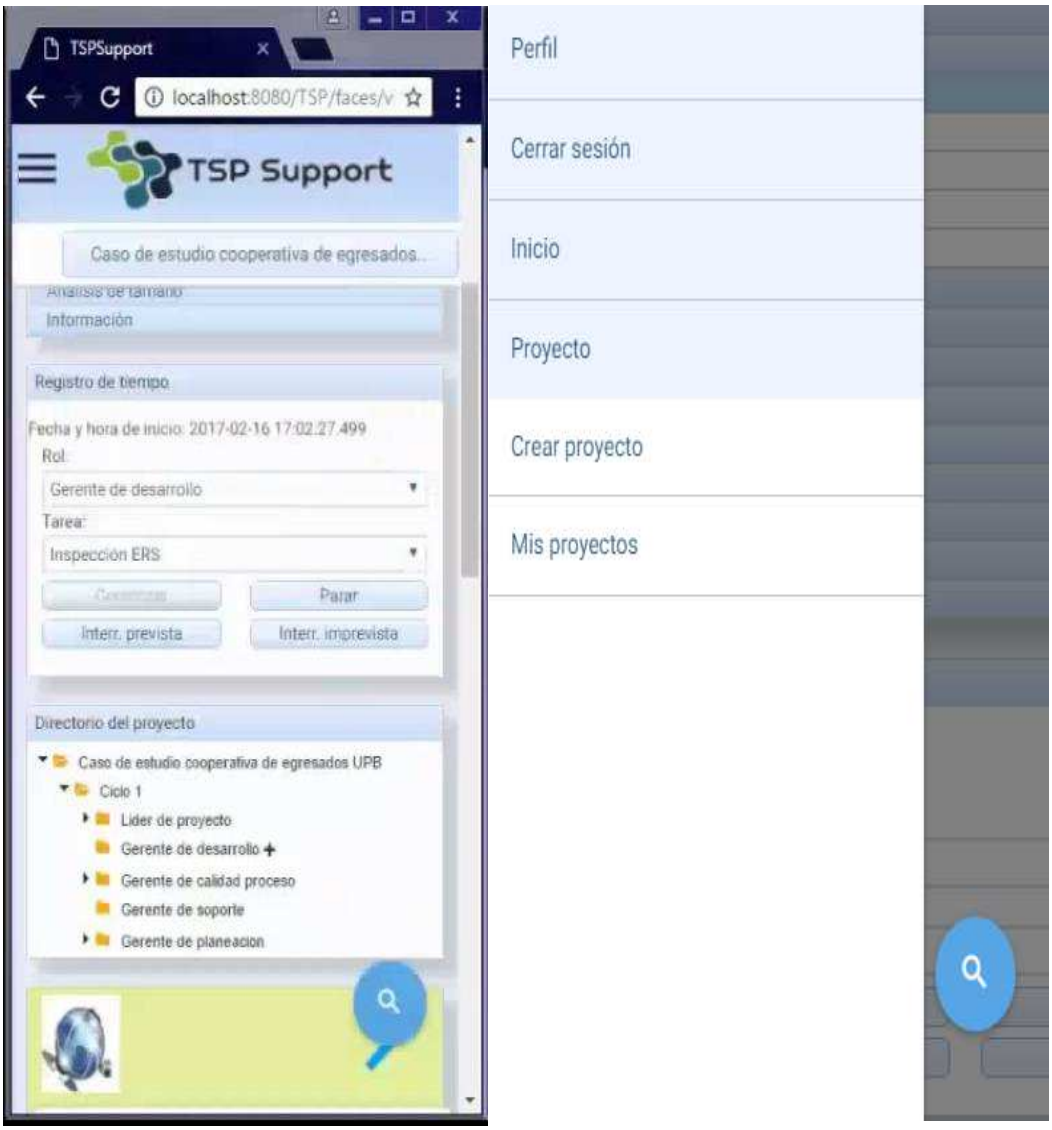
La siguiente figura muestra la distribución de colores para cada etapa de cada ciclo de un proyecto en específico. Esto ayudará a identificar rápidamente la fase actual del proyecto, es decir, la etapa sobre la que se está trabajando.



4.1 Características responsivas[2]

La herramienta posee componentes responsivos para facilitar su utilización en diferentes pantallas, sin embargo existen limitaciones sobre el tamaño de las mismas, pues algunos formatos son demasiado amplios, incluso para pantallas corrientes y se vuelve un requerimiento el uso de un ordenador para esos casos. El control de tiempos por ejemplo, es uno de los componentes que inclusive se puede utilizar desde un celular o Tablet.

[2] Responsivo:



En el lado izquierdo de la imagen se puede observar cómo se despliega la aplicación en la menor resolución, mostrando que el menú se ajusta correctamente a las necesidades de registro de tiempo, documentación, entre otros. El menú de los paneles se despliega verticalmente, bajo la misma jerarquía del menú horizontal.

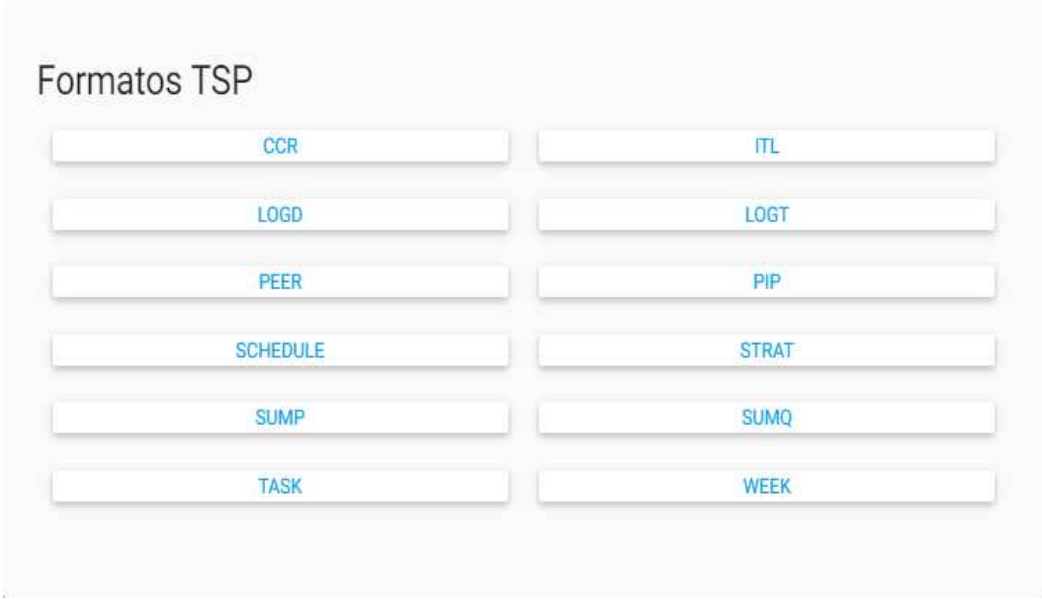
4.2 Acceso rápido a los formatos

Puesto que gran parte de la metodología se fundamenta en la utilización de los formatos, la herramienta brinda dos menús de acceso rápido a los formatos: los formatos guía y los formatos de gestión.

Los formatos guía se localizan en el menú del proyecto en la sección de *Metodología TSP* donde se localizan los formatos mostrados en la siguiente figura.



