INSTITUTO UNIVERSITARIO AERONÁUTICO

Facultad de Ciencias de la Administración

Ingeniería en Sistemas

Proyecto de Grado

Caso de Reingeniería de un Proceso de Desarrollo de Software

2017

Autoras

Eugenia Brea
Gabriela Herrera

Tutora
Ing. María Alejandra Boggio

Este trabajo está dedicado

A nuestros padres y hermanos, por su apoyo y motivación permanente, por su amor incondicional.

A nuestra tutora, Alejandra Boggio, por su calidez, su respeto y acompañamiento en este proceso.

A nuestros formadores, por el gran valor de toda su enseñanza.

Eugenia y Gabriela

Capítulo 1 - Introducción

Listado de Figuras

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Capítulo 1: Introducción

1.1 Descripción de la Empresa

GobFactory es una empresa cordobesa que tiene como misión "Proveer soluciones administrativas confiables y transparentes a Gobiernos Municipales". Desde 1990 ofrecen soluciones para servicios públicos tales como: consultoría, servicios de desarrollo y mantenimiento de software, mesa de ayuda, capacitación y organización administrativa.

La actividad principal de la empresa está orientada a la realización de un único producto “WGob”. Ofrece una solución de software genérica, orientada a las necesidades generales de un sector o negocio, es por esta razón que podría encuadrase dentro de las llamadas "Fábricas de Software". Actualmente, ofrece servicios aproximadamente a 180 municipalidades y 6 cooperativas distribuidas a lo largo de todo el país.

Este tipo de negocio permite la reutilización de arquitecturas, de componentes software y ha logrado que a través del tiempo se haya obtenido un amplio conocimiento sobre el dominio del problema. De este modo, a través de una arquitectura básica, se van modelando particularidades entre sistemas similares en un segmento concreto de mercado, minimizando al máximo los costes y tiempo de desarrollo al poder reutilizar partes comunes tanto como sea posible.

Este trabajo está enfocado en analizar todos los aspectos relacionados con la manera actual en que la empresa encara y lleva adelante su proceso de desarrollo de software (Ver figura *Diagrama del Proceso Actual del Producto WGob v7.0*, página 60). Entre los indicadores de calidad del proceso, se tendrán en cuenta recursos humanos, infraestructuras, formación continua, metodología, arquitectura y gestión.

1.1.1 Antecedentes

A partir del año 2004 y hasta 2010 inclusive, GobFactory toma como referencia para la definición de su proceso de desarrollo al CMMI V1.1 (Integración del Modelo de Madurez y Capacidad), en su representación por Etapas, siendo la opción elegida SW-SE, alcanzando el nivel de madurez 2 (Repetible - En este nivel, las organizaciones disponen de prácticas institucionalizadas de gestión de proyectos, existen métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente).
Los ciclos de vida elegidos por la organización fueron: Ciclo de Vida en Cascada con Realimentación y Ciclo de Vida Iterativo e Incremental.

En el año 2011 la empresa comienza a investigar otras opciones metodológicas. Siendo un periodo en que se dejó de lado también la metodología anterior generando una situación de improvisación y pruebas permanentes.

A partir de 2012 y hasta la actualidad se establece como metodología única de desarrollo a SCRUM buscando introducir mejoras, agilidad y flexibilidad en los procesos.

Sin embargo, a pesar de haber incorporado el cambio de metodología, la empresa continúa con algunas deficiencias, por lo que están comenzando a presentarse interrogantes sobre si la nueva metodología adoptada es la adecuada.

1.1.2 Organigrama

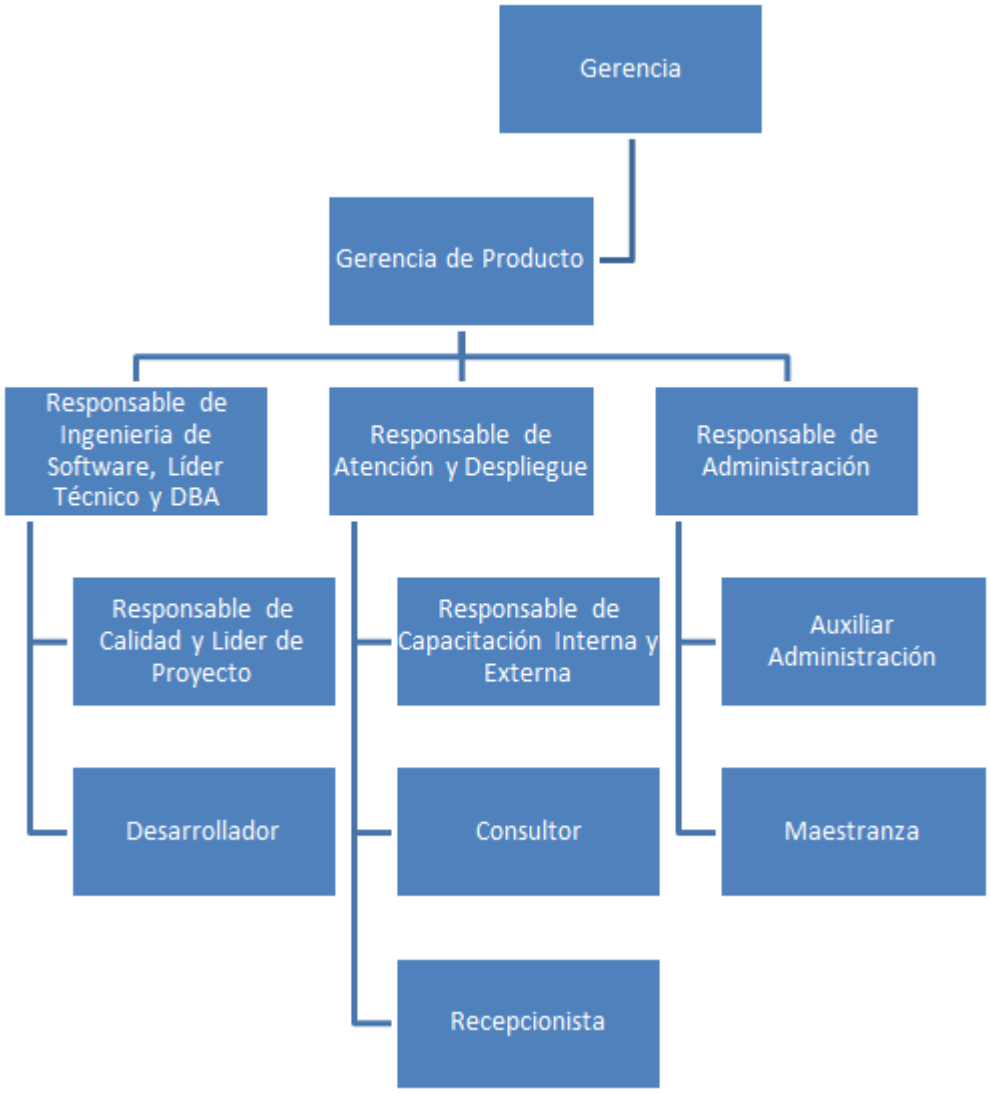

Figura 1: Organigrama de la Empresa (Fuente: GobFactory)

1.1.3 Descripción de Puestos del Área de Desarrollo

1.1.3.1 Analista Funcional: es el encargado de realizar la captura de requerimientos que ingresan al sprint, profundizando su conocimiento a través de las elicitaciones que sean necesarias (entrevistas, lectura de código e investigaciones). Comprendidos los requerimientos, realiza la especificación de los mismos en Historias de Usuarios utilizando la plataforma Team Fundation Server (TFS). Además, realiza los prototipos y diseños que sean necesarios, actualiza los manuales de configuración y crea los instructivos de las nuevas funcionalidades. Realiza las revisiones de la historia con el Dueño del Producto, Líder Técnico y/o Consultor (Cliente Interno) asignado. Realiza la presentación del producto terminado al Dueño del Producto, Consultores y Gerencia. Como funciones inherentes a su rol, participa de la Sprint Planning, estima las tareas que debe llevar a cabo, reporta su avance en la plataforma (horas completadas) y controla el avance de la historia hasta su finalización. Brinda soporte al resto del equipo.

1.1.3.2 Analista de Prueba: es el encargado de escribir los casos de pruebas de las historias que ingresaron sprint en la plataforma TFS y de ejecutarlos (utilizando testing operacional, de stress, de integración y sobrecarga). Ante la presencia de defectos, es el encargado de cargarlos en la plataforma, asignárselos a un desarrollador y efectuar su seguimiento hasta su corrección definitiva. Realiza revisiones de los casos de prueba y, en caso de corresponder, de las historias con el Dueño del Producto, Líder Técnico y/o Consultores. Como funciones inherentes a su rol, participa de la Sprint Planning, estima las tareas que debe llevar a cabo, reporta su avance en la plataforma (horas completadas), adjunta evidencia de las pruebas realizadas, ejecuta tantos ciclos de prueba como defectos sean detectados.

1.1.3.3 Consultores (Clientes Internos): son los encargados de responder las necesidades y requisitos del cliente externo (municipalidades y cooperativas) dando soporte telefónico (mesa de ayuda) o personal en cada visita al municipio. Realizan la implementación de los servidores, del sistema y de los nuevos productos en el cliente, y se ocupan de su mantenimiento realizando visitas mensuales (o quincenales en caso de ser necesarios). Ofrecen y comercializan nuevas prestaciones del sistema. Como funciones inherentes a su rol, asesoran y generan mejoras en el cliente en cuanto a los aspectos operativos y de gestión que permitan un mejor desempeño de las áreas administrativas y un mejor uso del sistema. Brindan soporte al área de Desarrollo.

1.1.3.4 Programadores: son los encargados de la codificación de las historias que ingresaron al sprint. Realizan la prueba unitaria de cada componente de código que liberan, registran esta acción y cargan los mismos

en la herramienta "Gestión de Solicitudes". Publican los componentes de código en la plataforma TFS. Conjuntamente con el Líder Técnico diseña y crea la estructura de datos. Brindan soporte a los analistas funcionales, analistas de prueba, consultores y clientes externos (ante necesidades puntuales de configuración y/o fallas). Reporta las horas completadas en la plataforma TFS.

1.1.3.5 Líder Técnico: es el encargado de la base de datos del sistema. Realiza la migración de datos de los nuevos clientes a la base de datos del sistema implementado. Asesora a los Consultores y Equipo de Desarrollo en cuanto a aspectos técnicos para lograr esta integración al sistema actual de la mejor manera para evitar errores e inconsistencias. Reporta las tareas en la plataforma TFS. Realiza soporte y asesoramiento telefónico al cliente. Participa del Comité de Productos.

1.1.3.6 Scrum Master (Facilitador): Coordina con el Dueño del Producto (PO) las actividades requeridas para poder cumplir con el release. Colabora con la priorización del backlog y ayuda a seleccionar los elementos que formarán parte del próximo sprint. Debe asegurar que todos los miembros del proyecto sigan los valores y principios ágiles, las reglas y proceso de SCRUM y guiar la colaboración intraequipo y con el cliente, ya que es el nexo entre ellos. Motiva y participa como moderador en las reuniones de planning, retrospectivas, diarias y de auditoría de código. Identifica y elimina los impedimentos que se presentan en el sprint. Realiza el seguimiento diario del proyecto (reporte de horas realizadas y trabajo terminado). Enseña al equipo a autogestionarse. Administra las solicitudes de requerimiento a través de la herramienta "Gestión de Solicitudes". Identifica que las bases de datos y el entorno para las pruebas se encuentren actualizados correctamente en tiempo y forma. Controla el versionado y que todos los elementos intervinientes en las iteraciones se encuentren en el TFS. Envía al Encargado de Soporte Técnico las solicitudes que deben desplegarse en el cliente. Capacita sobre la metodología y el proceso a los nuevos miembros del equipo. Participa del Comité de Productos.

1.1.3.7 Dueño del Producto (Product Owner): Representa al cliente externo y a los consultores de la empresa. Es el encargado de preparar y priorizar el backlog. Fija las fechas de entrega. Establece los objetivos del producto y del proyecto. Brinda, durante todo el proyecto, soporte y capacitación sobre el negocio y sobre las normativas tributarias vigentes a todos los miembros. Realiza las validaciones y revisiones pertinentes con los analistas (funcionales y de testing) y con los desarrolladores. Participa del Comité de Productos, de las planning, de las reuniones de validación y de las reuniones de presentación.

1.1.3.8 Comité de Producto: Está integrado por el Dueño del Producto, el Líder Técnico, el Scrum Master y el Responsable de Soporte. Tiene como función principal analizar las solicitudes de los clientes externos/internos, establecer las prioridades de su desarrollo, analiza las factibilidades de lo solicitado, analiza el impacto y riesgo, realiza una pre estimación de las fechas de entrega, incorpora más detalle funcional y técnico, aplica mejoras a las solicitudes. Prepara el backlog del sprint.

1.1.4 Contexto cultural

"Cultura organizacional es todo aquello que identifica a la organización y la diferencia de otra haciendo que sus miembros se sientan parte de ella ya que profesan los mismos valores, creencias, reglas, procedimientos, normas, lenguaje, ritual y ceremonias.
Esta cultura se transmite en el tiempo y se va adaptando de acuerdo a las influencias externas y a las presiones internas producto de la dinámica organizacional. Una de las cuestiones en que parece existir un mayor acuerdo en relación a la cultura es que ésta cumple dos funciones de suma importancia: integra a los miembros de la empresa para que sepan cómo relacionarse y para ayudar a ésta a adaptarse al entorno" (Schein, 1985).
Para conocer un poco más GobFactory analizaremos algunos aspectos que caracterizan su cultura como empresa:

1.1.4.1 Creatividad e innovación: en la actualidad, el equipo de desarrollo comenzó a aplicar mejoras sobre las técnicas de trabajo a los fines de aumentar la productividad del área y disminuir las ineficiencias en las soluciones aportadas. Los analistas funcionales y de testing establecen criterios básicos a tener en cuenta a la hora de realizar análisis y/o testing, estándares de diseño y buenas prácticas.
El líder técnico, por su parte, establece mejoras en cuanto a cómo optimizar las consultas y el acceso a los datos. También, investiga sobre actualizaciones que sean necesarias a nivel de software y que garanticen un buen rendimiento del producto independientemente del hardware en el que se implemente.

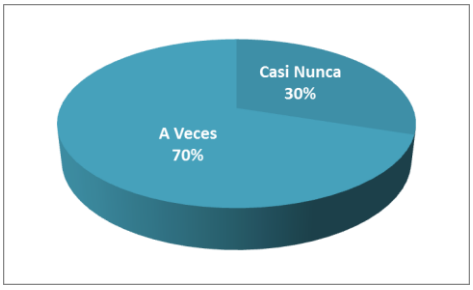

Figura 2: Pregunta "¿GobFactory innova y mejora continuamente?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.2 Formación: la empresa no cuenta con programas anuales de capacitación en las diferentes disciplinas. No obstante, ante la necesidad puntual (por ejemplo, al decidir la implementación de SCRUM como metodología de trabajo o al seleccionar una herramienta de gestión de proyectos como Team Fundation Server), se organiza un plan de capacitación para todo el personal involucrado.

Existe buena predisposición de la empresa en apoyar y asumir el costo de las capacitaciones externas solicitadas por el personal, siempre que éstos tengan alguna relación con el rol que ejerce (Ej. el Scrum Master realizó una capacitación sobre PMI o analistas que realizaron la Diplomatura en Calidad de Software, etc.).

En cuanto a la formación sobre aspectos de negocio, no se requiere acudir a entidades externas dado que la empresa cuenta con personal altamente capacitado para cubrir estas necesidades. Las reuniones para consultas sobre el negocio se realizan generalmente cuando son necesarias o urgentes para enfrentar un requerimiento (sin una planificación previa).

1.1.4.3 Flexibilidad ante problemas internos: no se cuenta con una estrategia proactiva para identificar posibles conflictos o situaciones internas que produzcan que el personal se desmotive o trabaje con bajo rendimiento. En general, la atención está puesta en cuestiones de negocio o compromisos con los clientes y no en el personal.

1.1.4.4 Adaptación a cambios en el entorno: el producto que se fabrica está destinado a un público que se rige por numerosas legislaciones e impuestos nacionales, provinciales y municipales. La empresa posee flexibilidad y capacidad para adaptarse a éstas (y a sus actualizaciones) de forma rápida.

Por otro lado, no lleva adelante políticas de innovación como sucede en otras empresas del rubro. Solo incorpora cambios por una necesidad comercial o de productividad. Los clientes a los cuales apunta no solicitan mayores actualizaciones o innovación más allá de las necesarias para ejecutar sus funciones de forma fácil y segura.

1.1.4.5 Fijación de objetivos concretos a los empleados: sólo se plantean objetivos a nivel de negocio o comercial, no a nivel de área o individual.

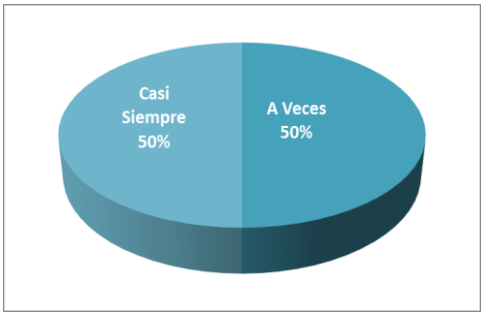

Figura 3: Pregunta "¿cuenta con la información que necesita para realizar su trabajo con excelencia?" (Fuente: Encuestas realizadas por el grupo)

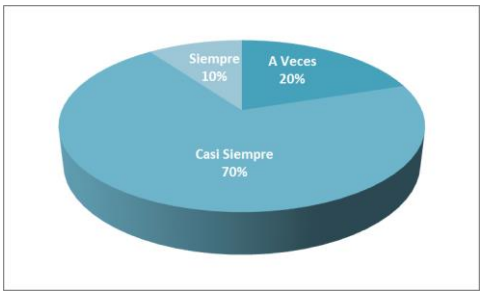

Figura 4: Pregunta "¿tiene claro cuáles son sus tareas y responsabilidades?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.6 Comunicación: Si bien la comunicación debe ser vehículo de esclarecimiento, se detectaron ciertos inconvenientes que resultan del supuesto por parte del Product Owner (PO) de que todo el equipo entiende lo que explica y lo interpreta tal cual él lo ve.