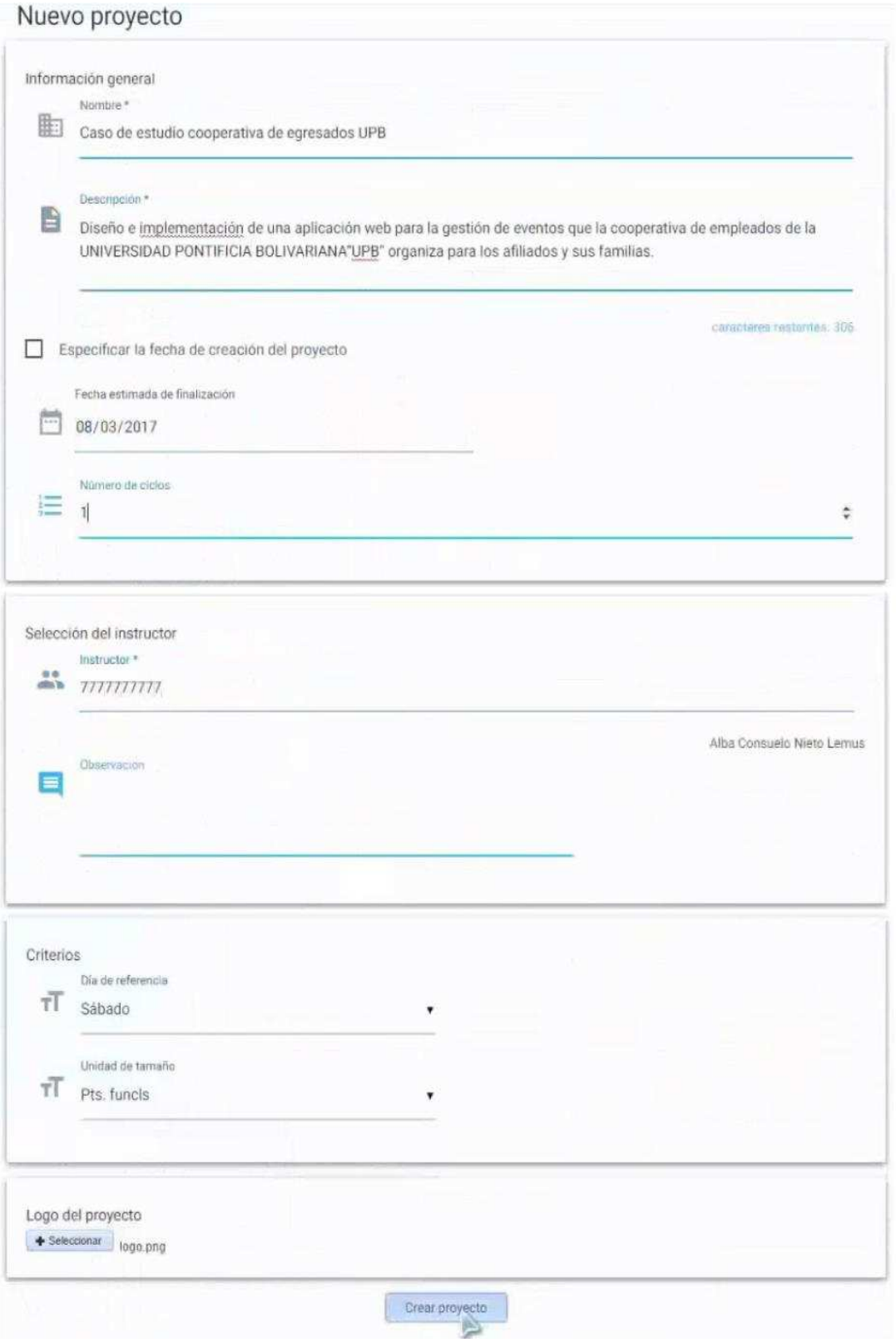
Los formatos de gestión se encuentran en el ícono inferior derecho, bajo el ícono de documento, el cual despliega la siguiente ventana



5. DESARROLLO DE LAS FUNCIONALIDADES

5.1 Creación de proyecto

Para crear un proyecto, se da click sobre el enlace superior *Proyecto* el cual desplegará dos opciones: *Mis proyectos y Crear proyecto.* Se elige la opción de crear proyecto y se diligencia el siguiente formulario.

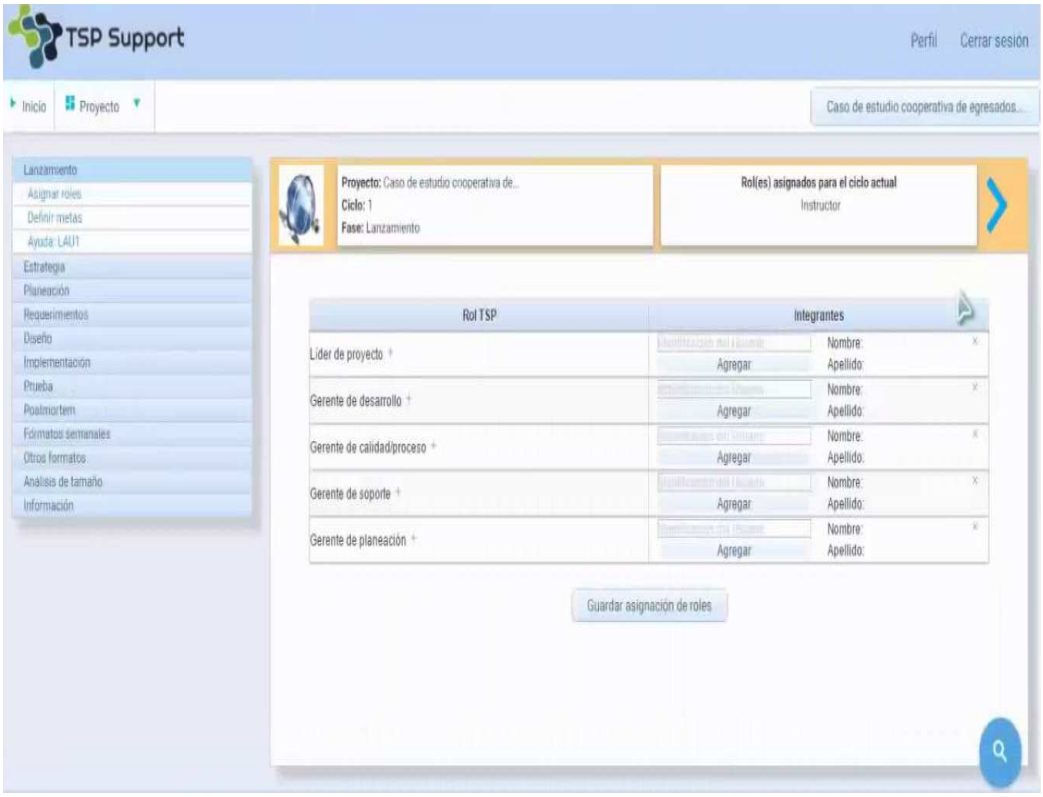


En este formulario debe diligenciarse el nombre del proyecto, una breve descripción, fecha estimada de finalización y número de ciclos a llevar a cabo.

Posteriormente se especifica el instructor de proyecto, y los criterios como el día de la semana en el que se va evaluar el progreso, así como la unidad de tamaño para la medición de las métricas. Finalmente se elige un logo para el proyecto para identificarlo rápidamente.

En caso de que el proyecto haya iniciado con anterioridad a la creación de éste sobre el software, se puede elegir la opción de especificar la fecha de creación, el cual mostrará la entrada de fecha de inicio del proyecto, en caso contrario, la fecha de inicio será la misma fecha de creación del proyecto.

Una vez todos los datos son diligenciados, se solicita la creación dando click en el botón *Crear proyecto*. Esta acción hará cambios significativos en la interfaz: podrá ver todos los servicios del menú del proyecto, la información del estado actual del proyecto en la parte superior del contenido y un botón en el segundo panel de la página, con el nombre del proyecto, que al darle click mostrará todos los proyectos en los que se encuentra, el usuario en sesión, involucrado.



5.2 Asignación de roles

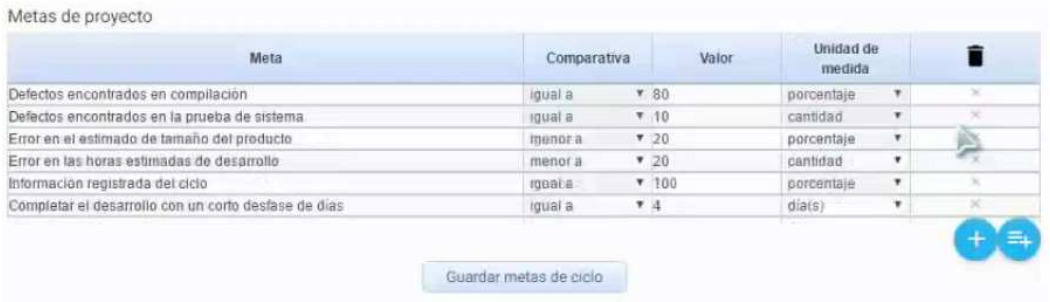
De acuerdo a la metodología TSP, el instructor evalúa la personalidad, experiencia e interés de los integrantes del equipo, para así aprobar la asignación de dichos roles. Es por esto que el software delega al instructor la asignación de los roles a los integrantes del equipo, a través de la siguiente interfaz, en la que puede asignar varios integrantes por rol, o varios roles por integrante.



Una vez establecidos los roles, se guarda dicha asignación dando click en el botón *Guardar asignación de roles*. En caso de requerir algún cambio, hay que tener en cuenta que sólo en la fase de lanzamiento, antes del establecimiento de las metas del ciclo, es posible hacer la reasignación accediendo a este menú y modificando los valores de la misma forma.

5.3 Definición de metas

Con los roles bien definidos, se establecen las metas para cada rol y proyecto en general. Puesto que la metodología TSP brinda metas recomendadas, el software provee a los usuarios la posibilidad de crear sus propias metas, añadiendo o eliminando las metas recomendadas, o creando nuevas metas al gusto.

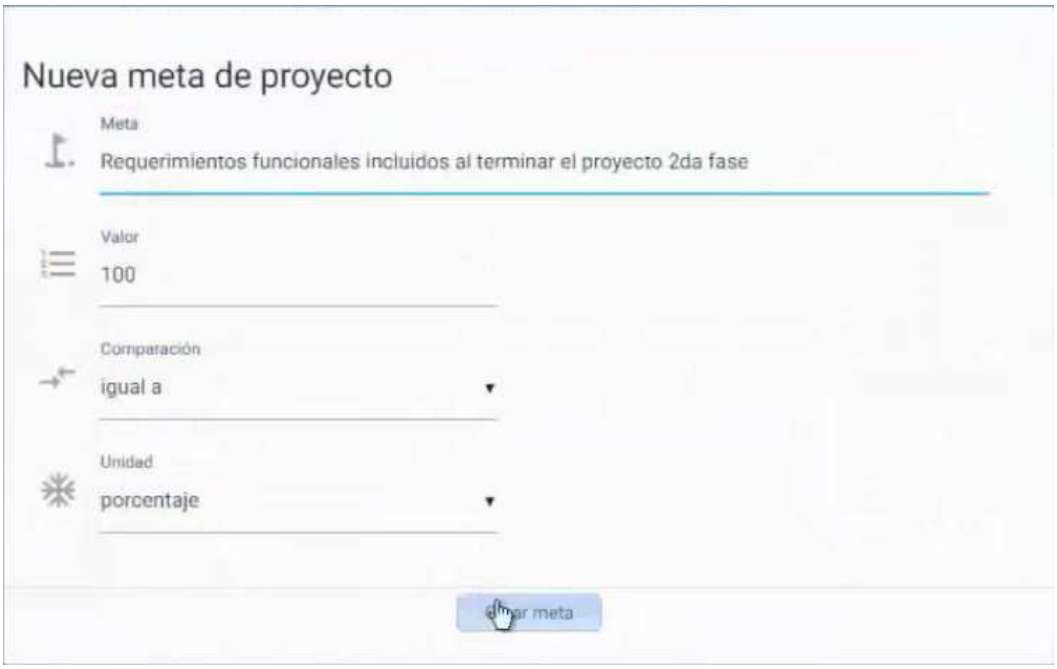


Como se observa en la imagen, la aplicación brinda dos íconos para agregar metas. El primero de ellos muestra un menú completo sobre una nueva meta definida por

el usuario, en cambio el segundo ícono despliega las metas predefinidas por la metodología para añadirse en caso de considerarlo necesario.

Cabe añadir que, dado que algunas metas son cuantificables, el usuario puede modificar este valor para su proyecto en específico, aclarando que las metas no modificables son medidas por el software, es decir que el sistema automáticamente determina el logro de éstas o no, en caso contrario, el usuario deberá, al final del ciclo, indicar si la meta fue o no lograda.

La siguiente imagen muestra el menú de una meta creada por el usuario cuando elige el primer ícono de una nueva meta, donde debe especificar los atributos de valor, comparación y unidad de medida.



En cuanto a las metas de rol, cada uno de ellos diligencia sus propias metas para asegurar las decisiones personales para el logro en el proyecto.

5.4 Estrategia

La fase de estrategia, desde el punto de vista del software, consta de diligenciamiento de dos formatos TSP: STRAT e ITL.

El formato STRAT se compone principalmente por las partes en las que se divide el producto. Cada una de estas partes son fácilmente agregables mediante el ícono (+) localizado en la parte inferior derecha, el cual despliega la siguiente ventana donde se solicita la información del producto a adicionar.



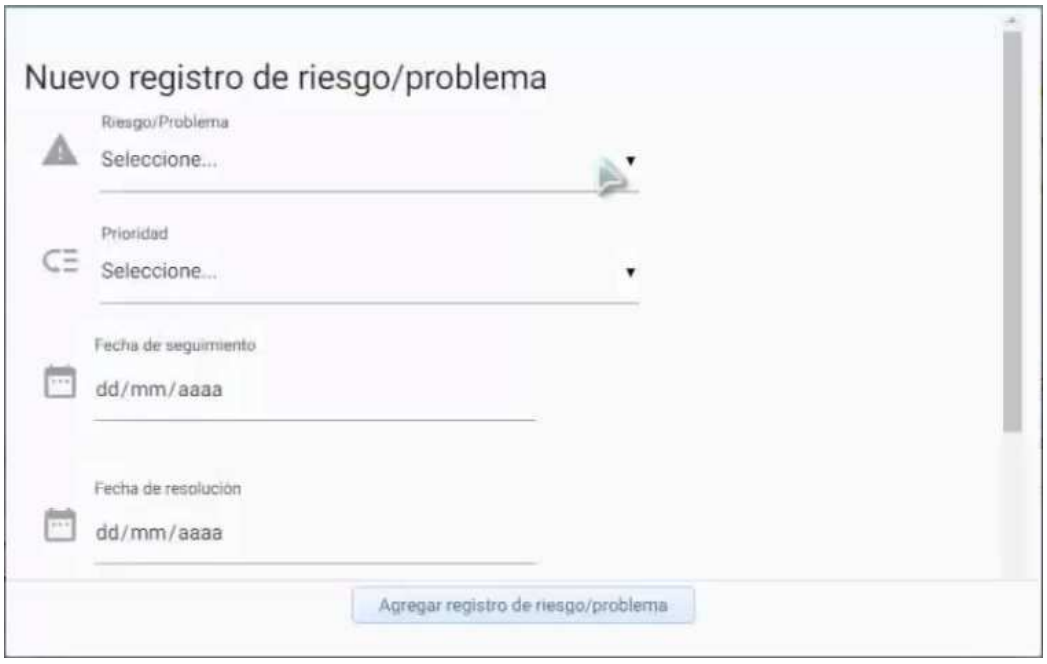
Una vez el producto es adicionado al formato STRAT, cada uno de ellos, considerado como parte, tiene los íconos **+** y **x** que permiten agregar partes de menor jerarquía de acuerdo a la clasificación de partes de TSP y eliminar la parte, respectivamente.

Recordando que la clasificación de TSP consta de Producto, Componente y Módulo en ese orden, las partes de tipo módulo no podrán tener otras partes que las compongan. Asimismo, se recalca que las únicas partes con funciones asociadas, equivalen al menor grado de jerarquía de la parte (nodos hoja), de modo que si una parte tiene funciones y ésta recibe una parte de menor jerarquía, sus funciones serán entregadas a la nueva parte creada.

Parte	Función	Pts. funcls	Horas		
		C1	C1		
Usuarios + ×	Crear usuario	125,0	3,0	⊗	+
	Modificar contraseña y datos de usuario - Activar/Desa	100,0	3,0	⊗	
Ventana Login + ×	Validar el usuario y el perfil en el sistema (funcionario	150,0	4,0	−	+
Inscripciones + ×	Crear inscripción de afiliado (con o sin grupo familiar)	100,0	2,0	⊗	+
	Modificar datos, cancelar inscripción	100,0	2,0	⊗	
	Detalle de la inscripción	75,0	2,0	⊗	
	Listar/Reportar inscripciones - filtros/opciones de cons	150,0	2,0	⊗	
Eventos + ×	Crear evento	125,0	3,0	⊗	+
	Modificar datos, abrir evento, cerrar evento, cancelar e	150,0	3,0	⊗	
	Detalle del evento - número de inscripciones, (total re	125,0	3,0	⊗	
	Listar eventos, filtro por estado	150,0	3,0	⊗	
Proveedores + ×	Crear proveedor	100,0	2,0	⊗	+
	Modificar datos	100,0	2,0	⊗	
	Ver detalle del proveedor	75,0	2,0	⊗	
	Listar proveedores - Opción de eliminar un proveedor	150,0	2,0	⊗	
Afiliados y familiares + ×	Crear afiliado, familiar	100,0	3,0	⊗	+
	Modificar datos del afiliado, activar o desactivar	100,0	3,0	⊗	
	Ver detalles	75,0	3,0	⊗	
	Listar/Reportar afiliados - Opción de eliminar un afiliac	150,0	3,0	⊗	
Total		2200.0	50.0		

Cada parte debe tener por lo menos una función. Para añadir y remover funciones, en las dos últimas columnas se encuentran los íconos + y ⊗, respectivamente.

Una vez se ha determinada la estrategia, el equipo puede proceder a establecer los riesgos del proyecto con base en las partes creadas accediendo al menú del proyecto, localizando el formato ITL, el cual se diligencia agregando información de cada defecto dando click sobre el ícono ⊕.



Por cada riesgo o problema a diligenciar, debe indicarse el tipo, prioridad, fecha de resolución y una descripción. La siguiente imagen muestra un ejemplo del formato ITL del equipo para la parte de Usuarios.



5.5 Planeación

La fase de planeación es el pilar tanto del desarrollo con la metodología TSP, como del software. Con la información brindada, especialmente sobre el formato TASK, los procesos tendrán un punto de referencia para hacer mediciones automáticas. Por tanto, cada integrante tendrá acceso a los datos de los roles que le corresponden, sin embargo el gerente de planeación tiene la responsabilidad de diligenciar toda la información del equipo, de ser necesario.

Las siguientes, dos imágenes, muestran el formato TASK dividido en dos por cuestiones de tamaño. La primera parte muestra cómo cada tarea está clasificada por etapas, con un producto asociado, y un nombre único y específico. La columna del *N. integts* se actualiza automáticamente, de acuerdo a la cantidad de personas del proyecto. Particularmente en esta primera parte del formato TASK se puede observar que sólo el líder de proyecto tiene habilitada la edición, porque, como se mencionó anteriormente, cada integrante edita la información que le corresponde dar a conocer explícitamente al software.