Debido a este supuesto, a menudo surgen malentendidos, interpretaciones erróneas, conversaciones mal encauzadas, con la consecuente pérdida de energía, tiempo y dinero.

Otra sensación que se percibe en los miembros de la empresa es la falta de confianza que los apoye y los impulse. Ante el surgimiento de un error siempre se pone en duda en primer lugar el desempeño del equipo de desarrollo.

Por otro lado, según lo relevado, en muy pocas oportunidades se observa una actitud de reconocimiento del otro cuando se logran los objetivos o ante un trabajo bien logrado. Lo que genera un cierto grado de indiferencia y desmotivación en el grupo.

Como puede verse en los gráficos correspondientes al Anexo sobre Encuesta Clima Organizacional, ítem Comunicación, hay algunos aspectos que deben ser mejorados:

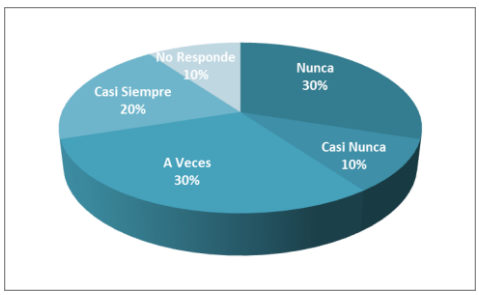

Figura 5: Pregunta "¿Tengo disponible información sobre la organización y la evolución de GobFactory?" (Fuente: Encuestas realizadas por el grupo)

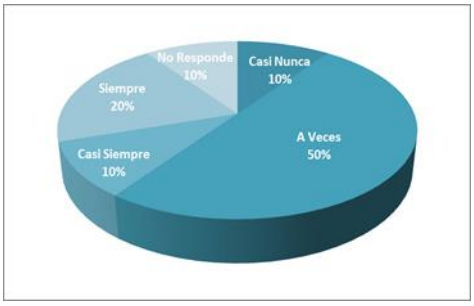

Figura 6: Pregunta "¿Los comunicados internos me proporcionan información útil?" (Fuente: Encuestas realizadas por el grupo)

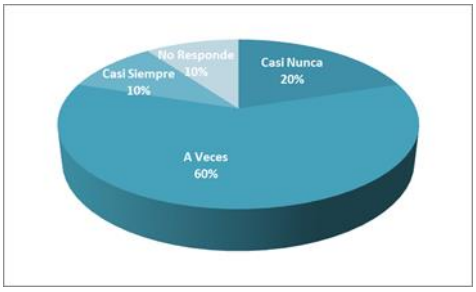

Figura 7: Pregunta "¿La comunicación sobre los resultados y marcha de la Compañía es clara y transparente?" (Fuente: Encuestas realizadas por el grupo)

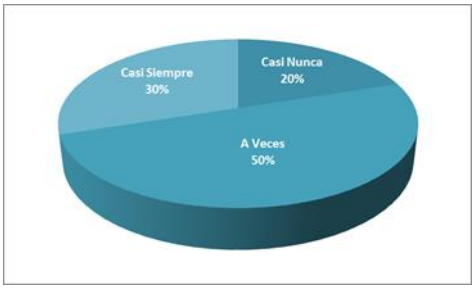

Figura 8: Pregunta "¿La comunicación interna en GobFactory es una actividad permanente y planificada?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.7 Iniciativa individual: generalmente surgen del área de desarrollo y del dueño del producto. Los primeros, tienden a mejorar la performance del producto y la satisfacción propia de hacer cosas nuevas. El segundo, busca mejorar las técnicas de trabajo del área de desarrollo y la satisfacción del cliente a través de nuevas prestaciones o de la reingeniería de las existentes.

1.1.4.8 Cooperación: los miembros del equipo de desarrollo tienen buena predisposición para colaborar y dar soporte al resto de sus miembros. También hay colaboración de los consultores al equipo.

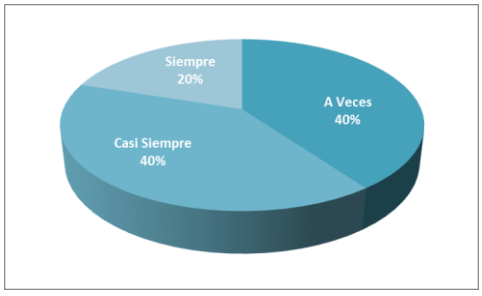

Figura 9: Pregunta "¿Cuento con la colaboración de las personas de otros departamentos?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.9 Descentralización de la toma de decisiones: la toma de decisiones se encuentra centralizada en el dueño del producto. Los mandos medios

contribuyen con sugerencias y transmitiendo las necesidades que van recabando de sus respectivas áreas.

1.1.4.10 Motivación: se observan algunos problemas de desinterés y de desmotivación en las personas. De las encuestas surgen algunos temas que quizás estén afectando tal punto. Por ejemplo,

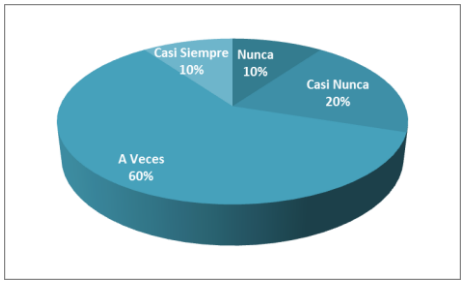

Figura 10: Pregunta "¿Las promociones internas se realizan de manera justa?" (Fuente: Encuestas realizadas por el grupo)

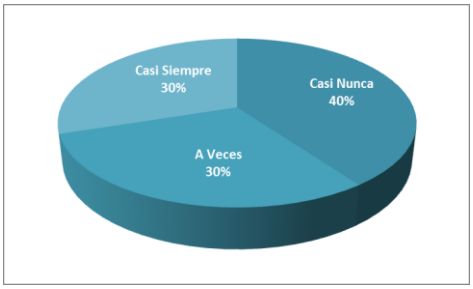

Figura 11: Pregunta "¿Creo que tengo la oportunidad de desarrollarme profesionalmente en GobFactory?" (Fuente: Encuestas realizadas por el grupo)

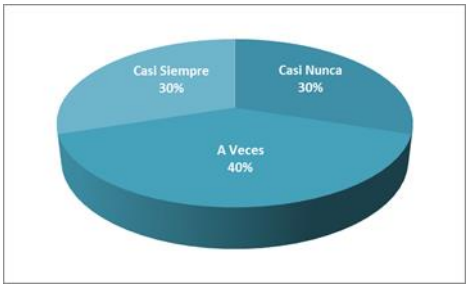

Figura 12: Pregunta "¿Pienso que si desempeño bien mi trabajo, tengo posibilidad de hacer carrera en GobFactory?" (Fuente: Encuestas realizadas por el grupo)

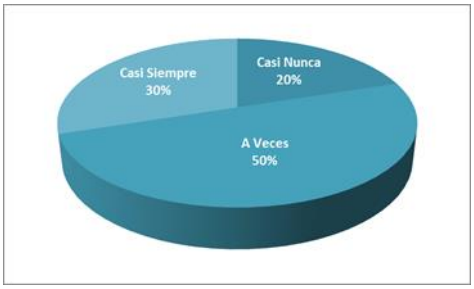

Figura 13: Pregunta "¿Recibo formación para actualizar los conocimientos de mi trabajo?" (Fuente: Encuestas realizadas por el grupo)

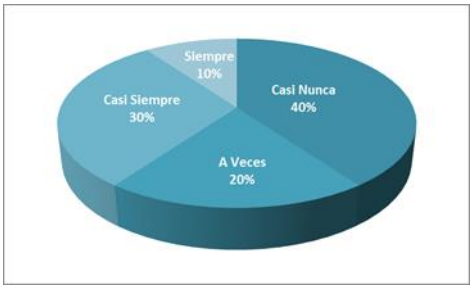

Figura 14: Pregunta "¿Los planes de formación de GobFactory se adecuan a mis necesidades de desarrollo profesional en la Compañía?" (Fuente: Encuestas realizadas por el grupo)

Generalmente, cuando el grupo está desmotivado y esto afecta al rendimiento, el Scrum Master y el encargado del área comienzan a planificar acciones que mejoren esta situación.

1.1.4.11 Formación para el puesto: al momento de contratar un miembro del equipo, no es fundamental tener experiencia previa. Durante la inducción, se transmite todo el conocimiento de negocio, de los roles, de la organización y de tecnología que requiere el puesto. Luego, la persona es encargada de informar sus necesidades de capacitación al jefe de área.

Ante las afirmaciones:

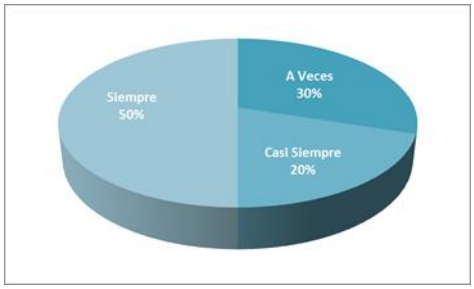

Figura 15: Pregunta "¿Cuando ingresé en la Compañía me sentí bienvenido?" (Fuente: Encuestas realizadas por el grupo)

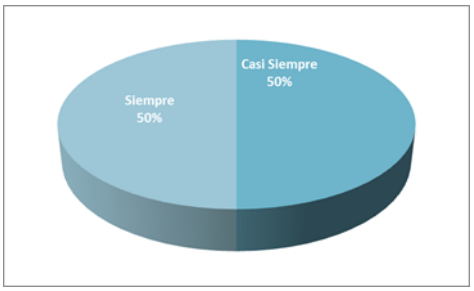

Figura 16: Pregunta "¿Al unirme a la Compañía, recibí suficiente información sobre el área donde trabajo y la función que realizo?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.12 Identificación con la empresa. Existe un importante número de personas que no se sienten identificadas con la empresa. Esto produce que no se logre el nivel de motivación esperado.
De las encuestas surgen algunas respuestas que tienen que ver con esta cuestión:

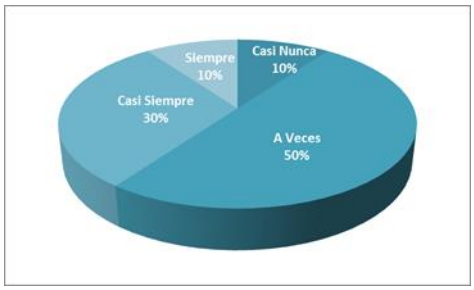

Figura 17: Pregunta "¿Me siento partícipe del proyecto de GobFactory?" (Fuente: Encuestas realizadas por el grupo)

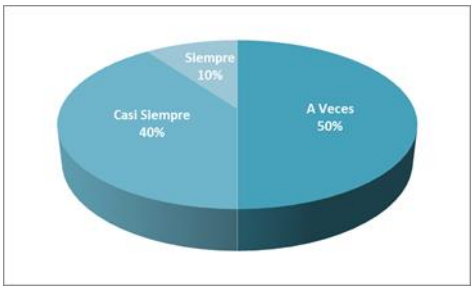

Figura 18: Pregunta "¿Actualmente estoy satisfecho con mi trabajo en GobFactory?" (Fuente: Encuestas realizadas por el grupo)

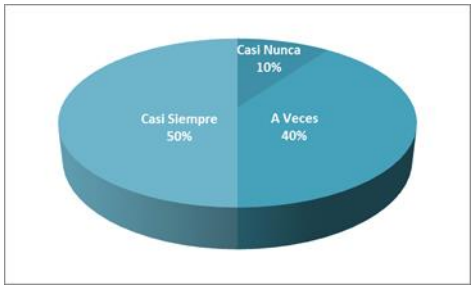

Figura 19: Pregunta "¿Me siento orgulloso de trabajar para GobFactory?" (Fuente: Encuestas realizadas por el grupo)

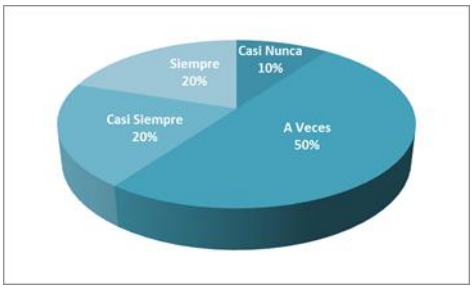

Figura 20: Pregunta "¿Pienso que GobFactory es un buen lugar para trabajar y me gustaría continuar trabajando aquí?" (Fuente: Encuestas realizadas por el grupo)

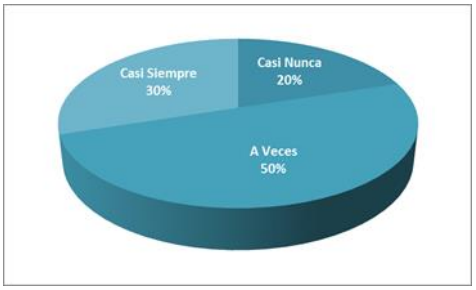

Figura 21: Pregunta "¿Recomiendo GobFactory como un lugar donde trabajar?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.13 Autonomía de decisión en la realización de su trabajo. Escasa autonomía. Todas las tareas y decisiones son informadas al Scrum Master, y las soluciones deben ser validadas y aprobadas por el dueño del producto.

Ante las afirmaciones:

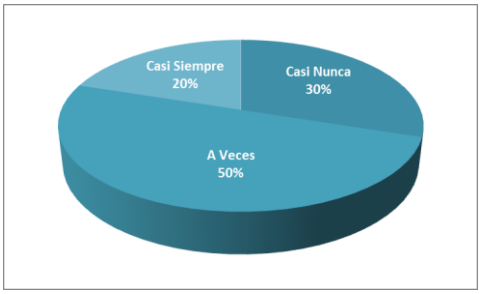

Figura 22: Pregunta "¿Tengo la oportunidad de proponer nuevos proyectos o nuevas formas de realizar el trabajo?" (Fuente: Encuestas realizadas por el grupo)

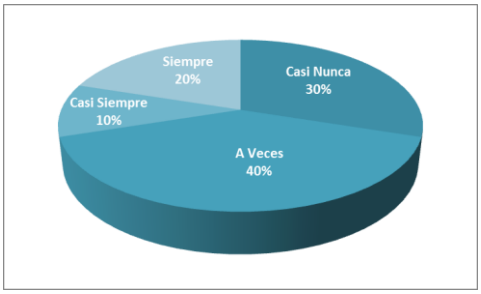

Figura 23: Pregunta "¿Tengo autonomía para llevar a cabo mi trabajo?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.14 Control sobre su trabajo. El control sobre el avance de las tareas las realiza el Scrum Master, el cual lleva un seguimiento diario, más allá de la meeting daily. En base a la información recabada y observada por el Scrum Master, el jefe informa los avances al dueño del producto en una reunión que se realiza al final del día. Éste último tiene información actualizada de todo el personal.

De las encuestas surge:

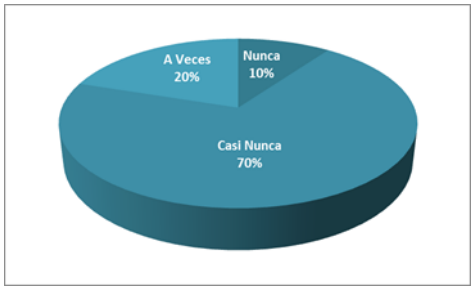

Figura 24: Pregunta "¿Mi responsable me proporciona periódicamente información sobre mi desempeño?" (Fuente: Encuestas realizadas por el grupo

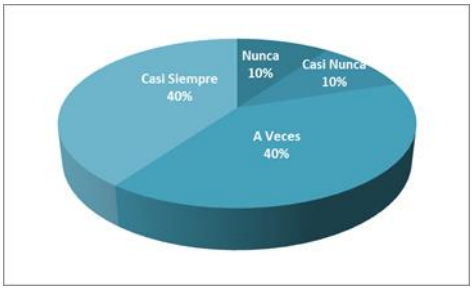

Figura 25: Pregunta "¿Mi responsable escucha mis opiniones y me hace partícipe de las decisiones?" (Fuente: Encuestas realizadas por el grupo)

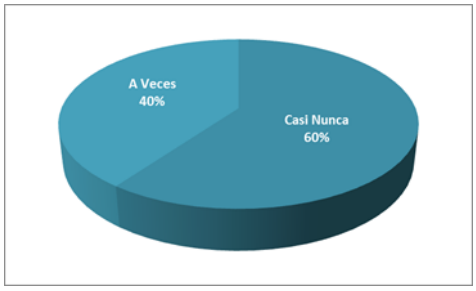

Figura 26: Pregunta "¿Mi responsable hace un seguimiento de mi Plan de Desarrollo Individual?" (Fuente: Encuestas realizadas por el grupo)

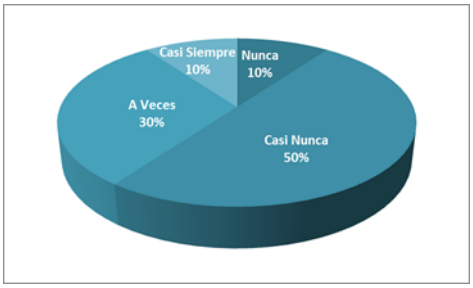

Figura 27: Pregunta "¿Puedo tomar decisiones propias sin necesidad de consultar con mi jefe?" (Fuente: Encuestas realizadas por el grupo)

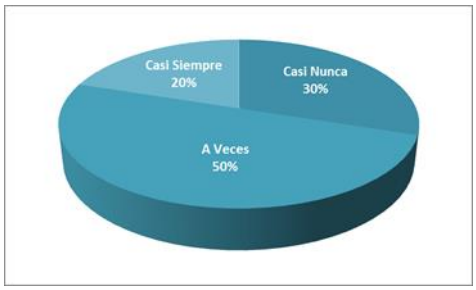

Figura 28: Pregunta "¿Mi responsable es claro y específico cuando define mis objetivos de trabajo o los del departamento?" (Fuente: Encuestas realizadas por el grupo)

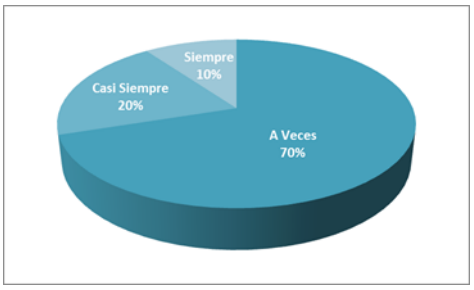

Figura 29: Pregunta "¿Mi responsable se preocupa por mantener un buen clima en el equipo?" (Fuente: Encuestas realizadas por el grupo)

1.1.4.15 Promoción. La empresa busca permanentemente la posibilidad de acceder a subsidios para proyectos de emprendedores del sector del software y los servicios informáticos. Por ejemplo, en 2015 tuvo la posibilidad de acceder al FONSOFT (Fondo Fiduciario de Promoción de la Industria del Software), otorgado por el Ministerio de Ciencia, Tecnología e Innovación Productiva.