Tarea				Horas del plan			
Etapa	Producto	Nombre de la tarea	N. Integrts.	Líder de proyecto	Gerente de desarrollo	Gerente de planeación	Gerente de calidad/proceso
Gerencia y miscelanea	Usuarios	Gestión documental	6	2,5	2,5	2,5	2,5
Lanzamiento y estrategia	Usuarios	Lanzamiento	2	0	2,5	2,5	0,0
Estrategia y planeación	Usuarios	Planeación del tamaño	2	0	0	2,5	2,5
	Usuarios	Identificación de riesgos	3	0	0	2,5	0
Requerimientos	Usuarios	Administración de la configuración	4	0	0	2,5	2,5
Plan de prueba de sistema	Usuarios	Planeación general	6	2,5	2,5	1,5	2,5
Inspección de requerimientos	Inscripciones	Plan de calidad	3	2,5	0	2,0	2,5
Diseño de alto nivel	Eventos	M&M	4	2,0	3,0	0	0
Plan de prueba de integración	Eventos	Especificación de requerimientos (ERS)	5	3,0	2,0	0	2,0
Inspección de diseño de alto nivel (HLD)	Eventos	Inspección ERS	6	1,0	2,0	2,0	2,0
Diseño detallado	Eventos	Plan de diseño detallado	3	0	2,0	0	0
Revisión de diseño detallado (DLD)	Ventana Login	Gestión de revisión de diseño detallado	4	0	2,0	0	2,0
Desarrollo de prueba	Inscripciones	Plan de prueba de sistema	5	1,0	2,0	0	2,0
Inspección de diseño detallado (DLD)	Afiliados y familiares	Inspección plan de pruebas de sistema	4	1,0	2,0	0	0
Código	Ventana Login	Implementación	4	0	2,0	0	2,0
Revisión de código	Ventana Login	Revisión de código	5	1,0	1,0	0	2,0
Compilación	Inscripciones	Inspección de código	4	0,0	1,5	0	2,0
Inspección de código	Inscripciones	Plan de pruebas de unidad	4	1,0	2,0	0	0
Prueba de unidad	Ventana Login	Pruebas de unidad	4	0	1,5	2,0	0,0
Prueba de construcción e integración	Eventos	Construcción e integración	5	2,0	1,15	0	1,0
Prueba de sistema	Inscripciones	Pruebas de sistema	5	2,0	1,0	1,0	0
Documentación	Eventos	Documentación	5	2,0	2,0	2,0	0
Postmortem	Eventos	M&M Postmortem	6	1,0	1,0	1,0	1,3
			Total:	24.5	35.65	24.0	28.8

			Tamaño/Valor del plan					Real			
Gerente de soporte	Total horas	Acumulado horas	Unidad de tamaño	Tamaño	Número de semana	VP	VP Acumulado	Horas	Horas acumuladas	Número de semana	
5,0	15.0	15.0	Páginas	8,0	1	0,078145355	0,078145355	19.0529	19.0529	2	-
0	5.0	20.0	Páginas	4,0	1	0,02604845	0,10419381	5.6563	24.7092	2	-
0	5.0	25.0	Páginas	5,0	1	0,02604845	0,13024226	5.4895	30.1987	2	
5,0	7.5	32.5	Páginas	10,0	1	0,039072677	0,16931494	7.7541	37.9528	2	
5,0	10.0	42.5	Páginas	7,0	1	0,0520969	0,22141184	10.3554	48.3082	2	-
5,0	14.0	56.5	Páginas	7,0	1	0,07293566	0,2943475	15.5573	63.8655	2	-
0	7.0	63.5	Funciones	12,0	1	0,03646783	0,33081532	7.5016	71.3671	3	-
2,0	7.0	70.5	Páginas	15,0	1	0,03646783	0,36728314	7.2427	78.6098	2	-
4,0	11.0	81.5	Páginas	12,0	1	0,05730659	0,42458972	11.3301	89.9399	2	-
4,0	11.0	92.5	Páginas	14,0	2	0,05730659	0,4818963	11.75548	101.69538	3	-
4,0	6.0	98.5	Páginas	12,0	2	0,03125814	0,51315445	4.3332	106.02858	3	-
4,0	8.0	106.5	Páginas	7,0	2	0,04167752	0,554832	6.2182	112.24678	3	-
4,0	9.0	115.5	Líneas de código	6,0	2	0,04688721	0,6017192	10.8258	123.07258	3	-
4,0	7.0	122.5	Archivos	12,0	2	0,03646783	0,63818705	5.4761	128.54868	3	-
4,0	8.0	130.5	Líneas de código	1,0	2	0,04167752	0,6798646	8.2245	136.77318	3	-
4,0	8.0	138.5	Páginas	9,0	3	0,04167752	0,7215421	9.6064	146.37958	4	-
4,0	7.5	146.0	Líneas de código	11,0	3	0,039072677	0,7606148	5.2717	151.65128	4	-
4,0	7.0	153.0	Líneas de código	4,0	3	0,03646783	0,79708266	7.2329	158.88418	4	-
4,0	7.5	160.5	Archivos	12,0	3	0,039072677	0,83615535	9.573	168.45718	4	-
2,0	6.15	166.65	Funciones	1,0	3	0,032039594	0,86819494	6.4108	174.86798	4	-
3,0	7.0	173.65	Funciones	12,0	4	0,03646783	0,9046628	7.2895	182.15748	4	-
4,0	10.0	183.65	Funciones	8,0	4	0,0520969	0,9567597	8.2532	190.41068	5	-
4,0	8.3	191.95	Páginas	12,0	4	0,043240428	1,0000001	8.69858	199.10926	5	-
79.0											

El acumulado en horas, calculado a partir de los datos diligenciados por los integrantes, presenta la sumatoria exacta de los valores dados. Posteriormente se debe especificar la unidad de tamaño, el tamaño y número de semana en el que se ejecutará la tarea. En forma automática el sistema también diligencia el VP, dado el valor total de horas del ciclo frente al número de horas acumuladas para la tarea.

Finalmente las columnas relacionadas con los datos reales del proyecto, se calculan automáticamente por el software, así que, una vez todos los integrantes terminan de diligenciar este formato, se continúan las demás tareas, sin importar que los campos reales no contengan valores, ya que sus valores serán calculados conforme avanza el proyecto.

Es importante añadir que la última columna sólo está disponible para el gerente de planeación mediante el ícono con el que puede eliminar las tareas, siempre que cada etapa tenga dos o más tareas asignadas.

5.6 Registro de tiempo

Con las tareas definidas por la fase de planeación, el equipo TSP puede ejecutar dichas tareas mediante una interfaz simple y amigable, que llevará el control, tal como el formato LOGT lo propone.

Cada integrante debe seleccionar el rol con el cual va a ejecutar una tarea específica del proyecto, y que lógicamente, le ha sido asignada a dicho rol en la planeación.

Una vez el integrante del equipo selecciona el rol con el cual desea trabajar, el sistema carga las tareas que le fueron asignadas y las despliega sobre la sección de tarea. Una vez cargadas dichas tareas, el usuario selecciona la que desea ejecutar en el momento.

En las siguientes figuras se puede observar la selección de las opciones, para posteriormente dar click sobre el botón comenzar, para dar inicio al marcado de tiempo. Sin embargo, para asegurar la decisión del usuario, el sistema despliega

un mensaje de confirmación para dar el inicio asegurado de la tarea y proceder a marcar el tiempo.





Una vez que se ha aceptado el comienzo de la tarea, el sistema da a conocer dicho momento, mostrando la fecha y tiempo exacto cuando se dio comienzo a la tarea.

Así como lo menciona la metodología TSP, el objetivo es conocer exactamente el tiempo dedicado a una tarea, y por tanto, debe diligenciarse el tiempo que tardan las interrupciones que normalmente ocurren en la ejecución de una tarea. Frente a esta necesidad, el software brinda dos tipos de interrupción: interrupciones previstas e interrupciones imprevistas. Las interrupciones previstas constan de interrupciones que el usuario conoce de antemano para así marcar dicho tiempo en el software dando click al botón *Interr. Prevista,* el cual mostrará una ventana

a la espera del regreso del usuario para ejecutar la actividad. Esta ventana es la que se muestra a continuación.

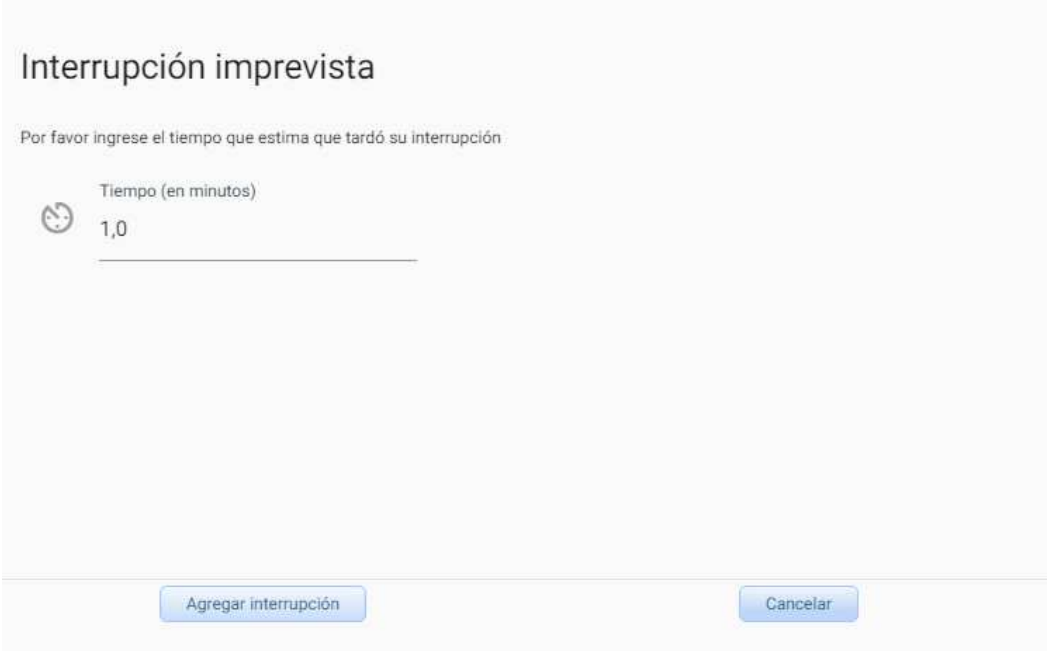


Una vez el usuario regresa a la ejecución de la tarea, y requiere terminar el tiempo de interrupción, da click sobre el el botón *Terminar interrupción* para que la ventana sea cerrada y el tiempo total de interrupción sea consolidado al total de interrupciones como se muestra en la siguiente figura.

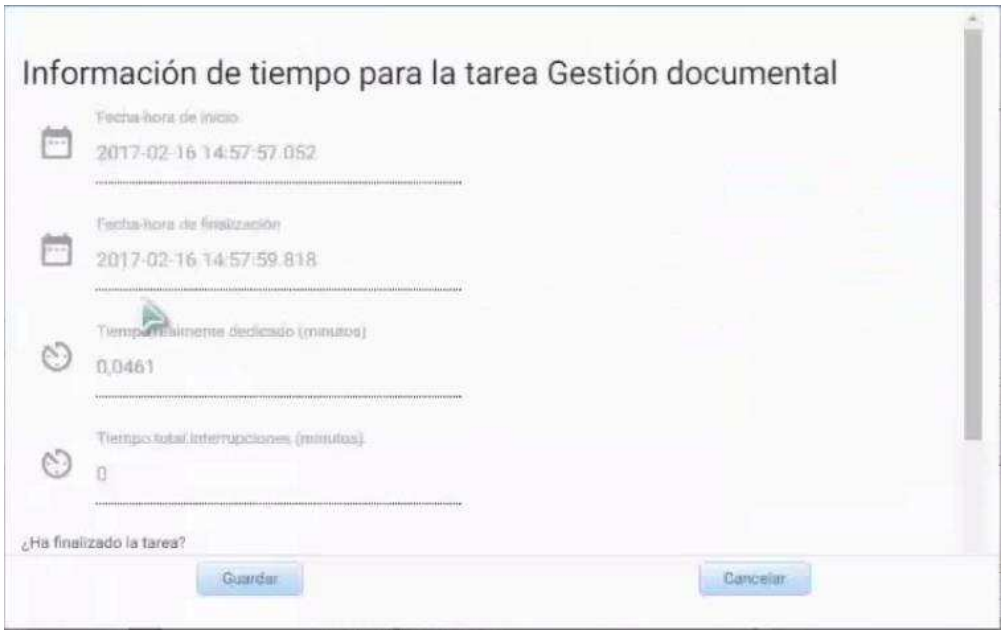


En cuanto a las interrupciones imprevistas, basta con que el usuario de click sobre el botón *Interr. Imprevista* para que la herramienta despliegue la siguiente ventana, la cual solicita el tiempo aproximado que considera el usuario, tardó en la interrupción. Cabe aclarar que dicho tiempo es validado por el software para

garantizar que no exceda el total de tiempo transcurrido sobre la tarea en ejecución.



Luego de varias iteraciones con las interrupciones, el usuario puede parar la tarea dando click en *Parar,* lo que despliega la siguiente ventana, la cual muestra la información de tiempo dedicado e interrumpido total y que además, solicita al usuario especificar si la tarea ha finalizado o no.



Además de lo mencionado, el sistema solicita comentarios al usuario sobre la tarea, lo que no es obligatorio diligenciar pero puede brindar una información adicional para quienes ejecutan el proyecto.

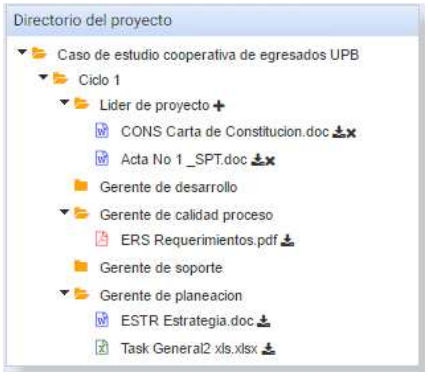
Cuando se deseen consultar los registros del formato LOGD; éste se puede observar como lo muestra la literatura TSP, al acceder al link *logd* y efectuando el filtro a través del integrante y el rol, se cargan todos los registros de sus tareas.

Formato LOGD

Parte		Usuarios				
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	1	50 - Interfaz	Plan de prueba de sistema	Código	15	Seleccior
Descripción						
Al afiliado se le está mostrando gestión de eventos. Por consistencia de roles ésta sólo debe aparecerle a los funcionarios						
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	2	40 - Asignaci	Prueba de unidad	Código	15	Seleccior
Descripción						
En listado de eventos creados, ajustar el formato de fechas para que no muestre las horas						
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	3	50 - Interfaz	Prueba de unidad	Inspección de código	15	Seleccior
Descripción						
En Listado de eventos creados. Está mostrando los valores y no los nombres						
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	4	40 - Asignaci	Prueba de construcción e int	Compilación	10	Seleccior
Descripción						
En la creación y edición de eventos para los campos fecha inicio y fecha fin, activar máscaras para validar la consistencia de la información						
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	5	90 - Sistema	Prueba de sistema	Compilación	5	Seleccior
Descripción						
En creación y edición de eventos quitar el campo valor total, se pretendía incluir al aplicativo valor de cuota del evento						
Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	6	40 - Asignaci	Prueba de sistema	Compilación	15	Seleccior
Descripción						
En creación de eventos validación entre fecha inicio y fecha fin, pero no presenta mensaje de error						

5.7 Directorio del proyecto

Al costado izquierdo, bajo el registro de tiempo, se localiza el directorio del proyecto o cuaderno de proyecto. Este directorio tiene por raíz el directorio del proyecto del que se desprenden los ciclos y roles, permitiendo a los usuarios crear y eliminar tanto directorios y archivos sobre el ciclo actual.





Cada integrante tiene dominio sobre los directorios asociados a los roles que tiene asignados para el ciclo, de modo que sólo podrá descargar para consultar los archivos de otros integrantes. Para agregar un directorio o archivo, basta con seleccionar el ícono + sobre el directorio donde se desea agregar dicho elemento, lo que mostrará la siguiente ventana que divide la creación en dos partes, y el cual permite subir varios archivos simultáneamente, inclusive permite arrastrar los archivos para adjuntarlos más amigablemente.

5.8 Reporte semanal

El reporte semanal es **automático**. La metodología TSP cuenta con los formatos WEEK y SCHEDULE con el propósito de visualizar el progreso del proyecto; en

este caso basta con que el usuario dé click sobre *Actualizar formato* para que el sistema haga síntesis de los datos y los registre en el formato.

A continuación se muestran los formatos SCHEDULE y WEEK respectivamente.



N. Semana	Fecha	Plan			Real			
		Horas directas	Horas acumuladas	VP Acumulado	Horas de equipo	Horas acumuladas	VG semanal	VG acumulado
1	18/02/2017	81.5	81.5	0.42458972	0.0	0.0	0.0	0.0

Registro Schedule de esta semana (semana 1 creado satisfactoriamente)

Actualizar formato SCHEDULE

Formato WEEK (Semana 1)

Información semanal	Plan	Real
Horas del proyecto para esta semana	81.5	0.0
Horas del proyecto para este ciclo a la fecha	81.5	0.0
Valor ganado para esta semana	0.42458972	0.0
Valor ganado este ciclo a la fecha	0.42458972	0.0
Horas totales para las tareas completadas en esta fase hasta la fecha	81.5	0.0

Información semanal	Horas planeadas	Horas reales	Valor Ganado	Valor planeado
Líder de proyecto	12.5	0.0	0.0	0.06885473716666667
Gerente de desarrollo	12.5	0.0	0.0	0.06972301883333333
Gerente de planeación	16.0	0.0	0.0	0.10532256816666667
Gerente de calidad/proceso	14.5	0.0	0.0	0.08708865216666667
Gerente de soporte	26.0	0.0	0.0	0.09360076566666667
Total	81.5	0.0	0.424589742	

Tareas completadas	Horas planeadas	Horas realess	VValor Ganado	Semana planeada
Totales	0.0	0.0	0.0	

Generar Week

5.9 Formatos de resumen

Los indicadores y métricas son finalmente recolectados en los formatos de resumen. Estos formatos solicitan parte de los estimados de los integrantes para con ello unificar los datos y determinar los valores asociados a la información real y planeada del proyecto.