1.1.5 Clima organizacional

Para detectar posibles aspectos que puedan estar impactando de manera importante al ambiente laboral, se analizaran algunas cuestiones generales relacionadas a los siguientes ítems:

1.1.5.1 Vinculación con las tareas. En general, todas las personas se encuentran comprometidas con su rol, con los clientes y con la empresa, lo que lleva a que se esfuercen por alcanzar los mejores resultados. Hay quienes tienen mayor empatía por algunas funciones, temas o herramientas que por sobre otras. Por ejemplo, los analistas funcionales son también analistas de prueba, es decir, deben cumplir dos funciones. No todos se sienten cómodos realizando ambas tareas. Su motivación podría ser mayor si se dedicara solamente a uno de los dos roles.

En el equipo de desarrollo existen personas con una antigüedad superior a 5 años que han atravesado diferentes cambios a nivel organizacional, comercial y metodológico. Estas personas son flexibles a los cambios pero se adaptan más lentamente que el resto. Por muchos años trabajaron bajo un modelo rígido y exigente (ciclo de vida en cascada con CMMI nivel 2), donde sus funciones eran sistemáticas, con escasa innovación y poca interacción. Al pasar a una metodología menos estructurada, donde la creatividad y la innovación son claves, estas personas tuvieron que romper con su mecanización y estar abiertos a un ambiente de mayor flexibilidad, cooperación e interacción, y como todo cambio tiene un periodo de adaptación. Los nuevos miembros del equipo son personas jóvenes, algunos provenientes de otras empresas de tecnología, que tienen “asumido” el trabajo cooperativo y se adaptan rápidamente a los cambios sin mayores apegos.

1.1.5.2 Obstaculización. Los consultores son pocos, tienen muchos viajes semanales y muchas llamadas telefónicas de los clientes internos, se ven agobiados y enfocados en los clientes externos y en menor grado los internos y a las tareas administrativas. Si bien, existe buena predisposición con el soporte interno, sus tiempos los limitan. En general, tienen tareas

administrativas que cumplir dado que además de las funciones propias del rol, deben estar atentos en la presentación de documentación de cada viaje para la rendición contable, de los contratos con los clientes, de los vehículos asignados para trasladarse hasta los clientes, etc.
En cuanto al resto de la planta, sus funciones se acotan a su rol. En cuanto al equipo de desarrollo, sus obstáculos están dados por la herramienta de trabajo (TFS), con la manera de asumir los roles, con las temáticas de sus desarrollos, etc.

1.1.5.3 Relaciones Sociales. El personal, en general, tiene una buena relación, trato cordial y respetuoso. Existe un ambiente bastante informal, donde los gerentes y empleados de todas las áreas comparten espacios comunes de charlas y actividades extra laborales, dentro y fuera de la empresa.

1.1.5.4 Alejamiento. Hay un comportamiento formal e impersonal que denota una distancia entre jefes y empleados cuando se trata de lo estrictamente laboral. De las entrevistas surge que hay una diferenciación en el trato de los jefes hacia algunos empleados.

Algunas de las opiniones al respecto vertidas en las encuestas:

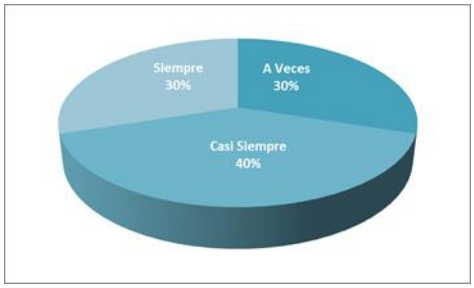

Figura 30: Pregunta "¿Mantengo una buena relación con mi responsable?" (Fuente: Encuestas realizadas por el grupo)

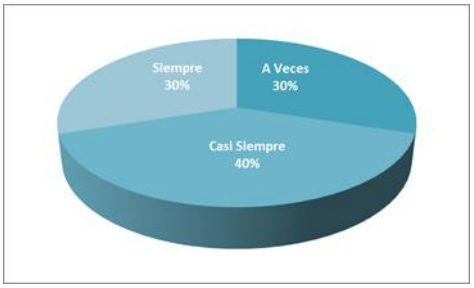

Figura 31: Pregunta "¿Mi responsable es un referente en la Compañía?" (Fuente: Encuestas realizadas por el grupo)

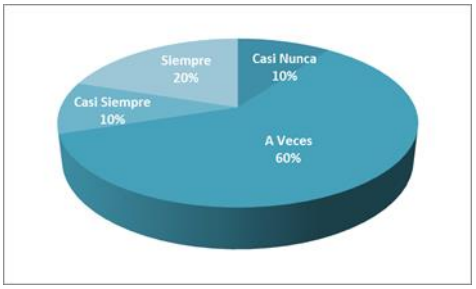

Figura 32: Pregunta "¿Mi responsable se preocupa por transmitir los valores, misión y objetivos de GobFactory?" (Fuente: Encuestas realizadas por el grupo)

1.1.5.5 Énfasis en la producción. Si bien el énfasis está puesto sobre la producción (respuestas rápidas al cliente externo e interno), la empresa admite falencias administrativas y escaso esfuerzo e inversión en innovación. Se trabaja para mejorar estos aspectos.

1.1.5.6 Empuje. El comportamiento está enfocado en cumplir con las necesidades del cliente interno y externo, sin atender demasiado a las necesidades de cada persona. No se hacen devoluciones de desempeño a las personas.

De las encuestas surge:

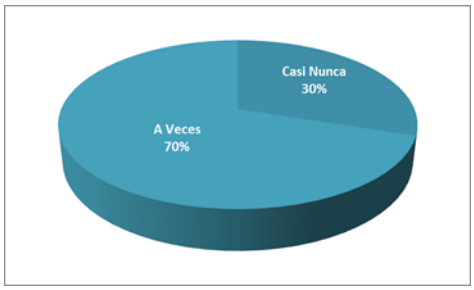

Figura 33: Pregunta "¿Mi trabajo es reconocido y valorado?" (Fuente: Encuestas realizadas por el grupo)

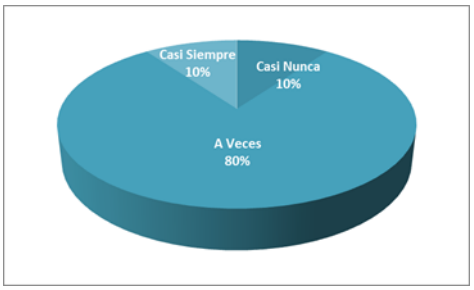

Figura 34: Pregunta "¿GobFactory me da/ofrece la oportunidad de trabajar en proyectos/actividades que suponen nuevos retos?" (Fuente: Encuestas realizadas por el grupo)

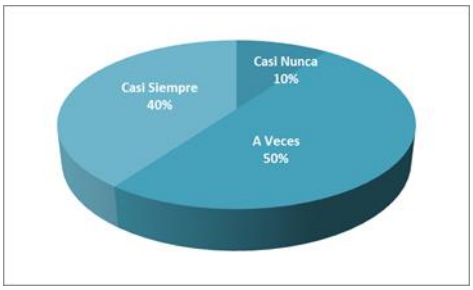

Figura 35: Pregunta "¿Mi trabajo me ofrece retos y la oportunidad de seguir mejorando?" (Fuente: Encuestas realizadas por el grupo)

1.1.5.7 Consideración. La empresa reconoce que se encuentra enfocada en el negocio y no tanto en lo humano. Por ello, se está comenzando a trabajar para mejorar estos aspectos. Se solicitó asesoramiento y colaboración a una consultora externa.

De las encuestas surge lo siguiente:

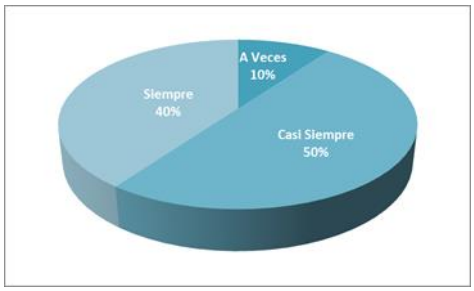

Figura 36: Pregunta "¿Mi responsable respeta las diferencias de cultura, sexo, religión…?" (Fuente: Encuestas realizadas por el grupo)

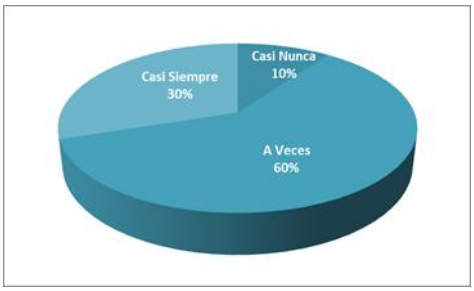

Figura 37: Pregunta "¿Mi responsable me trata justamente y evita cualquier tipo de favoritismos?" (Fuente: Encuestas realizadas por el grupo)

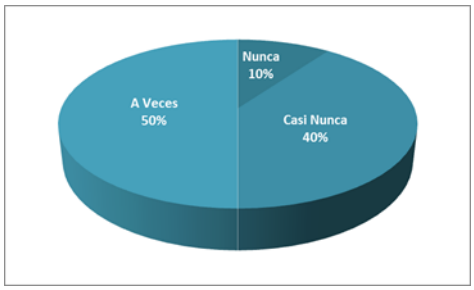

Figura 38: Pregunta "¿Mi responsable se preocupa por conocer mis necesidades e intereses?" (Fuente: Encuestas realizadas por el grupo)

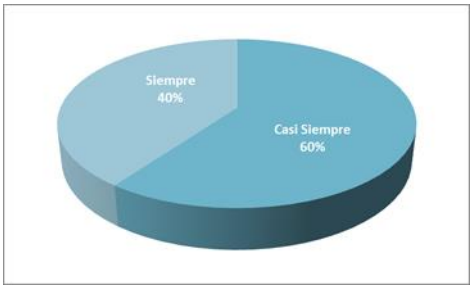

Figura 39: Pregunta "¿Las condiciones de espacio, ruido, temperatura, iluminación, tecnología... me permiten desempeñar mi trabajo con normalidad?" (Fuente: Encuestas realizadas por el grupo)

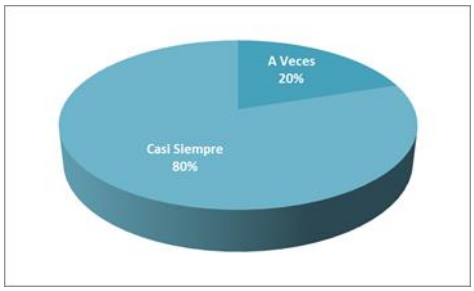

Figura 40: Pregunta "¿Considero que existe un buen ambiente de trabajo?" (Fuente: Encuestas realizadas por el grupo)

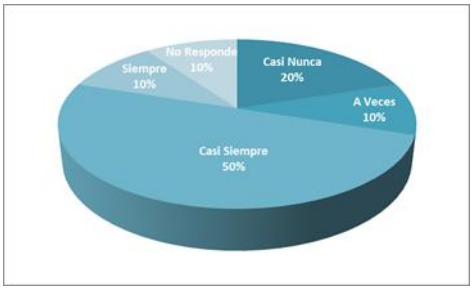

Figura 41: Pregunta "¿Estoy satisfecho con los beneficios sociales extras?" (Fuente: Encuestas realizadas por el grupo)

1.1.5.8 Estructura. Si bien hay diferencias sutiles en algunas áreas de la empresa, la estructura impone ciertas reglas y procedimientos que los grupos tienen que cumplir. De acuerdo al área de la empresa, éstas pueden ser más o menos estrictas. Actualmente, los procedimientos más estrictos se refieren a los horarios de ingreso y egreso, permisos/licencias y horas extras.

El área de Atención al Cliente y de Administración, son las que más procedimientos deben cumplimentar, ya que deben dejar registrada cada una de las tareas que realizan (Ej. Llamadas entrantes y salientes a los

clientes, compras de insumos, seguimiento de la flota de los vehículos de la empresa, seguimiento de los clientes, gestión de RRHH interno, etc.). Esto se debe a que el impacto es a nivel organizacional, de negocio, comercial, contable y humano.

En cuanto al área de desarrollo, sus procedimientos son más informales. Sólo cumplimentan, a nivel organizacional, los permisos y los horarios. En cuanto a las tareas diarias, se refieren al seguimiento de los proyectos y documentación asociada.

1.1.5.9 Responsabilidad. Las personas tienen escasa autonomía. Todas las decisiones deben ser definidas en conjunto con los gerentes y/o jefes medios. Las decisiones afines a las tareas diarias son más independientes, salvo que estén relacionadas al producto y/o afecten a los clientes.

1.1.5.10 Recompensa. No existen recompensas por buen desempeño. No obstante, las personas que demuestran un alto compromiso con la empresa suelen tener mayores flexibilidades ante pedidos puntuales.

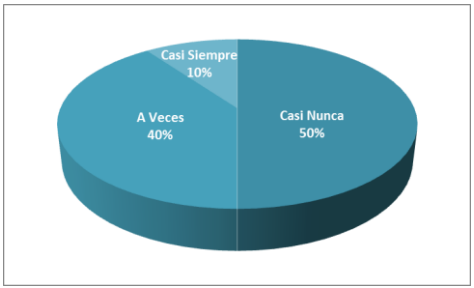

Figura 42: Pregunta "¿Mi responsable me felicita cuando realizo bien mi trabajo?" (Fuente: Encuestas realizadas por el grupo)

1.1.5.11 Apoyo. Existe buena predisposición de todo el personal (incluida la gerencia) de brindar apoyo y colaboración al resto de los compañeros. No obstante, el soporte proveniente de la gerencia y de Atención al Cliente (consultores) hacia el equipo de desarrollo, se ve afectado por las prioridades y compromisos diarios (cuestiones de negocio, atención al cliente, aspectos administrativos. El soporte interno al equipo de desarrollo no es la prioridad salvo que se trate de un tema que debe ser desplegado con urgencia.

1.1.5.12 Conflicto. La empresa se encuentra trabajando en mejorar los aspectos que generan problemas. Se dio el paso más importante: reconocer que existen. Se busca que el personal tenga predisposición y confianza en expresar todo aquello considerado negativo para el desempeño de las tareas o para las comunicaciones internas, que mejoren las relaciones

interpersonales, que mejoren la manera de hacer el trabajo, que logre que las personas se sientan cómodas al momento de desarrollar sus tareas, etc.

1.1.5.13 Formalización. En general, existe documentación de todos aquellos aspectos administrativos y organizacionales. También, existe documentación del proceso de desarrollo implementado. Esta información se encuentra disponible para todo el personal. No obstante, se encuentra desactualizada ya que algunas tareas que dejaron de realizarse o fueron modificadas siguen estando en la documentación como en un principio.

1.1.5.14 Adecuación del plan de inicio de iteración. No existen planes a nivel organizacional que involucren a todas las áreas. Los planes actuales son a nivel estratégico y comercial.

En cuanto al área de desarrollo, existe un documento de inicio de iteración que establece las pautas y objetivos en cada iteración.

1.1.5.15 Tolerancia de errores. Los errores detectados en el cliente y que fueron consecuencia de defectos no identificados durante los ciclos de pruebas o en el reléase, implican una fuerte reacción por parte de la gerencia y/o mandos medios con quien fuera el responsable. En ocasiones este reclamo o llamada de atención se hace de modo público, exponiendo a la persona.
Cuando la presencia de errores se hace frecuente, se realizan reuniones donde se intenta detectar todos los causantes de fallas.

1.2 Justificación

En los últimos años, se han presentado algunas dificultades que han impactado en el servicio ofrecido a los clientes. Esto se ve manifestado en demoras en el tratamiento de los pedidos, entregables defectuosos, requerimientos mal interpretados, personal desmotivado, entre otros.

Teniendo en cuenta a M. Piattini – F. García (2003), para poder analizar la calidad del producto se deben tener en cuenta los factores principales que influyen en su calidad: gestión, equipo de desarrollo (profesionalización), infraestructura, metodología, proceso, producto resultante de ese equipo siguiendo ese proceso y arquitectura. “Del proceso de software, y de su seguimiento por parte del equipo, depende la calidad del producto software. Por lo tanto, para lograr la calidad del producto hay que combinar la calidad de los recursos humanos con la calidad del proceso” (H. Oktaba, G. Ibargüengoitia González, 2003).

Desde el punto de vista exclusivo del proceso de desarrollo, el estudio está enfocado en encontrar las causas por las cuales se presentan permanentemente fallas y desviaciones.

En cuanto a la metodología utilizada en el proceso de desarrollo, se analiza si la misma implementa las mejores prácticas de acuerdo al tipo de producto que se desarrolla en la empresa, y a la demanda de requerimientos que se maneja.

1.3 Objetivos

1.3.1 Objetivo General

Proponer mejoras en el proceso de desarrollo de software que ofrezcan un marco de trabajo adecuado para lograr incrementar la calidad del producto final.

1.3.2 Objetivos Específicos

- Identificar posibles áreas de desarrollo a mejorar.
- Detectar prácticas relevantes del proceso de desarrollo actual de WGob, que necesiten ser revisados para su optimización.
- Disminuir reclamos de clientes.
- Lograr mayor frecuencia de los despliegues.
- Mejorar la comunicación interna.
- Fortalecer el trabajo en equipo.
- Propiciar la mejora continua.
- Generar condiciones que permitan tomar mediciones.