A continuación se muestran los formatos SUMP y SUMQ que sintetizan la información de mayor relevancia para el equipo de desarrollo.

Formato SUMP			
Tamaño del producto		Plan	Real
Páginas de requerimientos (SRS)		5,0	10,0
Otras páginas de texto		3,0	4,0
Páginas de diseño de alto nivel (SDS)		7,0	10,0
Total de líneas de diseño		0	0
Código base (B)		1550,0	1600,0
Código eliminado (D)		0	0
Código modificado (M)		0	0
Código añadido (A)		0	0
Código reutilizado (R)		0	0
Código total nuevo y reutilizado (N)		0,0	0,0
Código total (T)		1550,0	1600,0
Intervalo de predicción (70%)		70,0	70,0
Tiempo en fase (horas)	Plan	Real	Real%
Gerencia y miscelanea Tiempo en fase (horas)	15.0	19.0529	9.569067757069663
Lanzamiento y estrategia	5.0	5.6563	2.8408020802246967
Estrategia y planeación	12.5	13.2436	6.651423444595193
Requerimientos	10.0	10.3554	5.20086308391684
Plan de prueba de sistema	14.0	15.5573	7.813448756727838
Inspección de requerimientos	7.0	7.5016	3.7675796695743835
Diseño de alto nivel	7.0	7.2427	3.637550558924281
Plan de prueba de integración	11.0	11.3301	5.690393304660969
Inspección de diseño de alto nivel (HLD)	11.0	11.75548	5.904034799787815
Diseño detallado	6.0	4.3332	2.176292554148411
Revisión de diseño detallado (DLD)	8.0	6.2182	3.123008944938071
Desarrollo de prueba	9.0	10.8258	5.437115280323979
Inspección de diseño detallado (DLD)	7.0	5.4761	2.7502990066860775
Código	8.0	8.2245	4.130646661034248
Revisión de código	8.0	9.6064	4.824687711661427
Compilación	7.5	5.2717	2.647641802294881
Inspección de código	7.0	7.2329	3.632628638165799
Prueba de unidad	7.5	9.573	4.807913002137621
Prueba de construcción e integración	6.15	6.4108	3.219739754946606
Prueba de sistema	7.0	7.2895	3.6610552417300934
Documentación	10.0	8.2532	4.145060857541231
Postmortem	8.3	8.69858	4.368747088909878
Total	191.95000000000002	199.10926	100.0
Defectos inyectados	Plan	Real	Real%
Estrategia y planeación	2.5	0	0.0
Requerimientos	2.0	0	0.0
Plan de prueba de sistema	2.8000000000000003	1	8.333333333333332
Inspección de requerimientos	1.4000000000000001	0	0.0
Diseño de alto nivel	1.4000000000000001	0	0.0
Plan de prueba de integración	2.2	0	0.0
Inspección de diseño de alto nivel (HLD)	2.2	0	0.0
Diseño detallado	1.2000000000000002	0	0.0
Revisión de diseño detallado (DLD)	1.6	0	0.0
Desarrollo de prueba	1.8	0	0.0
Inspección de diseño detallado (DLD)	1.4000000000000001	0	0.0
Código	1.6	0	0.0
Revisión de código	1.6	0	0.0
Compilación	1.5	0	0.0
Inspección de código	1.4000000000000001	0	0.0
Prueba de unidad	1.5	3	25.0
Prueba de construcción e integración	1.2300000000000002	1	8.333333333333332
Prueba de sistema	1.4000000000000001	7	58.333333333333336
Total	30.73	12	100.0
Defectos removidos	Plan	Real	Real%
Estrategia y planeación	1.75	0	0.0
Requerimientos	1.4	0	0.0
Plan de prueba de sistema	1.9600000000000002	0	0.0
Inspección de requerimientos	0.9800000000000001	0	0.0
Diseño de alto nivel	0.9800000000000001	0	0.0
Plan de prueba de integración	1.54	0	0.0
Inspección de diseño de alto nivel (HLD)	1.54	0	0.0
Diseño detallado	0.8400000000000002	0	0.0
Revisión de diseño detallado (DLD)	1.12	0	0.0
Desarrollo de prueba	1.26	0	0.0
Inspección de diseño detallado (DLD)	0.9800000000000001	0	0.0
Código	1.12	3	25.0
Revisión de código	1.12	0	0.0
Compilación	0.75	4	33.33333333333333
Inspección de código	0.9800000000000001	5	41.66666666666667
Prueba de unidad	1.35	0	0.0
Prueba de construcción e integración	0.9840000000000002	0	0.0
Prueba de sistema	1.12	0	0.0
Total	21.774000000000008	12	100.0
Prueba de aceptación	5.821399999999995	0.0	
Vida de producto	14.777399999999988	0.0	
Resumen		Plan	Real
Código/Hora		0.0	0.0
CPI (índice de costo_desempeño)			0.9640435608067651
Porcentaje de reuso		0.0	0.0

Formato SUMQ		
Porcentaje libre de defectos		Real(%)
En compilación		100.0
En prueba de unidad		0.0
En construcción e integración		0.0
En prueba de sistema		0.0
Defectos/Página	Plan	Real
Inspección de requerimientos	0.196	0.0
Inspección de diseño de alto nivel (HLD)	0.22	0.0
Defectos/Código	Plan	Real
Revisión de diseño detallado (DLD)	0	0
Inspección de diseño detallado (DLD)	0	0
Revisión de código	0	0
Compilación	0	0
Inspección de código	0	0
Prueba de unidad	0	0
Construcción e integración	0	0
Prueba de sistema	0	0
Desarollo total	0	0
Prueba de aceptación	0	0
Vida de producto	0	0
Total	0	0
Proporciones de defecto	Plan	Real
Revision diseño detallado/Diseño detallado	0.8296296296296296	0.0
Revisión de código/Compilación	1.4933333333333334	0.0
Proporciones de tiempo de desarrollo	Plan	Real
Inspección de requerimientos/Requerimientos	0.7	0.724414315236495
Inspección de diseño de alto nivel (HLD)/Diseño de alto nivel	1.5714285714285714	1.6230797906857939
Diseño detallado/Código	0.75	0.5268648550063832
Revisión diseño detallado (DLD)/Diseño detallado	1.3333333333333333	1.4350133850272317
Revisión de código/Código	1.0	1.1680223721806797
A/FR	Plan	Real
A/FR	23.466666666666665	21.787937869407937
Rendimientos de fase	Plan (%)	Real (%)
Estrategia y planeación	70,0	0.0
Requerimientos	70	0
Plan de prueba de sistema	70,0	0.0
Inspección de requerimientos	70	0.0
Diseño de alto nivel	70,0	0.0
Plan de prueba de integración	70,0	0.0
Inspección de diseño de alto nivel (HLD)	70	0.0
Diseño detallado	70,0	0.0
Revisión de diseño detallado (DLD)	70	0.0
Desarrollo de prueba	70,0	0.0
Inspección de diseño detallado (DLD)	70	0.0
Código	70,0	0.0
Revisión de código	70	-1.5
Compilación	50	-0.0
Inspección de código	70	-0.6666666666666666
Prueba de unidad	90	-0.625
Prueba de construcción e integración	80	-0.0
Prueba de sistema	80	0
Rendimientos de proceso	Plan	Real
Antes de compilación	0.7000000000000001	3.0
Antes de prueba de unidad	0.7119047619047762	0.75
Antes de prueba de construcción e integración	0.7160045402951194	0.6
Antes de prueba de sistema	0.7197396963123647	0.5
Antes de prueba de aceptación	0.7085584119752687	1.0
Tasa de inyección de defectos	Plan(.)	Real(.)
Estrategia y planeación	0,2	0.0
Requerimientos	0,2	0.0
Plan de prueba de sistema	0,2	0.06427850591040862
Inspección de requerimientos	0,2	0.0
Diseño de alto nivel	0,2	0.0
Plan de prueba de integración	0,2	0.0
Inspección de diseño de alto nivel (HLD)	0,2	0.0
Diseño detallado	0,2	0.0
Revisión de diseño detallado (DLD)	0,2	0.0
Desarrollo de prueba	0,2	0.0
Inspección de diseño detallado (DLD)	0,2	0.0
Código	0,2	0.0
Revisión de código	0,2	0.0
Compilación	0,2	0.0
Inspección de código	0,2	0.0
Prueba de unidad	0,2	0.31338138514572234
Prueba de construcción e integración	0,2	0.15598677232170713
Prueba de sistema	0,2	0.9602853419301735
Tasa de remoción de defectos	Plan(.)	Real(.)
Estrategia y planeación	0.14	0.0
Requerimientos	0.13999999999999999	0.0
Plan de prueba de sistema	0.14	0.0
Inspección de requerimientos	0.14	0.0
Diseño de alto nivel	0.14	0.0
Plan de prueba de integración	0.14	0.0
Inspección de diseño de alto nivel (HLD)	0.14	0.0
Diseño detallado	0.14000000000000004	0.0
Revisión de diseño detallado (DLD)	0.14	0.0
Desarrollo de prueba	0.14	0.0
Inspección de diseño detallado (DLD)	0.14	0.0
Código	0.14	0.36476381542950936
Revisión de código	0.14	0.0
Compilación	0.1	0.7587685186941594
Inspección de código	0.14	0.6912856530575564
Prueba de unidad	0.18000000000000002	0.0
Prueba de construcción e integración	0.16000000000000003	0.0
Prueba de sistema	0.16	0.0
Inspección/Proporciones de revisión	Plan	Real
Inspección de requerimientos	0.7142857142857143	1.3330489495574278
Inspección de diseño de alto nivel (HLD)	0.6363636363636364	0.850667093134436
Revisión de diseño detallado (DLD)	0.0	0.0
Inspección de diseño detallado (DLD)	0.0	0.0
Revisión de código	0.0	0.0
Inspección de código	0.0	0.0