1.4 Alcance del trabajo y pertinencia (factibilidad)

1.4.1 Objeto de Estudio

El trabajo se enfoca en el proceso de desarrollo utilizado en la fabricación del producto de software WGob fabricado en la empresa GobFactory.

1.4.2 Alcance del Trabajo

Se analiza el proceso desde el momento en que las solicitudes de requerimientos son capturadas por la empresa hasta que la solución se encuentra lista para ser incluida en el release. (Ver figura *Diagrama del Proceso Actual del Producto WGob v7.0*, página 60)

1.5 Estudio de Factibilidad

A fines de determinar la factibilidad de realizar este trabajo, se analiza la posibilidad desde sus tres perspectivas: Técnica, Operativa y Económica. Se cuenta con información suficiente para realizar cada tipo de análisis.

1.5.1 Factibilidad Técnica

A través de este estudio se busca obtener suficientes datos para determinar si la empresa posee los recursos tecnológicos apropiados para el desarrollo del trabajo, permitiendo identificar la necesidad (o no) de actualizar y/o adquirir nuevo hardware, o software, que aseguren un adecuado desempeño de las actividades.
El análisis se realizó aplicando las técnicas de entrevistas y observación directa.
Se logró concluir que la tecnología existente, tanto hardware como software, es adecuada y aporta un ambiente de trabajo óptimo. La empresa no necesita invertir en el corto plazo en la adquisición de nuevos equipos, ni en la mejora de los existentes. Ofrece un ambiente y entorno de trabajo adecuado en cuanto a instalaciones, equipos, redes, servidores, comunicaciones y software. Se observa, además, que la empresa tiene como política actualizar constantemente el equipamiento, asegurándose que cada persona lleve a cabo sus funciones sin inconvenientes.

En cuanto al software, la empresa trabaja actualmente con dos herramientas principales. La primera, es Team Fundation Server (TFS) utilizada para la gestión de los proyectos, para la administración del repositorio y de las herramientas de especificación, prueba y desarrollo de software. La segunda, es una herramienta de desarrollo propio para la gestión de las solicitudes de requerimientos, de sus componentes de código y de su correspondiente trazabilidad. (Ver archivo adjunto: Hardware y Software disponible.docx)

Desde el punto de vista de los recursos humanos, el personal que trabaja en la empresa, específicamente los involucrados en el desarrollo de software, poseen los conocimientos necesarios para responder idóneamente a sus funciones. Recientemente, se incrementó la capacidad del equipo de desarrollo, incorporando tres (3) personas que dan soporte a los roles principales del proceso de desarrollo (análisis, testing y codificación).
De acuerdo a lo mencionado anteriormente, se concluye que el proyecto es factible técnicamente.

1.5.2 Factibilidad Operativa

De acuerdo a las entrevistas realizadas en los primeros encuentros, se estima que buscar la mejora en el proceso de desarrollo, será positivo y motivador. Dado que el

personal involucrado se encuentra familiarizado con la metodología y que manifiesta su interés en que se produzcan cambios que logren un proceso más acorde a las necesidades de desarrollo actual, se espera predisposición y apertura a todo lo que involucre este proceso y a su metodología.
Considerando el ambiente de trabajo, se puede decir que el proyecto es factible desde el punto de vista operativo.

1.5.3 Factibilidad Económica

La empresa no tiene necesidad de invertir en personal extra ni en recursos técnicos para encarar este mejoramiento en sus procesos. En la actualidad cuenta con los perfiles necesarios en cantidad e idoneidad.
Este trabajo involucra mejorar el proceso manteniéndose bajo la misma metodología. Dado que la metodología es conocida por todo el personal, este proyecto no implica invertir en capacitación sobre la misma. Ante propuestas de ajustes, éstos no tendrían gran impacto sobre las tareas actuales y tampoco implicarían la necesidad de realizar nuevos desarrollos específicos de aplicaciones de gestión. Consiste en evaluar las prácticas de trabajo utilizadas y proponer ajustes, reemplazar o incorporar nuevos conceptos y/o prácticas que permitan aumentar el desempeño y rendimiento del equipo.

En cuanto a la capacidad de producción, ésta puede verse afectada dado que ante la necesidad de aplicar mejoras o ajustes, el personal tendría un período de adaptabilidad compartida con sus tareas habituales. Por ello, el costo económico para la empresa es reducido y dependiendo de la carga de trabajo pueden ser necesarias pagar algunas horas extras u optimizar la pila del sprint, lo cual es un sacrificio que la empresa está dispuesta a afrontar, dado que se espera que los beneficios de un proceso adecuado en las futuras iteraciones compensen rápidamente la inversión, tanto para el equipo como para la empresa.

Considerando los beneficios que se obtendrán al realizar este trabajo y los escasos costos a los que incurrirá, se concluye que el proyecto también es factible económicamente.

1.6 Metodología a utilizar en el desarrollo del trabajo

Para poder abordar la temática se llevarán a cabo diferentes tipos de investigación, teniendo en cuenta las etapas y el propósito a lograr. Es decir, resulta necesario llevar adelante un abordaje que tenga en cuenta aspectos de la investigación descriptiva, exploratoria y explicativa.
Como técnicas básicas de investigación se utilizarán: estudio de campo (con base en la observación directa), encuestas y entrevistas, obteniendo como resultado información tanto cualitativa como cuantitativa sobre la situación real del proyecto,

expectativas del equipo, comunicación interna, colaboración, capacitación, identificación con la empresa, autonomía, autogestión, clima de trabajo, promociones, etc.

1.6.1 Investigación Exploratoria

Sus resultados ofrecen la posibilidad de lograr una visión aproximada de la situación real, es decir, nos ayudará a obtener un nivel superficial de conocimiento, a familiarizarnos con la situación. En nuestro caso, nos permitirá formular con mayor precisión el problema que será objeto de investigación y a definir con más claridad, las preguntas a realizar.
El estudio de campo nos dará una visión de cómo actúa el equipo durante la ocurrencia de una iteración desde su inicio hasta su fin.

Las entrevistas, por su parte, nos darán la posibilidad de un contacto personal con la gerencia, el dueño del producto y con la persona que cumple el rol de Scrum Master. En ellas, podremos explicar el propósito del estudio, especificar la información que se necesitará, las actividades previstas, etc. También se solicitarán los permisos pertinentes.
Ayudan a identificar los objetivos y políticas de la empresa, a conocer el proceso definido en el área de desarrollo y los criterios aplicados sobre la metodología, negocio, cliente, necesidades, etc. Estas investigaciones nos servirán como base para la posterior realización de la investigación descriptiva.

1.6.2 Investigación descriptiva o diagnóstica

Posibilitará definir los fenómenos o situaciones concretas mostrando sus rasgos sobresalientes, desviaciones, diferencias, etc.
Bunge explica que la descripción nos permite determinar con exactitud de qué se trata, cómo es la situación, dónde se sucede, quiénes intervienen, interrelaciones de hechos, objetos, sujetos, etc.

La observación directa da acceso a participar en el circuito de trabajo, en todas sus etapas. Se analizará la documentación digital disponible de las distintas iteraciones del proyecto.
Las entrevistas, en este caso, permitirán diagnosticar el nivel de conocimiento de las personas sobre la metodología de trabajo, sobre el producto, sobre el proceso, sobre las herramientas, y de sus funciones dentro del proceso de desarrollo actual.

Los cuestionarios brindarán información sobre las necesidades del equipo, sus propuestas de mejoras, cuáles son las prácticas que a su criterio consideran innecesarias o irrelevantes, intereses, expectativas, etc.

Se recogerán datos que expongan y resuman la información de manera cuidadosa y luego se analizarán los resultados, a fin de extraer generalizaciones que resulten significativas.

1.6.3 Investigación Explicativa

Se buscará el motivo del por qué se suceden los hechos, estableciendo relaciones causa-efecto. Los resultados y conclusiones de este tipo de investigación otorgarán el nivel más profundo de conocimiento y entendimiento sobre los procesos y la situación en general.

Se tendrán dos elementos:

- *Lo que se quiere intentar responder*. Se tratará de identificar cuáles son los desvíos de los procesos que se llevan a cabo actualmente y que forman parte del problema que genera la pregunta a explicar. Se deberá comparar como se desarrolla actualmente la metodología dentro de la empresa versus el marco de trabajo establecido por SCRUM.

- *La explicación* de los hechos particulares que suceden en la empresa (aspectos de implementación de la metodología, proceso, plan, etc.). Surgirá de los antecedentes, definiciones, especificaciones de la metodología, visión de negocio de la empresa, suposiciones y otros enunciados que expresan las regularidades que deberían suceder.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Capítulo 2: Marco Teórico

GobFactory es considerada una fábrica de software (SF) dado que está especializada en una línea de producto concreta, en este caso, software de Gestión para la Administración Pública.

A los fines de enmarcar los objetivos a tener en cuenta para una SF, tomaremos los definidos por Hitachi Software Works:

- "Mejora de la productividad y fiabilidad por medio de la estandarización y control de procesos.
- Transformación del software de un servicio desestructurado a un producto con un nivel de calidad garantizado"

El presente trabajo analiza el modo en que se entiende y se implementa el proceso de desarrollo de software. Se considera un proceso como efectivo, si proporciona normas para el desarrollo eficiente de software de calidad, a la vez que incorpora las mejores prácticas de la tecnología actual. Este proceso debe permitir una visión y cultura común a todos los involucrados, reduciendo la incertidumbre, el riesgo y logrando un proyecto más predecible. En esto juega un papel importante el desarrollo de metodologías y modelos.

R. Gacitúa (2003), plantea que una metodología impone un proceso de forma disciplinada sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos, y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema.

En la actualidad, la mayoría de las empresas están adoptando como metodologías de trabajo, las llamadas ágiles o livianas, ya que tienen como objetivo lograr mayor valor, fiabilidad y reducción de tiempos de desarrollo y permiten realizar cambios de un modo más rápido y sencillo en cualquier fase del proyecto. En este caso, GobFactory aplica SCRUM que permite abordar proyectos complejos desarrollados en entornos dinámicos y cambiantes de un modo flexible siendo una metodología más motivadora que las tradicionales. Está basada en entregas parciales y regulares del producto, priorizadas por las necesidades de los clientes.

2.1 Qué son las metodologías ágiles

Las Metodologías Ágiles son marcos metodológicos de trabajo que plantean mejorar la eficiencia en la producción y la calidad de los productos finales, potenciar la capacidad de respuesta ante cambios en los productos y sus definiciones, y brindar la mayor satisfacción posible al cliente, a través de la entrega temprana y la retroalimentación continua durante la construcción del producto.

Estas metodologías traen consigo diversos beneficios, pues permiten una mayor flexibilidad que las metodologías tradicionales (en cascada e iterativas), debido a que éstas son menos capaces a ajustarse a las cambiantes necesidades de los clientes, del mercado, y de los nuevos desafíos que plantea la tecnología.

El uso de procesos ágiles para la gestión de proyectos, permite en general que los clientes perciban algunos de estos beneficios:

- **Flexibilidad en el proceso y las definiciones de los productos:** Permite que el equipo de desarrollo se adapte de manera más rápida y fácil a los cambios.

- **Interacción y Comunicación:** La interacción entre los diferentes diseñadores y participantes es clave, es especialmente propicia para entornos orientados al trabajo en equipo. En el caso de la interacción constante con el cliente, esta es importante para darle tranquilidad sobre los avances del producto que recibirá (debido a que el producto se va analizando a medida que avanza).

- **Realimentación continua con el cliente:** De forma temprana el cliente recibe entregables de valor, lo que permite ver los constantes avances, logrando así, aportar en lo necesario para que el equipo vaya construyendo en la dirección correcta, inmediatamente reduce de forma drástica los errores y por lo tanto los costos inherentes a las correcciones, respondiendo a los cambios en requisitos de forma rápida y eficaz.

- **Calidad mejorada:** Las prácticas de desarrollo ágil y sus constantes interacciones, proporcionan el aprendizaje suficiente como para satisfacer las expectativas del cliente con una alta calidad. La clave se encuentra en la continuidad de la calidad, es decir, la calidad es integral al proceso, y no añadida (como en la vista clásica).

- **Cuando los proyectos no están claramente definidos:** Los requisitos del cliente se van clarificando a medida que el proyecto va avanzando.

2.2 Manifiesto Ágil

El Manifiesto Ágil comienza enumerando los principales valores del desarrollo ágil. Según el Manifiesto se valora:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas**. Las personas son el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el

equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

- **Desarrollar software que funciona más que conseguir una buena documentación**. La regla a seguir es “no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo fundamental.

- **La colaboración con el cliente más que la negociación de un contrato**. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

- **Responder a los cambios más que seguir estrictamente un plan**. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (de requisitos, de tecnología, de equipos, etc.), determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

  Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

  1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
  2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
  3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
  4. Los responsables del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
  5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
  6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
  7. El software funcionando es la medida principal de progreso.
  8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
  9. La atención continua a la calidad técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de equipos auto-organizados.
12. A intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

2.3 ¿Por qué usar Metodologías Ágiles?

Generalmente las metodologías tradicionales presentan los siguientes problemas a la hora de abordar proyectos:

- Existen costosas fases previas de especificación de requisitos, análisis y diseño. Corregir durante el desarrollo añade costos, lo que implica que se pierda flexibilidad ante los cambios.
- El proceso de desarrollo implica la utilización de una gran cantidad de documentos avalados.
- El desarrollo es más lento. Es difícil para los desarrolladores entender un sistema complejo en su globalidad.

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el costo.

2.4 SCRUM

De acuerdo a las características propias y a la complejidad de los proyectos, en algunos casos, no será posible especificarlos en detalle al comienzo ya que el problema no puede ser completamente entendido o definido. Será necesario revisar, analizar y corregir permanentemente. Para estos casos, resulta conveniente apoyarse en una metodología ágil. Las mismas aportan flexibilidad en el proceso y facilitan las definiciones de los productos permitiendo al equipo de desarrollo adaptarse a los cambios.

SCRUM es considerado un paradigma de metodología de desarrollo ágil. Define un framework para el abordaje del proceso de desarrollo de software logrando que sea más ágil y liviano, a través de la descripción de un conjunto de roles, componentes y organización de la actividad diaria. Agrupa buenas prácticas que permiten descubrir aquellas emergentes que van a ayudar a definir un equipo cada vez más eficiente en

un entorno complejo. Se considera complejo a un entorno en el que se debe aprender la forma de trabajar en equipo, por encima de la necesidad de negocio y la solución

Emplea un ciclo de vida iterativo e incremental en el cual se van liberando partes del producto periódicamente. En cada iteración y con cada nueva versión, normalmente, aumenta la funcionalidad y mejora en calidad respecto a la anterior.

Incremental implica que se desarrolla por partes el producto software, para después integrarlas a medida que se completan. Iterativo, que en cada ciclo o iteración, se revisa y mejora el producto. Es importante señalar que este ciclo no implica añadir funcionalidades en el producto, pero si revisión y mejora. Con ambos, se consigue una versión más estable del software, de más calidad, y con nuevas funcionalidades.

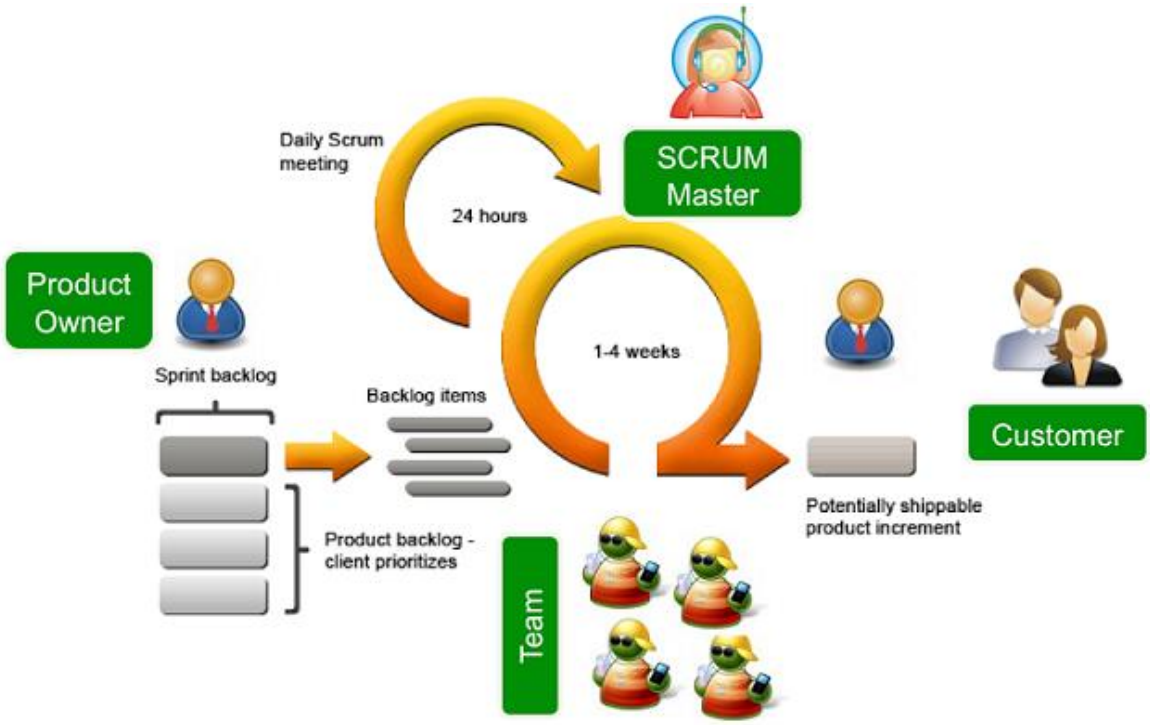

Figura 43: Ciclo de Desarrollo SCRUM (Fuente: https://santimacnet.wordpress.com)

Otro aspecto fundamental de SCRUM consiste en dividir el trabajo y organizarlo en lo que se llama Product Backlog, que es un conjunto de requisitos de alto nivel y priorizados que definen el trabajo a realizar. Estos bloques, pueden ser abordados en periodos cortos de tiempo (1-4 semanas) que se denominan Sprints.
Esta organización del proceso de creación de software permite potenciar los siguientes aspectos:

2.4.1 Agilidad: La división del trabajo en unidades funcionales pequeñas (sprints) permite maximizar la capacidad del equipo de entregar rápidamente y responder a requisitos emergentes, a la vez que aporta una visión clara del avance del proyecto y

del trabajo pendiente. También implica capacidad de responder rápidamente a las modificaciones solicitadas

2.4.2 Simplicidad: Facilita el desarrollo rápido tratando de reducir al máximo las complejidades (por ejemplo desde el punto de vista de la documentación a generar o de la organización de equipos).

2.4.3 Flexibilidad: Todo el desarrollo se contempla como un ciclo de iteraciones continuas de desarrollo, lo que facilita la introducción de modificaciones "sobre la marcha", mejorando continuamente el proceso.

2.4.4 Equipos Auto-gestionados: El planteamiento, desde el punto de vista de la organización del equipo, resulta bastante horizontal (en contraposición a una organización jerárquica inflexible), otorgando a los miembros del equipo de desarrollo un elevado grado de autonomía y auto-organización de su trabajo.

2.5 Roles en la metodología SCRUM

2.5.1 Product Owner (PO): Es la persona responsable de transmitir al equipo de desarrollo la visión del producto que se desea crear, aportando la perspectiva de negocio. Representa al resto de interesados (Stakeholders, clientes, directivos etc) en el desarrollo del producto. Sobre el Product Owner recae la responsabilidad de definir el conjunto de requerimientos (Product Backlog), de priorizarlos, y de finalmente validarlos. Es el responsable de la financiación necesaria para proyecto, del lanzamiento y del retorno de inversión. Es el responsable del éxito del producto construido.

2.5.2 Equipo de Desarrollo: Equipo multidisciplinar (de aproximadamente 5 y 8 personas) que cubre todas las habilidades necesarias para generar el resultado. Se auto-gestiona y auto-organiza, y dispone de atribuciones suficientes en la organización para tomar decisiones sobre cómo realizar su trabajo. Responsable de desarrollar el producto y de garantizar su calidad.

2.5.3 Scrum Master: El papel principal consiste en garantizar que el equipo de trabajo no tenga impedimentos u obstáculos para abordar sus tareas dentro del proyecto. Coordina con el Product Owner y es el enlace entre el equipo y éste. Facilita la demostración del sprint y la retrospectiva.
Protege al equipo del PO. Facilita las reuniones diarias. Se convierte en un coach ágil. Debe asegurar el cumplimiento de los criterios de aceptación establecidos por el Product Owner. Es el moderador en la reunión de planning.

2.5.4 Stakeholders en la metodología SCRUM: Conjunto de personas que no forman parte directa del proceso de desarrollo pero que si deben ser tenidos en

cuenta, por ser personas interesadas en el mismo, tales como directores, gerentes, comerciales etc. El PO será el encargado de recoger sus opiniones y sugerencias, y decidir si las aplica a la definición del proyecto (Product Backlog), así como decidir si invita a alguna de estas personas al proceso de revisión de entregables.

2.5.5 Usuarios: Al igual que los Stakeholders no forman parte del proceso de creación directamente (podrían estar en la fase de revisión de entregables si se considera necesario). Son los destinatarios finales de la aplicación a desarrollar, el público objetivo del mismo. Una vez que la aplicación esté completada serán los que accedan a ella con mayor frecuencia.

2.6 Componentes de la metodología SCRUM

2.6.1 Product Backlog (Definición del proyecto): Documento que contiene el conjunto funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones de desarrollo. Representa todo aquello que esperan los clientes, usuarios y en general los interesados del producto. Todo lo que suponga un trabajo que debe realizar el equipo tiene que estar reflejado en el Backlog. Es responsabilidad del Product Owner realizar esta definición y establecer las prioridades de cada requerimiento. Es un documento de alto nivel, que contiene descripciones genéricas (no detalladas), y que está sujeto a modificaciones a lo largo del desarrollo, está en continuo crecimiento y evolución. Mayor nivel de detalle a mayor prioridad, y menor nivel de detalle a menor prioridad.

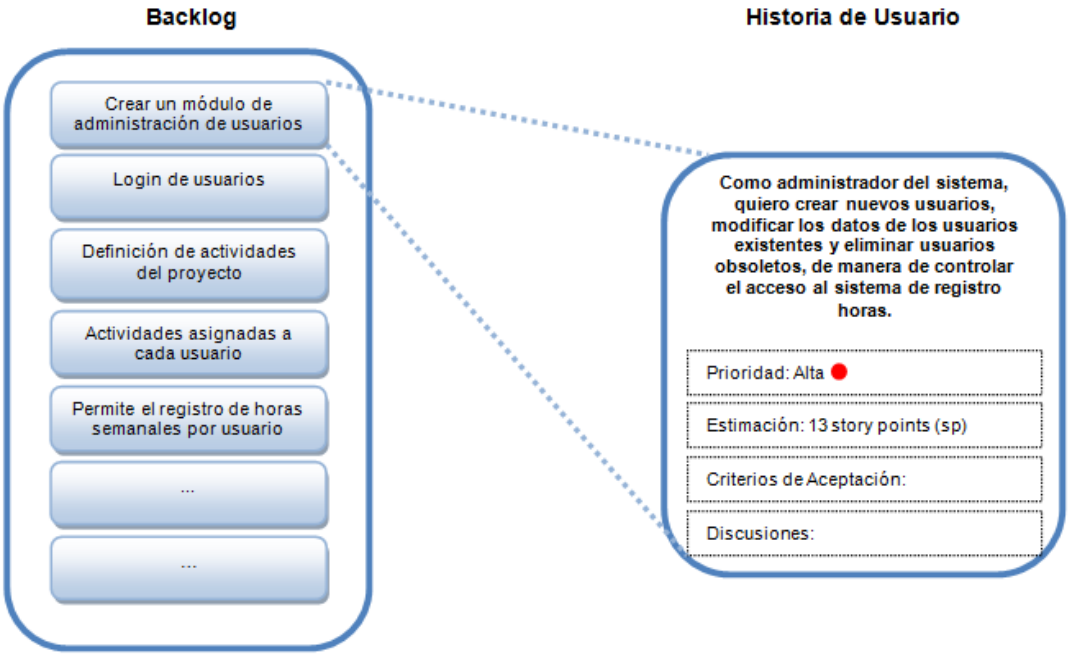

Figura 44: Ejemplo Product Backlog (Fuente: Equipo de trabajo)

2.6.2 Formato del Product Backlog

Es recomendable el formato de lista que incluya al menos la siguiente información para cada línea:

- Identificador único de la funcionalidad o trabajo
- Descripción de la funcionalidad
- Campo o sistema de priorización
- Estimación

Dependiendo del tipo de proyecto, pueden resultar recomendables otros campos:

- Observaciones
- Criterios de validación
- Persona asignada
- Nro. de Sprint en el que se realiza
- Módulo del sistema al que pertenece
- Etc.

2.6.3 Sprint Backlog (Definición del Sprint): Es la lista que descompone las funcionalidades del Product Backlog en las tareas necesarias para construir un incremento: una parte completa y operativa del producto.
En el sprint backlog se asigna a cada tarea la persona encargada de realizarla, y se indica el tiempo estimado y el pendiente. Descompone el proyecto en tareas de tamaño adecuado para determinar el avance diario e identificar riesgos y problemas sin necesidad de procesos complejos de gestión. Es una herramienta de soporte para la comunicación directa del equipo.
El tamaño de cada tarea oscila entre 4 y 16 hs de trabajo. Sólo el equipo puede modificar sus tareas durante el mismo sprint y debe ser visible para todos, idealmente en una pizarra o pared en el mismo espacio físico donde trabaja el equipo.

Las tareas o ítems del backlog del sprint deben contener al menos los siguientes campos:

- Descripción de la tarea
- Responsable de la tarea: generalmente, los miembros del equipo se "auto asignan" las tareas.
- Complejidad. sirve de guía para las estimaciones y para distribuir el trabajo entre los miembros del equipo.
- Estimación: en horas o días hombre. Representa una estimación de esfuerzo necesario.
- Discusiones: registro de todas las decisiones, especificaciones más detalladas o definiciones adicionales que se van tomando durante el sprint.

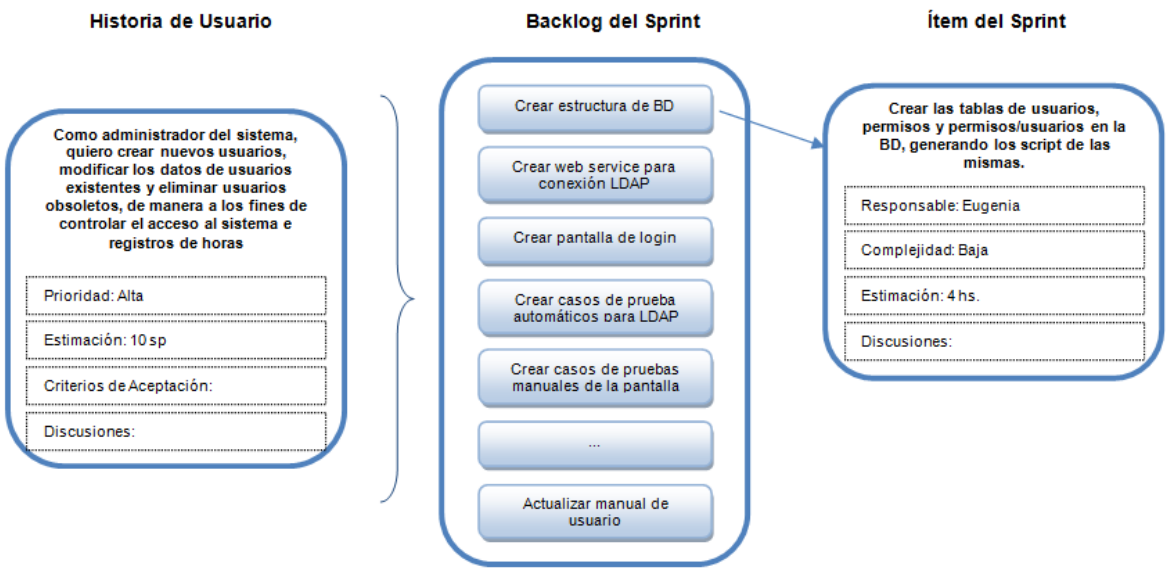

Figura 45: Estructura de ejemplo del Backlog del Sprint (Fuente: Equipo de trabajo)

Figura 46: Estructura de ejemplo de los Ítems de Backlog de Sprint (Fuente: Equipo de trabajo)

2.6.4 Cálculo de la capacidad del equipo

La capacidad del equipo es la cantidad de horas efectivas que dicho equipo tiene disponibles, para este sprint.

Miembro	Días disponibles	Horas diarias disponibles	% Asignación al proyecto	Capacidad en el sprint
Persona1	20	7	100	140
Persona2	20	5	100	100
Persona3	18	7	100	126
Persona4	20	7	50	70
Todos		**Capacidad total del equipo:**		**436**

Figura 47: Ejemplo cálculo de la capacidad del equipo (Fuente: Equipo de trabajo)

***Capacidad individual** = dias_disponibles * horas_diarias_disponibles * porcentaje_asignacion*
***Capacidad del equipo** = ∑(capacidad individual)*

2.6.5 Incremento: Es la parte de producto desarrollado en un sprint y tiene como características que está completamente terminada y operativa: en condiciones de ser entregada al cliente final.
Si el proyecto o sistema requiere documentación, o procesos de validación y verificación documentados, o con niveles de independencia que implican procesos con terceros, éstos también deben estar realizados para considerar que el producto está "**terminado**".

2.7 Reuniones de trabajo en un contexto SCRUM

2.7.1 Planificación de Sprint: Jornada de trabajo de análisis y planificación del Sprint, que involucran a todo el equipo ágil y proporciona la coordinación necesaria para lograr los objetivos y determinar las tareas que se deben cumplir con esa iteración. Esta reunión genera la Sprint Backlog. La rutina de la reunión es:

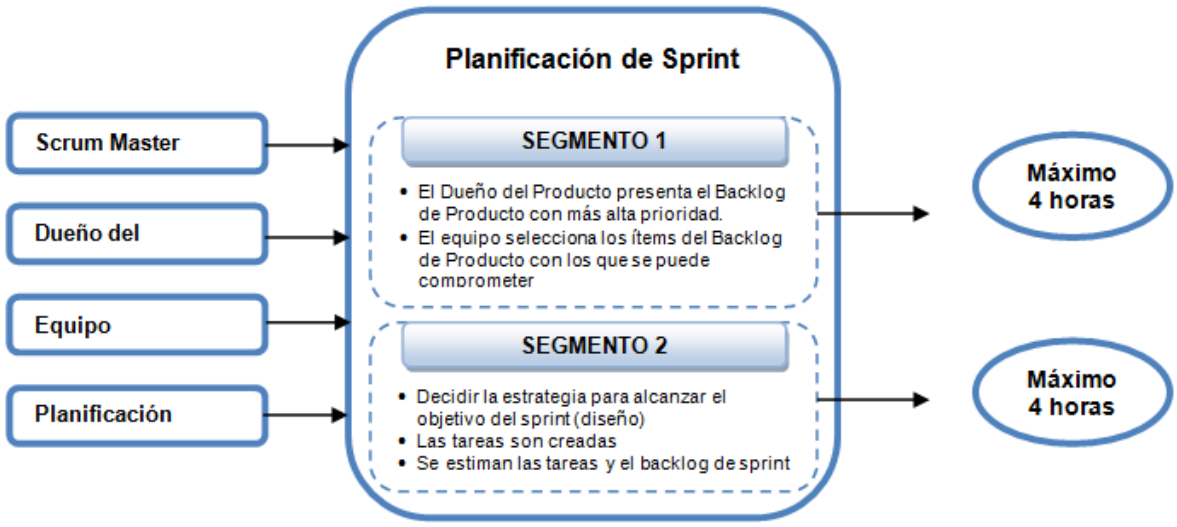

Figura 48: Ejemplo de Rutina de Reunión (Fuente: Equipo de trabajo)

2.7.2 Planning Poker: es una técnica para calcular una estimación basada en el consenso, en su mayoría utilizada para estimar el esfuerzo o el tamaño relativo de las tareas de desarrollo de software. Minimiza el anclaje (el anclaje se produce cuando un equipo discute abiertamente sus estimaciones) pidiendo a cada miembro del equipo jugar su tarjeta de estimación de manera tal que no puede ser visto por los demás jugadores. Después de que cada jugador ha seleccionado una tarjeta, todas las tarjetas son expuestas a la vez. Estas estimaciones son menos optimistas y más precisas que las estimaciones obtenidas a través de la combinación mecánica de las estimaciones individuales en las mismas tareas. Pasos:

1. Colocar una **Historia de Usuario** en el centro de la mesa
2. A cada miembro del equipo se le entrega un **mazo de 13 cartas.** Un mazo típico contiene tarjetas mostrando la secuencia de Fibonacci con los siguientes valores: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100.
3. Cada miembro **escoge un valor estimado** para la historia y pone su **carta dada vuelta** sobre la mesa para que ninguna otra persona pueda ver el valor
4. **Todos dan vueltas** las cartas al mismo tiempo
5. El moderador **identifica la dispersión** de las estimaciones y se **discuten las diferencias** tomando los extremos (el más alto y más bajo valor).
6. Se identifican las preguntas y suposiciones de ambos extremos. Comienza la segunda **ronda** (y así sucesivamente hasta que se logra el objetivo final: **una convergencia aceptable**).

2.7.3 Reunión diaria: Rápidas reuniones diarias de coordinación de equipo, lo que permite una aceleración en el entendimiento del trabajo a realizar y una mejora continua orientada a la calidad del producto. Permite dar repaso al avance de cada tarea y al trabajo previsto para la jornada. Se realiza al comienzo de cada día en que ese esté ejecutando un sprint. Es una reunión corta (no más de 30 minutos), solo interviene el equipo y cada miembro responde las siguientes preguntas:

1. ¿Qué has hecho desde la última reunión?
2. ¿Qué planeas hacer antes de la próxima reunión?
3. ¿Qué problemas has encontrado para realizar el trabajo previsto?

Reglas de las Reuniones Diarias

- Diaria
- En el mismo lugar físico
- Hora fija
- Duración máxima 15 minutos
- Participa todo el equipo
- El Scrum Master comienza la reunión en el horario convenido sin importar quién no se encuentra presente
- Los aspectos técnicos que surjan deben ser tratados en una reunión posterior donde participen sólo los miembros del equipo afectado
- No son reuniones de reporte de estado/avance al Scrum Master, sino que es un momento para que cada miembro renueve su compromiso con los demás.

2.7.4 Refinamiento: es una actividad en la que participa el equipo de desarrollo y el Product Owner y que se realiza durante todo el sprint, aunque algunos equipos prefieren concentrarla en una reunión que se realiza en cualquier momento de la iteración y en función de las necesidades. Su objetivo es obtener mayor entendimiento de los PBIs (Product Backlog Items) que se encuentran más allá del sprint actual y así dividirlos en PBIs más pequeños, si lo requieren, y estimarlos. Generalmente, se revisan y detallan aquellos PBIs que ingresaran en los próximos sprints.
También, tiene como objetivo la detección de riesgos implícitos en los PBIs que se estén analizando, y en función de ello revisar y ajustar prioridades del Product Backlog.
En caso de que se realice como una reunión, la responsabilidad de convocarlas es del Product Owner, entre una y dos veces en el sprint, facilitadas por el Scrum Master.

2.7.5 Revisión de sprint: Sesión de retroalimentación al cliente. Cada iteración de construcción de los productos incluye una revisión detallada con los clientes. Una vez concluido el ciclo de sprint se mantiene una reunión en la que se define qué parte del trabajo previsto se ha completado y qué parte permanece pendiente. En cuanto al trabajo completado se realiza una revisión (demo) del mismo al Product Owner y otros usuarios que pudiesen estar involucrados.