5.10 Otros formatos

5.10.1 Registro de defectos

Al igual que el registro de riesgos, el formato LOGD se diligencia a través de una ventana que solicita todos los datos en forma individual. En caso de requerir algún cambio, el integrante que ingresó el defecto es el único con el privilegio de modificarlo.



Formato LOGD

Parte	Usuarios

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	1	50 - Interfaz	Plan de prueba de sistema	Código	15	Seleccior

Descripción

Al afiliado se le está mostrando gestión de eventos. Por consistencia de roles ésta sólo debe aparecerle a los funcionarios

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	2	40 - Asignaci	Prueba de unidad	Código	15	Seleccior

Descripción

En listado de eventos creados, ajustar el formato de fechas para que no muestre las horas

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	3	50 - Interfaz	Prueba de unidad	Inspección de código	15	Seleccior

Descripción

En Listado de eventos creados. Está mostrando los valores y no los nombres

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	4	40 - Asignaci	Prueba de construcción e int	Compilación	10	Seleccior

Descripción

En la creación y edición de eventos para los campos fecha inicio y fecha fin, activar máscaras para validar la consistencia de la información

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	5	90 - Sistema	Prueba de sistema	Compilación	5	Seleccior

Descripción

En creación y edición de eventos quitar el campo valor total, se pretendía incluir al aplicativo valor de cuota del evento

Fecha	Número	Tipo	Etapa de inyección	Etapa de remoción	Tiempo de arreglo (en minutos)	Defecto arreglado
11/3/2017	6	40 - Asignaci	Prueba de sistema	Compilación	15	Seleccior

Descripción

En creación de eventos validación entre fecha inicio y fecha fin, pero no presenta mensaje de error

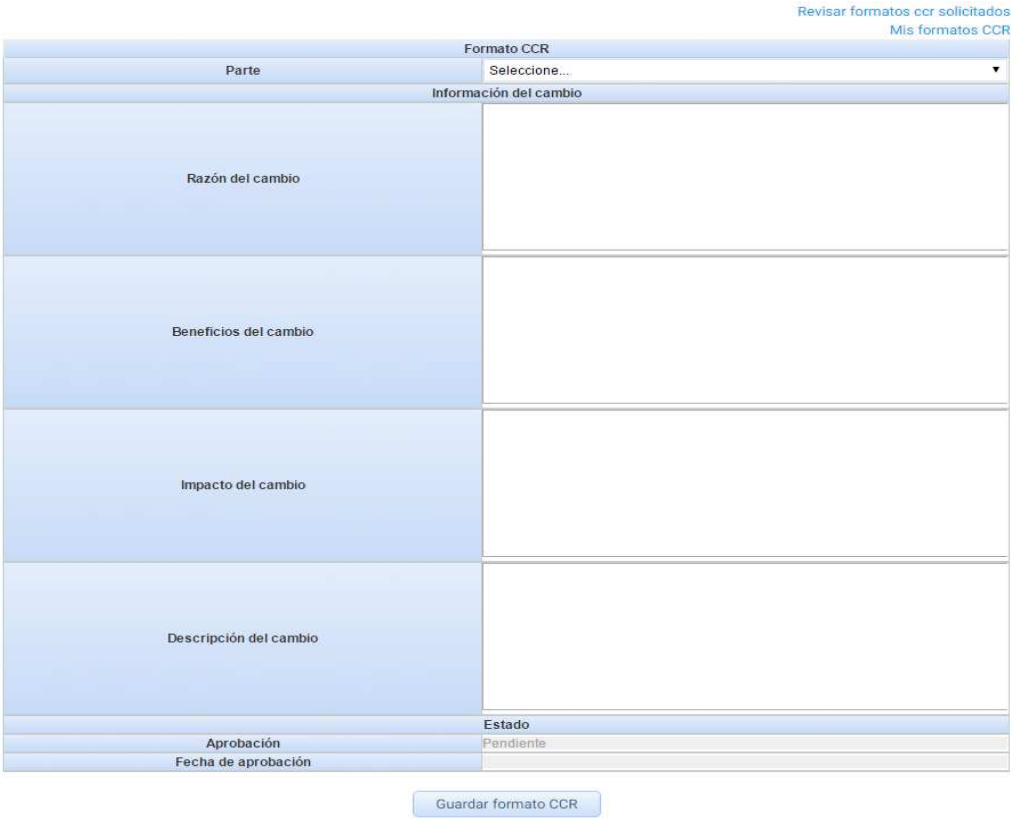
5.10.2 Formato PEER

Como el formato PEER se diligencia una sola vez por ciclo, éste se carga con los datos diligenciados hasta que el ciclo ha finalizado. El integrante diligencia la información completa.

Formato PEER						
Trabajo y dificultad por cada rol						
Rol	Trabajo requerido		Dificultad			
Líder de proyecto	20,0		20,0			
Gerente de desarrollo	20,0		20,0			
Gerente de planeación	20,0		20,0			
Gerente de calidad/proceso	20,0		20,0			
Gerente de soporte	20,0		20,0			
Total	100.0		100.0			
Evaluación de criterios del equipo						
Espíritu de equipo	○ 1	○ 2	○ 3	○ 4	● 5	
Efectividad general	○ 1	○ 2	○ 3	○ 4	● 5	
Experiencia gratificante	○ 1	○ 2	○ 3	○ 4	● 5	
Productividad de equipo	○ 1	○ 2	○ 3	○ 4	● 5	
Calidad del proceso	○ 1	○ 2	○ 3	○ 4	● 5	
Calidad del producto	○ 1	○ 2	○ 3	○ 4	● 5	
Contribución general						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	
Ayuda y soporte						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	
Desempeño						
Líder de proyecto	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de desarrollo	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de calidad/proceso	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de soporte	○ 1	○ 2	○ 3	● 4	○ 5	
Gerente de planeación	○ 1	○ 2	○ 3	● 4	○ 5	

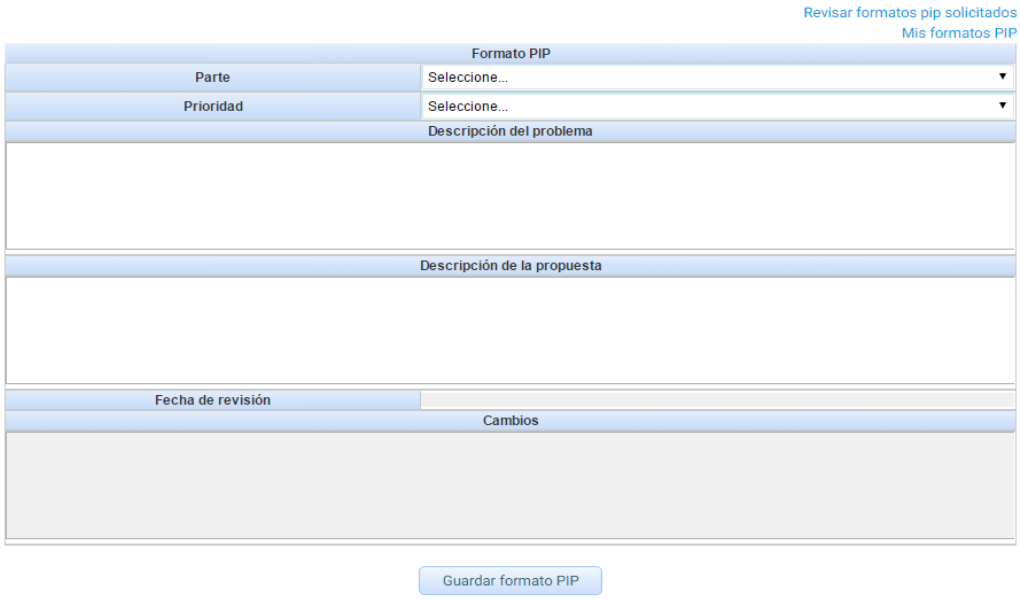
5.10.3 Formato CCR

El formato CCR puede ser diligenciado en cualquier momento del proyecto, tomando por referencia las partes en que está divido el producto. El formato CSR, el cual muestra las solicitudes CCR se sintetiza en el menú superior de la siguiente imagen, donde el rol responsable, debe hacer una validación de la solicitud de cambio.