Reglas de la Revisión del Sprint

- Es una reunión informal
- La preparación de la presentación no debe superar 1 hora
- El equipo presenta la funcionalidad terminada (done) al Dueño del Producto y demás stakeholders
- Los miembros del equipo responden las preguntas de las stakeholders en relación a la demostración, y toman nota de los cambios propuestos
- Al finalizar la presentación, los stakeholders dan su impresión acerca del producto, cambios deseados y la prioridad de esos cambios

2.7.6 Retrospectiva de sprint: Una reunión de revisión final de cada iteración, que permite a los participantes del proceso, retroalimentar el proceso (no el producto) identificando lo que se hizo bien y lo que es susceptible de mejorar, tratando de introducir un componente de mejora continua en el proceso. Todos los miembros del equipo realizan una valoración del trabajo realizado en el último sprint, identificando puntos de mejora de cara a los siguientes a realizar. El feedback obtenido en esta reunión puede ser incluido entre las funcionalidades a construir en futuros Sprints.

Reglas de la Revisión del Sprint

- Provee una visión periódica de qué está funcionando y qué no
- Generalmente tiene una duración aproximada de 1 hora
- Se realiza al final de cada sprint
- Todo el equipo participa (Scrum Master, Dueño del Producto y Equipo)
- Se pueden utilizar varias técnicas

2.8 Métricas

Las métricas son medidas cuantitativas que permiten obtener una visión de la eficacia del proceso de software y los proyectos que se llevan a cabo utilizando ese proceso como marco de trabajo.
Hay cuatro razones para medir: caracterizar, evaluar, predecir y mejorar.
Las métricas nos ayudan a entender el proceso para intentar mejorarlo, y al producto para intentar aumentar su calidad. Fenton y Pfleeger (1997) explican que: para poder asegurar que un proceso o sus productos resultantes son de calidad o poder compararlos, es necesario asignar valores, descriptores, indicadores o algún otro mecanismo mediante el cual se pueda llevar a cabo dicha comparación. Para ello, es necesario llevar a cabo un proceso de medición de software, que en general, persigue tres objetivos fundamentales: ayudarnos a entender qué ocurre durante el desarrollo y mantenimiento, permitirnos controlar qué es lo que ocurre en los proyectos y poder mejorar los procesos y productos.

Métricas utilizadas en SCRUM

2.8.1 Gráfico Burn-down

Los gráficos Burn ayudan a visualizar el avance del proyecto en el tiempo. Burndown Chart es una herramienta del equipo que representa de manera muy sencilla el avance del Sprint. Provee un indicador diario de la velocidad (velocity) del equipo y el progreso respecto del trabajo comprometido para el sprint actual. El Scrum Master es el responsable de registrar el avance del sprint una vez por día.

Entradas:

- Trabajo Pendiente: Cantidad de Sprint Backlog ítems que no fueron comenzadas y esfuerzo restante en las que ya comenzaron.

Tendencia de Regresión Lineal:

Trabajo Pendiente + Tendencia = Trabajo pendiente del día anterior a ayer + Trabajo pendiente de ayer + Trabajo pendiente de hoy) / 3

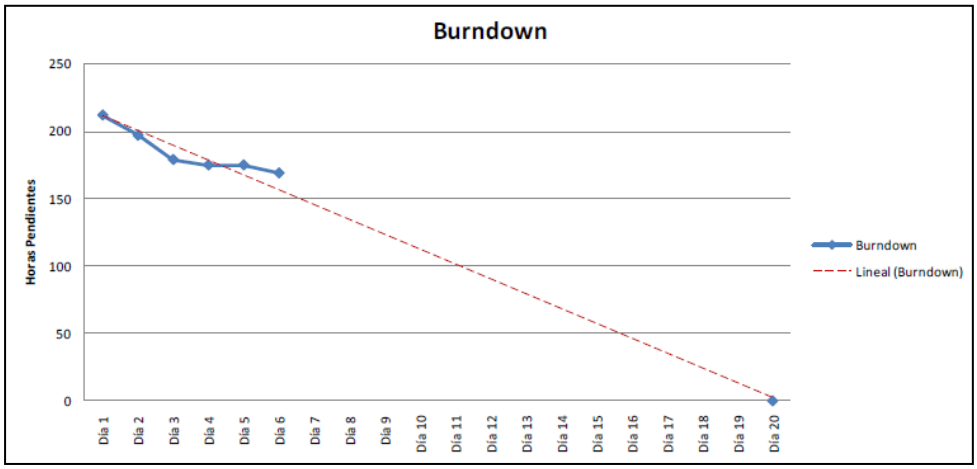

Figura 49: Ejemplo de gráfico Burn-down (Fuente: Equipo de trabajo)

Análisis:

- Cuanto esfuerzo vamos “quemando” en lo que va del Sprint?
- Haciendo una proyección, vamos a llegar a cumplir los objetivos del Sprint?

2.8.2 Gráfico Burn-Up

Herramienta de gestión y seguimiento para el dueño del producto. Presenta de un vistazo las versiones de producto previstas para el proyecto, las funcionalidades de cada una, velocidad estimada, fechas probables para cada versión, margen de error previstas en las estimaciones, y avance real.

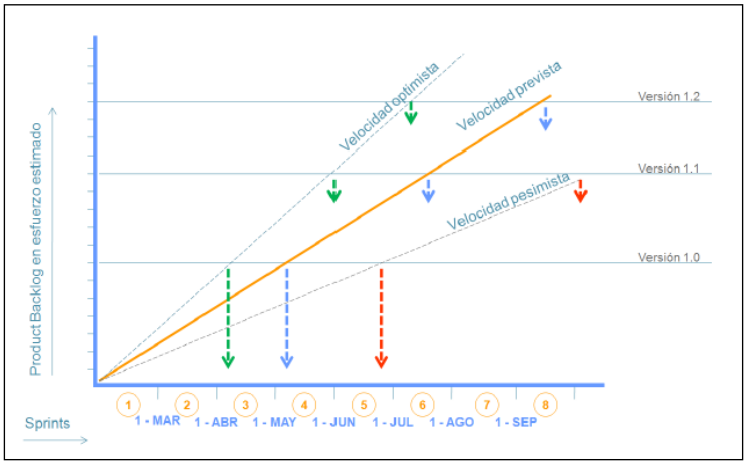

Figura 50: Ejemplo de gráfico Burn-Up (Fuente: Equipo de trabajo)

Análisis:

- Haciendo una proyección, ¿vamos a llegar a cumplir los objetivos del proyecto?

2.8.3 Velocidad

Es el número de puntos de historia que el equipo es capaz de terminar durante el Sprint. Es importante no confundir velocidad con productividad. La velocidad es una medida de la capacidad del equipo, no de lo bueno, malo o poco productivos que sean.

Se basa en la tendencia, y se obtiene el dato bueno normalmente tras el tercer o cuarto Sprint. Al principio, el equipo suele ser más optimista y selecciona más puntos de historia de los que puede terminar. Un equipo maduro que viene trabajando desde hace tiempo junto, tiene la ventaja de conocer su velocidad al inicio del proyecto.

Esta métrica permite conocer en cuantos puntos de historia se podrá comprometer el equipo para cada iteración, conociendo la velocidad a la que es capaz de trabajar.

Elementos de Entrada:

- Cantidad de Historias de Usuario por cada Sprint

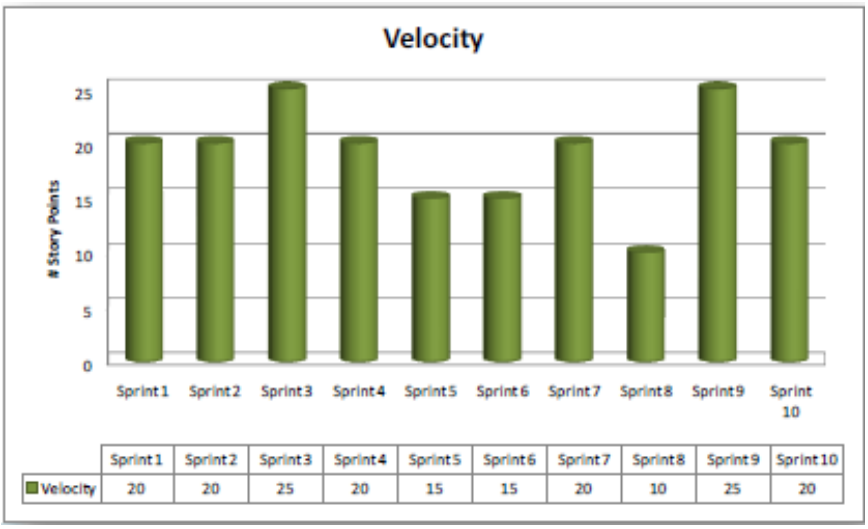

Figura 51: Ejemplo de gráfico de Velocidad (Fuente: Equipo de trabajo)

Velocidad = (nº desarrolladores del equipo *días de duración del Sprint) * Factor de Foco %

Factor de Foco (FF%) es el % del día que las personas del equipo son plenamente productivas y pueden estar dedicadas al 100% a la producción de software, sin verse afectados por otros impedimentos, reuniones o tareas auxiliares. Suele fijarse de partida entre el 60% y el 70 %, dependiendo de la madurez del equipo, y se va recalculando iteración tras iteración. Por ejemplo, supongamos los siguientes datos de partida:
Nº de desarrolladores del equipo. = 4

Factor de Foco inicial (FF) = 70 %
Días de duración del Sprint = 15

Una vez que se toma la velocidad de varios sprint, se calcula una velocidad media. Luego es posible hacer una proyección que abarque la totalidad de los sprints para estimar la velocidad total del proyecto.

Análisis:

- En general, cuántas Historias de Usuario desarrolló por Sprint?
- A qué se debe que en el Sprint N se desarrollaron menos Historias de Usuario que el promedio de Velocidad?

2.8.4 Tasa de Diferimiento

Muestra el número de Historias de Usuario diferidas por sprint en relación a la cantidad total de Historias de Usuario inicialmente comprometidos. Se utiliza para la planificación de cada sprint, y para los ajustes en el release.

Entradas:

- Trabajo Pendiente: Cantidad de Sprint Backlog Items que no fueron comenzadas y esfuerzo restante en las que ya comenzaron.

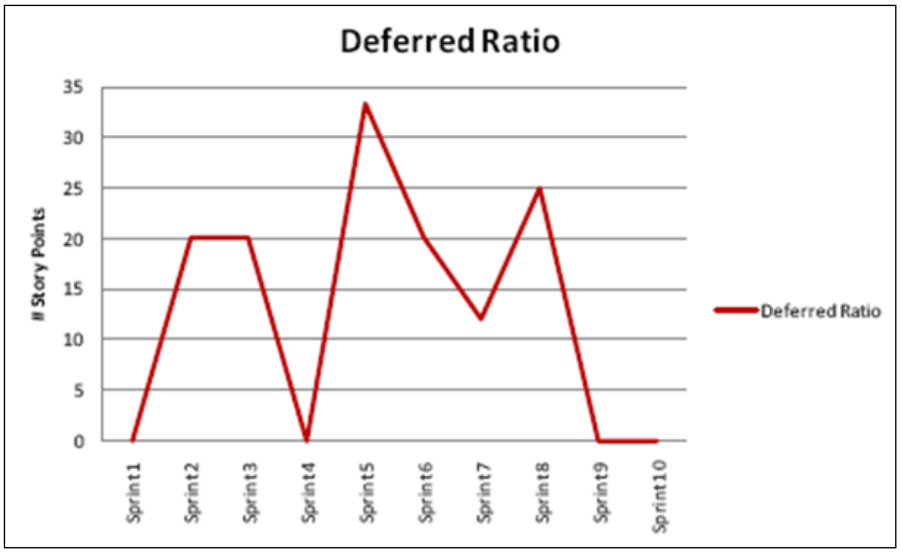

Figura 52: Ejemplo de gráfica de tasa de diferimiento (Fuente: Equipo de trabajo)

Análisis:

- Normalmente, cuántas Historias de Usuario difiero a siguientes sprints?
- ¿A qué se debe que estoy difiriendo?
- ¿Problemas en las estimaciones?

2.8.5 Gráfico Release Burn-Down

Release, es una versión del producto estable y lista para desplegar en los clientes. A partir de este momento el software se hace público. Esta técnica ayuda a determinar la fecha probable de finalización del mismo, teniendo en cuenta la performance (velocidad) del equipo y los cambios introducidos en el alcance.

La intersección de las líneas de tendencia de la "velocidad del equipo" y el "trabajo introducido al release" proveerá una aproximación a la fecha de release.

Entradas:

- Trabajo Pendiente
- Trabajos Completados
- Cambios en el Alcance

Ejemplo: Release Burn-down:

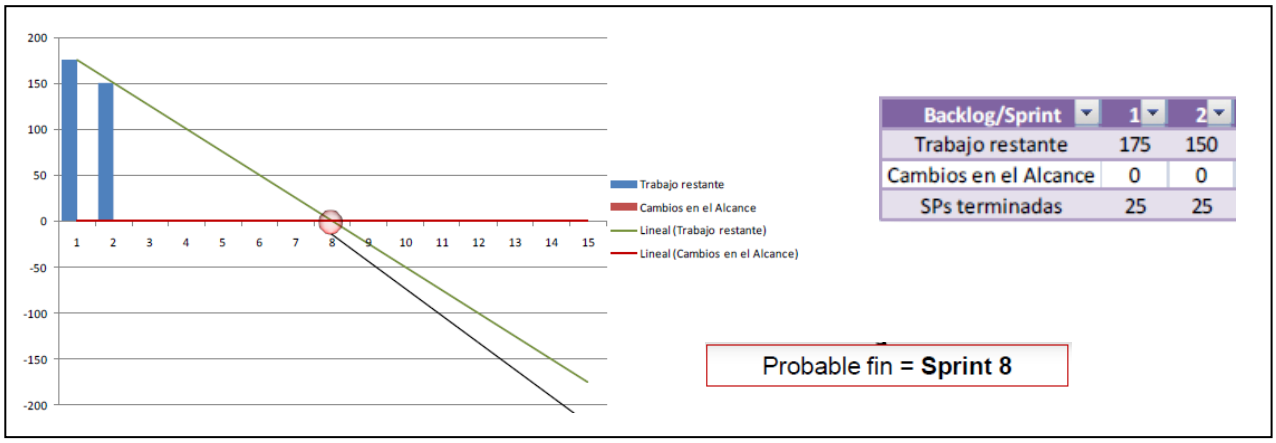

Figura 53: Ejemplo de gráfica de Release Burn-down (Fuente: Equipo de trabajo)

Cambio de Alcance:

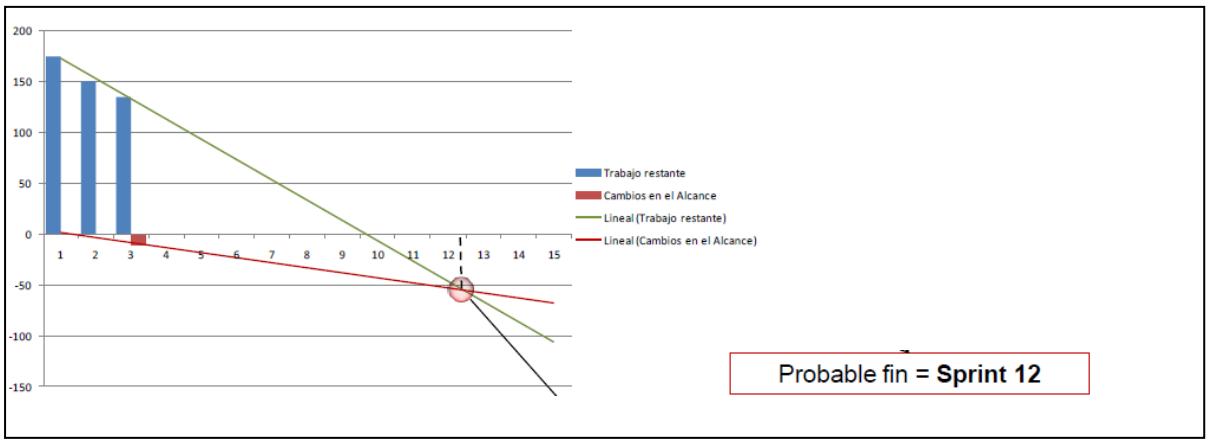

Figura 54: Ejemplo de gráfico de cambio de alcance (Fuente: Equipo de trabajo)

Cambio de Alcance y Velocidad:

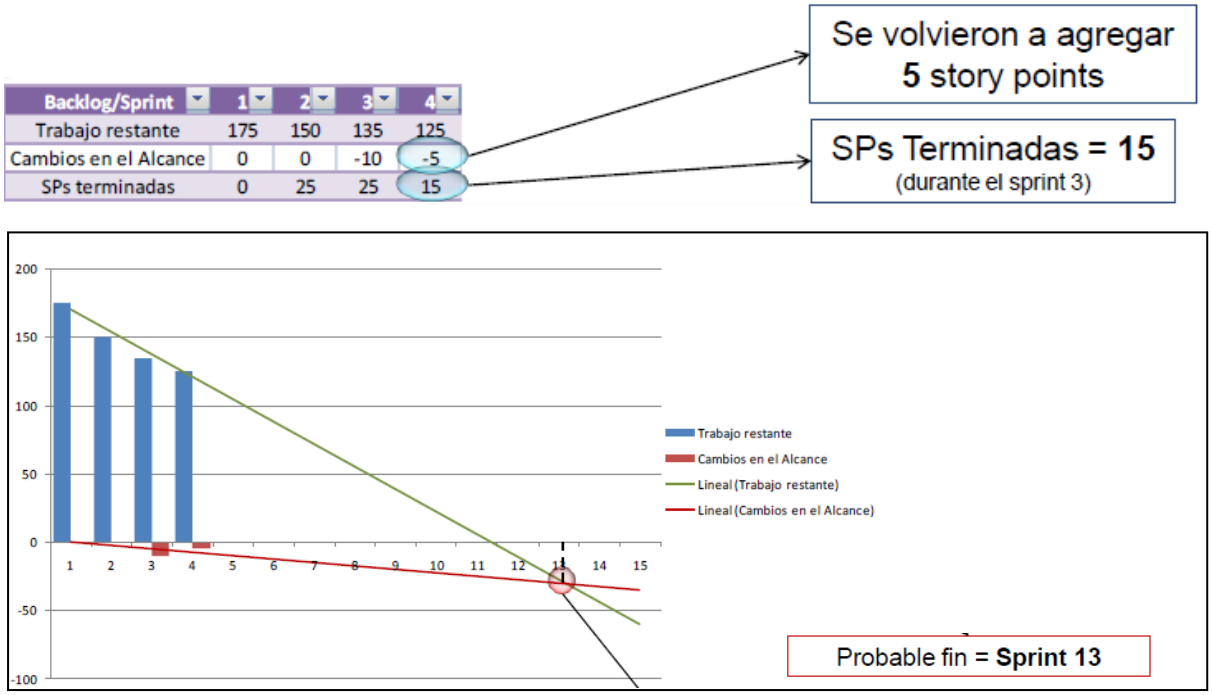

Figura 55: Ejemplo de gráfico de cambio de alcance y velocidad (Fuente: Equipo de trabajo)

Análisis:

- Como ha impactado cambios de alcance en el compromiso del Proyecto?
- Venimos adelantados o atrasados con las fechas?

2.9 Estimaciones

"Una estimación software es una predicción de cuánto tiempo durará o costará un proyecto" (McConnell 2006). El propósito de una estimación software es determinar si los objetivos y los tiempos que disponemos para realizar el proyecto, son suficientemente realistas.

Hacer una buena estimación software antes de ofertar un proyecto nos puede ayudar a detectar aquellos que no nos conviene abordar y que no son rentables.

En un proceso ágil, se modifica el modo de llevar a cabo las estimaciones con respecto a los métodos clásicos. Se da preponderancia al concepto de valor y es uno de los elementos más importantes a la hora de guiar un proceso ágil. El valor, sin tener una definición exacta, es, como decía Ron Jeffries, en The Nature of Software Development, "lo que tú quieres", más concretamente, lo que esperan los usuarios potenciales del producto que se está desarrollando. Es posible fallar en el cálculo de qué tiene más valor, e incluso el usuario potencial puede no tenerlo claro antes de empezar. Este valor puede cambiar posteriormente en avance y revisiones que se hagan con el mismo usuario. Es la dirección estratégica del producto, lo que se asume como preferencia de los usuarios. Nos ayuda a crear prototipos para mostrar e interactuar con los usuarios potenciales. "Valor frente a estimación", como decía Ron Jeffries, un proceso ágil se basa en hacer las cosas de mayor valor primero y las de menor valor luego o nunca, entregando valor pronto.

La principal premisa en la que se basa la estimación ágil es el conocimiento, la experiencia del propio equipo frente a otros métodos de estimación que se basan en la experiencia de otros equipos, es decir, en la estadística. SCRUM se conoce como método empírico.

Existen numerosos métodos de estimación software, estos se pueden clasificar en dos grandes grupos:

Los métodos heurísticos se basan en la experiencia y los principales son:

– El **método basado en juicio experto** o "a ojo", consiste básicamente en preguntar a alguien con experiencia (o sin ella) cual es en su opinión la estimación software. En algunos casos, si es la primera vez que se realiza una estimación sobre un nuevo desarrollo, puede llegar a necesitarse la ayuda de personas externas. En general, el método basado en juicio experto es el más utilizado. El riesgo de este método es la dependencia creada con el experto.

- Planning Poker: se basa en el método Wideband Delphi para descomponer el trabajo en piezas menores y luego realizar la estimación de requerimientos (o User Stories) de forma colaborativa en un Equipo. Ver Página 31.

– El **método basado en Intermediarios (Proxies):** el objetivo es estimar las líneas de código de un programa antes de implementarlo.

- Puntos de Historia (SP): este método se desarrolló como una manera de dimensionar y relacionar la complejidad de las historias de usuarios con respecto a otras.
  Como primer paso del proceso de estimación, se selecciona una historia de usuario y se define una complejidad nominal que servirá de referencia para catalogar otras historias de usuario (mayor o menor complejidad).

  Los valores a utilizar para representar la complejidad no tienen un valor absoluto sino que su valor es función de su posición en escala.
  Al comienzo se utilizaba la serie de Fibonacci: 1, 2, 3, 5, 8, 13, 21,…, luego, para evitar que se pensara que hay una precisión matemática en los valores a partir de cierto número se sustituyeron por otros aproximados: 3, 5, 8, 13, 40, 100,… (el 1 y el 2 no se recomienda utilizarlos al no tener mucha diferencia respecto al 3)
  También se pueden utilizar los valores: Extra Small, Small, Medium, Large, Extra Large.
  Como el método se basa en la comparación de historias de usuarios ya realizadas se necesita contar con una línea base de historias realizadas por el equipo.

  Los puntos de historias no miden horas: la complejidad de las historias y los puntos de historia, no se pueden comparar a horas de esfuerzo, ya que el sentido que tienen es catalogar la dificultad de la tarea. El número de horas empleados en su realización dependerá de la capacitación y/o capacidad de la persona que la lleve a cabo, la carga de trabajo del equipo, etc. y por ello su valor dependerá de cada situación.

  Medida propia de un solo equipo: los puntos de historia deben ser relativos a cada equipo de desarrollo y por ello no podemos comparar los puntos de historias medidos por un equipo con lo de otros equipos ya que las condiciones pueden ser diferentes. Es más, tampoco podemos hacer comparaciones de la velocidad de desarrollo de cada uno de los equipos por el número de puntos de historia que hayan implementado.

2.10 Testing en Metodologías Ágiles:

2.10.1 Testing de Software:

Es el proceso de verificación y validación de un producto software con el objetivo de informar a los interesados si el mismo satisface los requerimientos técnicos y del negocio para el cual fue desarrollado.

Verificación y Validación según la Norma ISO/IEC 9000

La verificación "es el proceso de evaluar un sistema o un componente para determinar si los productos de una fase de desarrollo satisfacen las condiciones impuestas al comienzo de dicha fase, es decir, comprueba la consistencia del software con respecto a especificaciones y requisitos; responde a *¿se ha construido correctamente el software?*

La validación "es el proceso de evaluación de una solución o de uno de sus componentes durante o al final de un proceso de Desarrollo de Software para determinar si se satisfacen todos los requisitos establecidos por el usuario inicialmente, es decir, comprueba si lo que se ha especificado (y construido) es lo que el usuario realmente desea; responde a *¿se ha construido el software correcto?*

"La calidad no es intangible. El propósito del testing es hacer que la calidad sea visible. El testing es la medida de la calidad del software" (Bill Hetzel, 1998).

El objetivo del Testing es prevenir defectos en un sistema, un componente o producto intermedio, logrando de esta manera aumentar la confianza del mismo. Su necesidad se debe a los siguientes motivos:

- Porque el sistema probablemente tenga defectos que debamos remover.
- **Para saber qué tanta confianza podemos tener en el sistema.**
- Porque los defectos pueden causar fallas muy costosas.
- Para cumplir de manera eficiente con los clientes.

Diferencia entre error, defecto y falla según la Norma ISO/IEC 9000

Error (bug): es una variación en los atributos deseados para la solución. Se trata de una acción humana que ingresa un resultado incorrecto (Ej. un error de programación).

Defecto: desperfecto en un componente o sistema que puede causar que el componente o sistema falle en desempeñar las funciones requeridas (Ej. una sentencia o una definición de datos incorrecta).

Falla: es un error que causa un problema en la operación de una solución, dada por la interrupción o reducción de la calidad del servicio o algo que impacta negativamente al usuario.

En resumen, un **error** introduce un **defecto** en el software que a su vez causa un **fallo** al momento de ejecutar pruebas.

El testing ágil es una práctica de pruebas de software que sigue los principios del desarrollo ágil de software. Los equipos ágiles utilizan un enfoque de "todo el equipo" al testing (todo el equipo es responsable de la calidad del producto, no sólo el tester), con la finalidad de integrar la calidad al desarrollo del producto, al contrario de un enfoque de primero fabricar el producto y luego inspeccionar para determinar su nivel de calidad. El rol del tester es el de un experto multifuncional, garante que se entregue el valor de negocio deseado por el cliente a un ritmo sostenible y continuo.

Las metodologías ágiles no ven al testing como una fase separada, sino como parte integral del Desarrollo de software al igual que la programación.

2.10.2 Principios del testing ágil

De forma similar a que el Manifiesto Ágil contiene principios que se aplican al desarrollo ágil de software, el testing ágil engloba los siguientes principios:

- **El Testing no es una fase**: El testing continuo es la única forma de garantizar avance continuo, por esto, se realiza continuamente junto con el desarrollo de software y demás actividades.

- **El Testing hace avanzar el proyecto**: proporciona retroalimentación continua, permitiendo corregir el rumbo permanentemente durante el desarrollo de software.

- **Todo el equipo realiza pruebas**: los analistas de negocio y desarrolladores de software también ejecutan pruebas, no sólo los testers como en métodos convencionales.

- **Reducir el tiempo para recibir retroalimentación**: los equipos del área de negocio (el cliente) están involucrados en cada iteración, no solo al final durante la fase de aceptación, como resultado, el tiempo de retroalimentación se reduce y el costo de correcciones también es menor.

- **Código limpio**: Los defectos en el código se corrigen en la misma iteración, por lo que se mantiene el código limpio.

- **Reducir la documentación de pruebas**: los testers usan listas de chequeo reusables en lugar de documentación extensa, se enfocan en la esencia de la prueba en lugar de detalles. Siguiendo principios ágiles estas listas de chequeo son el inicio de las definiciones de las pruebas y no el final, y el tester cuenta con libertad para aportar valor.

- **Guiado por pruebas**: las pruebas se hacen "durante" el desarrollo y no después del desarrollo como en métodos convencionales.

Algunas de las prácticas relacionadas con el testing ágil son:

- **Desarrollo Dirigido por Pruebas (TDD):** es una metodología que involucra dos prácticas: escribir los casos de prueba antes de comenzar con el código y refactoring. Apunta a desarrollar software de forma incremental de manera simple y robusta. Refactorizar el código busca lograr su optimización.
- **Desarrollo Dirigido por Pruebas de Aceptación (ATDD):** todo el equipo discute en colaboración criterios de aceptación, con ejemplos, y luego los divide en un conjunto de pruebas de aceptación en concreto antes de que comience el desarrollo. ATDD es más cercano a un proceso más que una actividad.

- **Desarrollo Guiado por el Comportamiento (BDD):** se crea una especificación/requerimiento ejecutable que falla porque la característica no existe, entonces prosigo con escribir el código más simple que puede hacer pasar la especificación, a continuación, se realiza la refactorización para eliminar la duplicación y el ciclo se repite hasta que la especificación esta lista. El analista en conjunto con los testers diseñan los tests y los Desarrolladores solo generan el código necesario para ejecutarlos.

- **Pruebas de Exploración**: enfoque en el cual el aprendizaje de la funcionalidad, diseño de pruebas y ejecución de pruebas ocurren simultáneamente, en contraposición con el enfoque convencional en el cual primero se documenta la funcionalidad o requisito, luego se diseña el caso de prueba y luego se ejecuta de acuerdo a guiones prestablecidos. Las pruebas exploratorias no están predefinidas ni se ejecutan según un plan.

- **Automatización de pruebas de regresión:** tanto la integración continua como la refactorización son prácticas necesarias para poder implementar una metodología ágil de desarrollo de software. Ambas técnicas implican modificar las fuentes de código constantemente, por lo que la automatización de pruebas de regresión por medio de herramientas es una necesidad imperiosa.

- **Automatización de pruebas unitarias:** consiste en usar un marco de trabajo o framework para ejecutar tus tests unitarios, en lugar de ejecutar estos manualmente una y otra vez cada vez que se modifique el código. Para ello existen múltiples frameworks, muchos de los cuales pueden integrarse en los ambientes IDE.

2.10.3 La pirámide del Testing

Originalmente propuesta por Mike Cohn, la pirámide del testing sirve para explicar las diferencias del testing trabajando con metodologías convencionales y trabajando con metodologías ágiles o de forma iterativa.

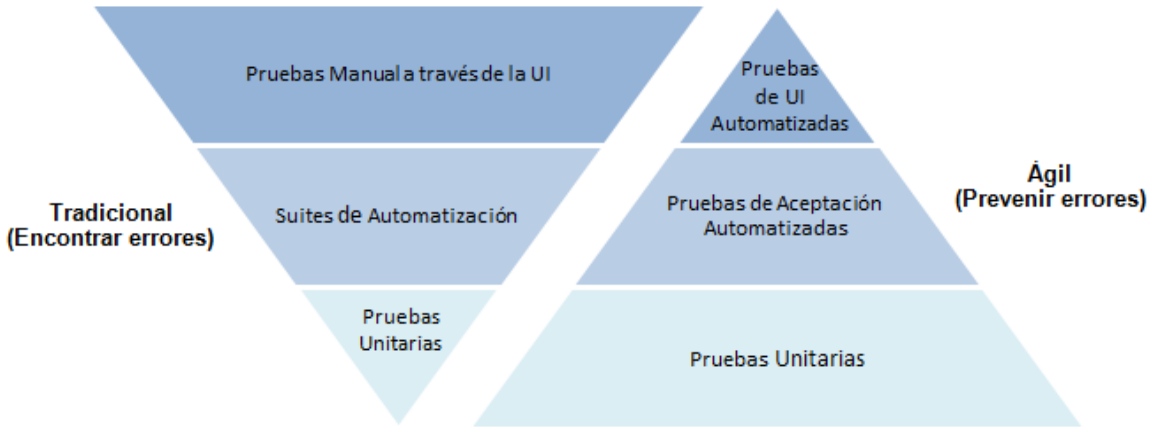

Figura 56: Pirámide del Testing (Fuente: http://www.kleer.la)

En la forma tradicional (pirámide de la izquierda), la gran mayoría de las pruebas son manuales y funcionales, pudiendo existir algún pequeño grado de automatización y de pruebas unitarias por desarrolladores.

La pirámide de la derecha representa como se ejecuta el Software Testing en Agile, donde la gran mayoría de las pruebas son automatizadas tanto unitarias, de aceptación como las de interfaz gráfica, buscando reducir al mínimo las pruebas funcionales manuales.

2.10.4 Los cuadrantes del Testing Ágil

Una clasificación útil para poner el testing ágil en contexto y guiar a los equipos y gerencia sobre cómo integrarlo en sus prácticas de desarrollo iterativos, son los cuadrantes originados por Marick y luego adaptados por Crispin y Gregory en su obra “Agile Testing”.

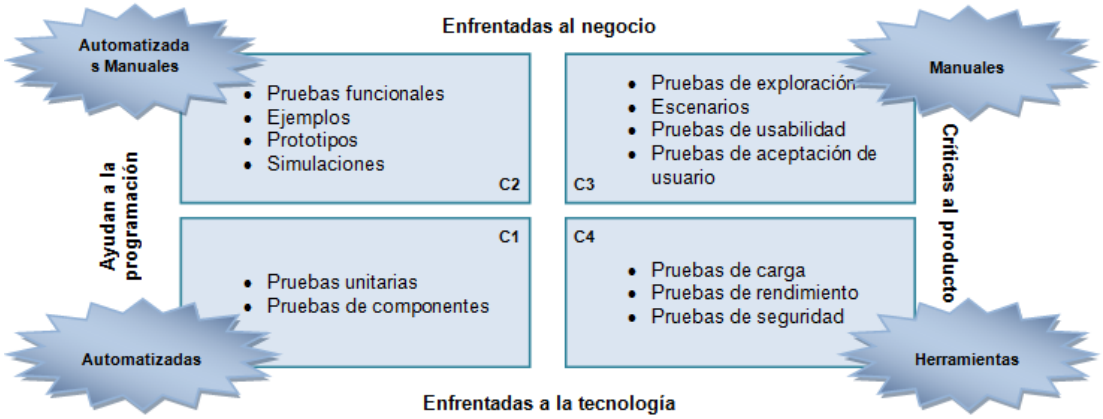

Figura 57: Cuadrantes del Testing Ágil (Fuente: http://www.kleer.la)

Las pruebas de los cuadrantes del lado izquierdo ayudan al equipo a definir qué código necesitan escribir y cuándo han terminado de escribirlo.

Los cuadrantes del lado derecho ayudan al equipo a aprender más acerca de las funcionalidades y códigos que han escrito, lo cual se traduce usualmente en nuevas historias de usuario o modificaciones a las existentes. Permiten identificar aspectos que sirven para mejorar el producto.

Los cuadrantes del lado superior abarcan pruebas de cara al negocio, que suelen ser de naturaleza más funcional, mientras que las pruebas de los cuadrantes inferiores son de mayor naturaleza técnica y no funcional.

Las pruebas del cuadrante C1, están orientadas a dar ayuda al equipo. Generalmente se encuentran automatizadas, lo que garantiza que el código se mantendrá estable ante cambios en los componentes.

Las pruebas del C2, tienden a dar soporte al equipo bajo lenguaje de negocio. Sólo algunas pueden ser automatizadas, la mayoría deben ser ejecutadas manualmente.

Las pruebas del C3, se crean en lenguaje de negocio y están orientadas a ayudar a mejorar el producto.

Por último, el C4, son pruebas que se realizan a través de herramientas de soporte (Ej. Prueba de seguridad)

2.10.5 El rol del Tester en un marco Ágil

El rol del tester en un equipo ágil es el de un experto, garante de entregar el valor de negocio deseado por el cliente a un ritmo sostenible y continuo. Se supone que posee una mayor especialización técnica, considerando que debe manejar herramientas de automatización, gestión ágil y metodologías. Su rol tiene mayor interacción con otras personas como clientes o desarrolladores, por lo que requiere de habilidades para la comunicación, orientación al cliente, poder de negociación, entre otras. Esto se contrapone con el rol convencional del tester, en el cual es un profesional encargado de elaborar diseños de prueba (a partir de diseños funcionales), y luego ser un simple ejecutor de escenarios preestablecidos.