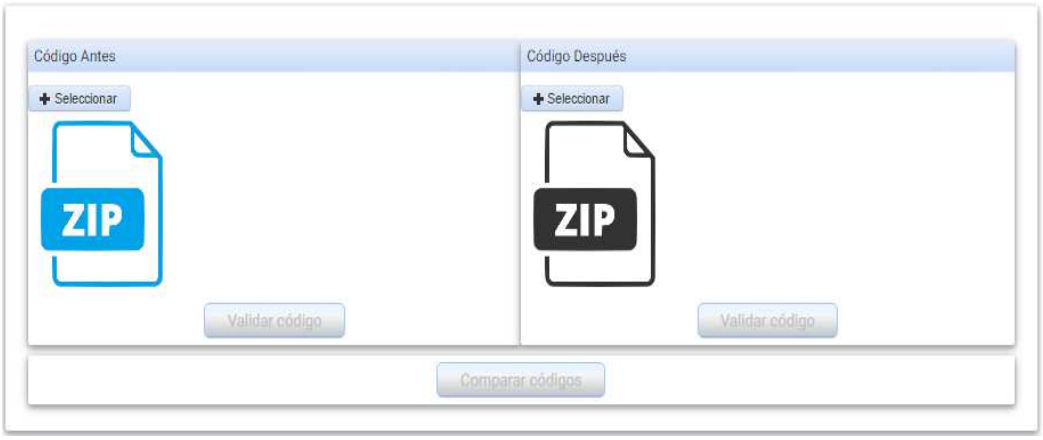
5.10.4 Formato PIP

Se comporta en forma similar al formato CCR, pues se diligencia en varias ocasiones, sin embargo requiere dar a conocer el nivel de prioridad además de la parte en la que está dividida el producto. El formato PIP debe ser igualmente revisado por el rol a cargo de dicha responsabilidad.



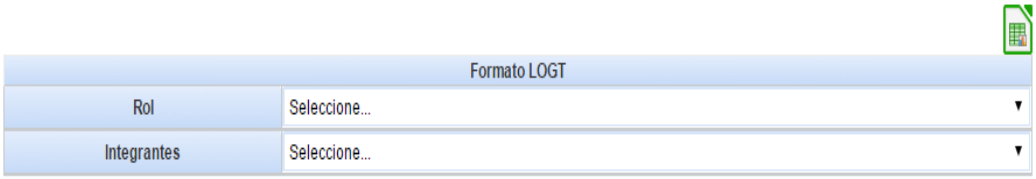
5.11 Comparador de código

Medir el tamaño del producto software genera costos por la especial atención a los cambios presentados durante el desarrollo; por esta razón se propone una herramienta software capaz de comparar dos versiones de código a fin de identificar rápidamente las funciones agregadas, eliminadas y modificadas.



5.12 Exportador de datos

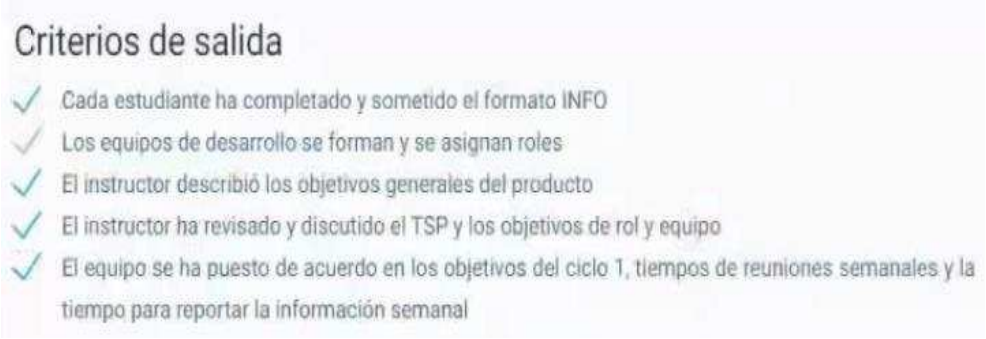
En el prototipo, resultado de este proyecto, se presenta el ícono mostrado en la parte superior derecha de cada formato, tal como se muestra en la siguiente figura. Dando click en dicho ícono, el usuario podrá descargar la información del formato en una hoja de cálculo.



5.13 Criterios de salida de fase

A la derecha del panel de proyecto se muestra un ícono que permite pasar de una fase a otra. Esta transacción debe ser validada por los procesos ejecutados durante la fase actual, de modo que el software es capaz de validar algunos criterios de salida, y el usuario deberá marcar aquellas opciones ajenas al software, dando por hecho que el criterio realmente se ha cumplido.

5.13.1 Fase de Lanzamiento



5.13.2 Fase de Estrategia



5.13.3 Fase de Planeación



5.13.4 Fase de Requerimientos



5.13.5 Fase de Diseño



5.13.6 Fase de Implementación



5.13.7 Fase de Pruebas



5.13.8 Fase de Postmortem



6. 3.14 Resultados

7. *3.14.1 Logro de metas*

Cuando ha llegado la fase de Postmortem, los usuarios tienen la responsabilidad de indicar si han logrado o no sus metas, recordando que algunas de ellas son evaluadas en forma automática por el software.

Si los usuarios no diligencian dicha información, el sistema considera que no fue lograda la meta, así que se recomienda diligenciarlo bajo la mayor honestidad posible.

Meta	Comparativa	Valor	Unidad de medida	Logro
Información registrada del ciclo	igual a	70.00	porcentaje	No lograda
Completar el desarrollo con un corto desfase de días	igual a	1.00	día(s)	Lograda
Defectos encontrados en compilación	igual a	80.00	porcentaje	No lograda
Defectos encontrados en la prueba de sistema	igual a	10.00	cantidad	No lograda
Requerimientos funcionales incluidos al terminar el proyecto 1era fase	igual a	60.00	porcentaje	Lograda
Requerimientos funcionales incluidos al terminar el proyecto 2da fase	igual a	100.00	porcentaje	Lograda
Error en el estimado de tamaño del producto	menor a	20.00	porcentaje	No lograda
Error en las horas estimadas de desarrollo	menor a	18.00	cantidad	Lograda

Líder de proyecto	
Meta	Logro
Construir y mantener un equipo efectivo	Lograda
Motivar a todos los miembros del equipo a participar activamente en el proyecto	No lograda
Resolver todos los problemas que exponen los integrantes del equipo	Lograda
Mantener al instructor informado acerca del progreso del equipo	Lograda
Desempeñarse efectivamente como facilitador de las reuniones de equipo	Lograda

Gerente de desarrollo	
Meta	Logro
Producir un producto superior	Lograda
Utilizar por completo las habilidades de los integrantes	No lograda

Gerente de calidad/proceso	
Meta	Logro
Asegurar que todos los integrantes del equipo reportan de manera precisa y utilizan apropiadamente la información de TSP	Lograda
El equipo sigue fielmente TSP y produce un producto de calidad	Lograda
Todas las inspecciones de equipo son apropiadamente reportadas y moderadas	No lograda
Todas las juntas de equipo se reportan de manera precisa y los reportes se almacenan en el cuaderno del proyecto	Lograda

Gerente de soporte	
Meta	Logro
El equipo tiene herramientas adecuadas y métodos para soportar el trabajo	Lograda
No se hacen cambios no autorizados a los productos base	No lograda
Todos los riesgos y problemas del equipo se registran en el lanzamiento de problemas y se reportan cada semana	Lograda
El equipo cumple sus metas de reuso para el ciclo de desarrollo	Lograda

Gerente de planeación	
Meta	Logro
Reportar de manera exacta el estatus semanal del equipo	Seleccione...
Producir un plan completo, preciso y exacto, para cada integrante y el equipo en general	Seleccione...

8. *3.14.2 Resultados cualitativos*

Los resultados cualitativos se evidencian gracias al menú de resultados, donde se brinda una muestra de los resultados de acuerdo al tiempo o contexto. De igual forma, un usuario de la herramienta puede comparar varios de sus proyectos accediendo al menú de proyecto en *Mis proyectos,* cargando la comparación de aquellos que desee.



Información de contacto

alconi@gmail.com

jccamachob94@gmail.com