El foco del tester está en la aplicación de enfoques tipo Desarrollo Guiado por el Comportamiento (BDD), usualmente trabajando en paralelo con el equipo de desarrollo y no en la fase final. En Ágil, todo el equipo es responsable de la calidad del producto.

2.11 Historias de Usuarios

Una historia de usuario (o User Storys) describe una funcionalidad que, por sí misma, aporta valor al usuario o al negocio. No se amplía su detalle hasta el momento en que la historia de usuario se vaya a desarrollar. Pueden ser creadas

durante la conversación con los usuarios interesados (stakeholders) incluyen nuevas funcionalidades o mejoras del proyecto.
La creación de la Historia de Usuario completa consta de:

Tarjeta: una descripción escrita en lenguaje de negocio que sirve como identificación y recordatorio del requerimiento y ayuda para la planificación mediante la priorización.

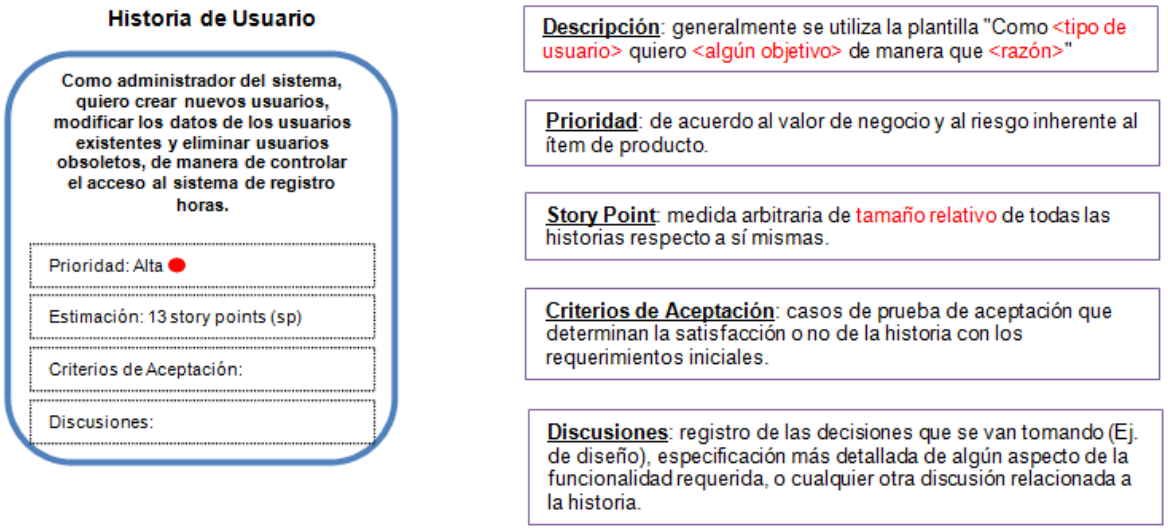

Figura 58: Composición de una Historia de Usuario (Fuente: Equipo de trabajo)

Conversación: es el diálogo que ocurre entre los miembros del equipo y el dueño del producto para aclarar los detalles y dudas sobre esta Historia de Usuario. Es la parte más importante de la historia.

Confirmación: define las pruebas que se llevarán a cabo para poder decir que la Historia de Usuario se ha completado con éxito.

Una buena historia de usuario sigue el modelo de INVEST:

- **Independiente**: una Historia de Usuario debe ser independiente de otras. Facilitan la planificación, la priorización y la estimación.
- **Negociable**: la tarjeta de la historia es tan sólo una descripción corta que no incluye detalles. Los detalles se añaden mediante la conversación.
- **Valiosa**: cada Historia de Usuario debe tener valor para el cliente.
- **Estimable**: para que la historia pueda ser estimada debe ser concreta y acotada.

- **Pequeña**: una buena historia debe ser pequeña en esfuerzo, debería ser realizable en menos de una semana.
- **Testeable**: una historia necesita debe poder probarse y saber que la Historia de Usuario se ha completado con éxito.

Criterios de Aceptación: Dentro de los criterios de aceptación deben definir que pruebas se llevarán a cabo para poder decir que la Historia de Usuario se ha completado con éxito. Si es necesario, se podrán adjuntar los mensajes de error o de éxito.

Como buena práctica, se recomienda el uso de **prototipos** siempre que se pueda como explicación e ilustración de la Historia de Usuario.

Según el nivel de detalle, podemos organizar nuestro proyecto en:

- **Temas**: grandes proyectos, peticiones globales sin más análisis ni detalle.
- **EPICS**: "Super" historias de usuarios. Un poco más concretas que los temas.
- **Historias de Usuarios**: una manera simple de describir una tarea concisa que aporta valor.
- **Tareas**: las Historias de Usuario pueden ser divididas en diversas tareas por necesidades técnicas.

Malas prácticas:

- Escribir historias que dicen cómo se hará en vez de qué se debe hacer.
- Una Historia de Usuario no es un Caso de Uso porque no se centra en el cómo ni tampoco es una definición exhaustiva de los requisitos.
- No escribir el criterio de aceptación o no ser suficientemente explícito.
- No estimar una tarjeta puede crear falsas expectativas y dificulta la autodisciplina.
- Confiar en todo lo escrito en la tarjeta: a veces es necesaria documentación externa.
- Dar una historia por completada cuando está “prácticamente completa” en vez de “completa-completa”.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Capítulo 3: Detalle del Producto y del Proceso

El producto WGobv7.0 es un sistema software integrado de aplicaciones WEB que tiene como objetivo gestionar la mayoría de las áreas administrativas de una municipalidad o comuna. Es flexible, escalable y admite diferentes configuraciones de acuerdo a las necesidades y a las diferentes legislaciones locales, provinciales y nacionales.

3.1 Características del Producto (Fuente: GobFactory)

- **Tecnología**
  Web Enabled, multiusuario y sin límite de licencias.

- **Contribuyentes y delegaciones municipales en línea**
  El Sistema utiliza el medio de comunicación más económico: Internet. Posibilitando la consulta instantánea con entes regionales, delegaciones municipales, atención de los vecinos "en línea" desde su casa, procuradores municipales desde sus estudios, etc.

- **Optimización permanente**
  Lo garantiza el permanente proceso de optimización, gracias al monitoreo de rendimiento que le dan los miles de operadores, dispersos en numerosos municipios con diversas problemáticas.

- **Seguro**
  La concepción modular de todo el sistema permite asignar a cada usuario los procedimientos que le corresponden por sus responsabilidades y permisos, al nivel de grupos, usuarios, funciones y acciones.

- **Cajero municipal y compatible con múltiples entidades de recaudación**

  Trabaja con lector de código de barras e impresora tickeadora, utilizando Ticket Recibo con cinta testigo, optimizando la seguridad, además ahorra trabajo ya que hace el control de recaudación en línea ante el contribuyente, al igual que la discriminación contable de los ingresos.

- **Capacidad de realizar auditorías**
  El sistema puede realizar controles lógicos sobre el curso que han seguido los datos. Cada transacción realizada, que genera modificación sobre los mismos, queda registrada en archivos de auditoria, junto con la identidad del usuario que la hubiere generado, fecha, hora y los datos pertinentes.

- **Modular**
  Sistema integral e integrado que se puede implementar paulatinamente por cada área.

- **Parametrizable**
  Se ajusta a las reglamentaciones de cada municipio.

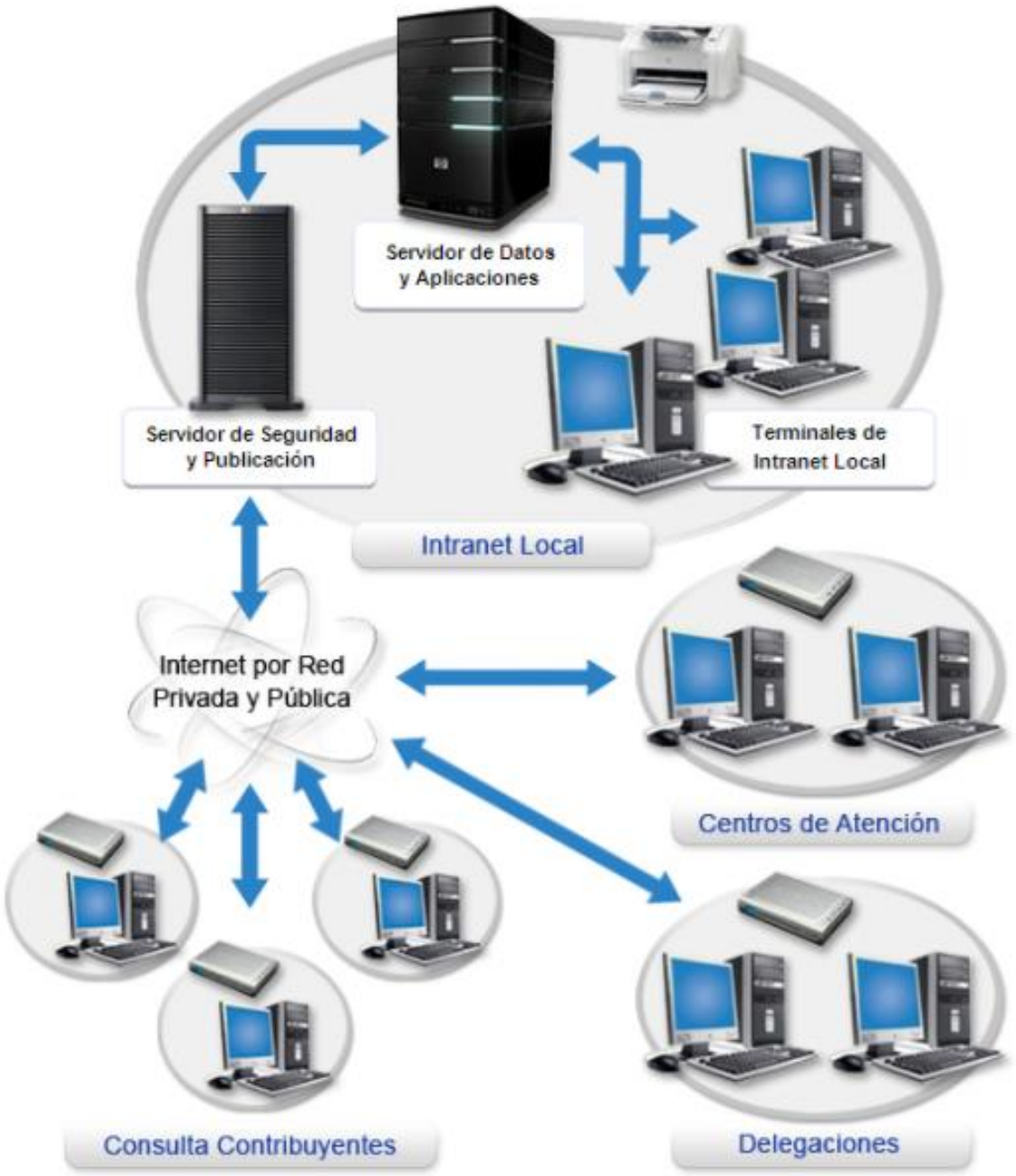

Figura 59: Esquema de comunicación de GobFactory (Fuente: GobFactory)

3.2 Módulos del Producto

El Software comprende las Áreas de:

- Padrón de Personas, Ficha Centralizada y unificación.
- Rentas: abarca todas las tasas, servicios, impuestos, derechos, sellados y contribuciones por mejoras (Ej. Automotores, Inmuebles, Comercio e Industria, Cementerio, Obras, Tarifarias, Mediciones de Servicios, etc.)
- Gestión de Deudas: incluye liquidación de deuda, planes de pagos, convenios de pago, gestión de contribuyentes al día, etc.
- Procuración Extrajudicial y Judicial: gestión de procuradores, bloqueos de deuda, generación de certificados de deuda y carpetas de demanda.
- Contaduría: incluye generación y aprobados de planes de cuenta (presupuestos), movimientos de la caja, transferencias, cuentas bancarias, conciliaciones, libros bancos, ejecución presupuestaria (cierre de balances), cierres de planillas de caja, operaciones de egreso (compromisos, orden de pago, pagos, contradocumentos), operaciones de ingreso (recibo de ingreso, nota de contabilidad de ingresos, contra asientos) e informes varios.
- Recaudación: ABM de cajas municipales, gestión de entes de recaudación, seguimiento del estado de cajas, cierres de cajeros, recibos digitales y reportes varios.
- Compras: ABM de proveedores, órdenes de compra, pedidos de abastecimientos, pedidos de suministro, órdenes de entrega, facturación y reportes.
- Inventario de bienes.
- Tribunal de Cuentas
- Juzgado de Faltas: ABM normativas, ABM de actas, gestión de inspectores, generación de causas, sentencias, demandas, notificaciones e informes varios.
- Liquidación de Sueldos y Jornales: ABM de legajos, ABM de novedades, liquidaciones, gestión de entes e informes varios.
- Catastro
- Mesa de Entradas: generación y seguimientos de expedientes.
- Obras Privadas: ABM de obras, seguimientos de expedientes de obras, certificados (Ej. final de obras) e informes.
- Subsidios: ABM de subsidios y gestión de beneficiarios.
- RNPA (Registro Nacional de la Propiedad Automotor): sincronización municipio-RNPA y RNPA-municipio, movimientos diarios e reportes.
- Licencia de Conducir: ABM de carnet, carga de jurisdicciones, carga de parámetros de control, captura de foto, firma digital e impresión del carnet.

No Contempla:

No están incluidos Sistemas de Información Geográficos, Atención a la Salud (Dispensarios y Hospitales), Registro Civil, Stock de Materiales, Mantenimiento de Equipos del Corralón Municipal y Gestión de Personal.

Perspectiva del Producto:

WGob es un software totalmente independiente y autocontenido que integra diferentes módulos de modo interrelacionado. En un solo producto, se abarcan todas las necesidades operativas y funcionalidades que requieren las diferentes áreas administrativas del municipio.

3.3 Requerimientos del Producto

En GobFactory los requerimientos se encuentran representados por solicitudes en la Herramienta “Gestión de Solicitudes”. Las mismas pueden ser receptadas en las distintas visitas del Consultor al Municipio o vía telefónica ante llamados al Área de Atención al Cliente. Luego son ingresadas a la herramienta por el responsable de atención. Existen también solicitudes que surgen internamente como sugerencias de mejora o iniciativas de nuevas prestaciones.

Según el origen de la solicitud (Cliente Interno, Cliente Externo o Soporte) se asigna el número de pedido:

- Aquellas que fueron relevadas en visitas o telefónicamente tienen asignada una numeración que va de 00001 - 99999 y tienen el respaldo de la solicitud impresa.
- Las solicitudes internas que implican desarrollo de software se cargan directamente en la herramienta pedidos (no tienen asociada una solicitud). Utilizan también el número de 00001 - 99999.
- Las solicitudes de soporte técnico telefónico (no involucra desarrollo) se representan con numeración del 100001 - 999999 y se cargan directamente en la herramienta (no tienen asociada una solicitud impresa).

Una solicitud puede tener asociado uno o más pedidos de acuerdo a la complejidad o temas a los que se refieren.

Desde el punto de vista del proceso, un pedido representa un ítem del Backlog del Producto. Una historia puede referenciar a uno o más pedidos.

Herramienta “Gestión de Solicitudes” - **Ejemplo Listado de Pedidos**

Rango en la Pila	Fecha Límite Entrega	Origen	Nro. pedido	Nro. solicitud	Fecha de pedido	Test Unitario	Fecha de Cierre	Código Producto	Tema/Subtema	Solicitud	Estado	Proyecto - Iteración	Tipo Pedido	Clasificación	Programadores	Solicitado Por	Fecha Suspendido	Clientes Externos	Prioridad	Complejidad	Proyecto Origen	Iteración Origen	Producto
7600	21/03/2016	Soporte	12793	100740	18/03/2016	No		9.000.001.001.000	Transferencias	Transferencias: Se solicita la transferencia de Rentas de prueba de Vx a V6.	En Ejecucion Proyecto	Muni - Iteración 32	Nuevo		Juan	Carlos		-		2-Mediano			WebGob v7
7580	23/03/2016	Soporte	12791	100738	17/03/2016	No		9.000.001.001.000	Transferencias	Transferencias: Se solicita la transferencia de Rentas de la Vx a V6 para Tortugas.	En Ejecucion Proyecto	Muni - Iteración 32	Modificacion		Luis	Raúl		-		1-Simple			WebGob v7
7598	23/03/2016	Soporte	12798	100743	21/03/2016	No		9.000.001.008.000	Verificación de Situaciones Particulares	Verificación de Situaciones Particulares: Se solicita verificar porque una cuota saldo salen mal los montos impresos. La boleta de contado tiene un descuento de 151.70 y no es considerado al momento de imprimir la boleta.	Asignado para Correccion	Muni - Iteración 32	Error		Juan	Felipe		-	Urgente	1-Simple			WebGob v7

Figura 60: Ejemplo Listado de Pedidos (Fuente: GobFactory – Herramienta Gestión de Solicitudes)

Rango de pila: es un valor asignado por el comité de producto a los requerimientos que se encuentran seleccionados para ingresar al sprint. Para determinarlo se tienen en cuenta el cliente, la urgencia y la prioridad según el PO. Es una clasificación por encima de la prioridad de la solicitud.

Código del producto: es el código comercial, representa el módulo, submódulo, tema y subtema del sistema impactado por el requerimiento. Este número es consistente con la estructura interna del sistema.

Estado: según la instancia en la que se encuentre el pedido, estos pueden ser:

- **Pendiente** (estado inicial), no tiene asignado una iteración ni desarrollador, generalmente se encuentra en el backlog.
- **Asignado para corrección:** se utiliza para pedidos de tipo error e indica que ya se encuentra tomado por un equipo de desarrollo.
- **Suspendido:** son pedidos para los cuales se tomó la decisión de detener su producción en la iteración actual. Pero se retomarán en futuros sprints.
- **Cancelado:** son pedidos para los cuales se tomó la decisión de desestimar su producción.
- **En ejecución:** aplica cuando el pedido es de tipo modificación o nuevo e indica que ya se encuentra por un equipo de desarrollo.

3.4 Requerimientos No Funcionales del Producto:

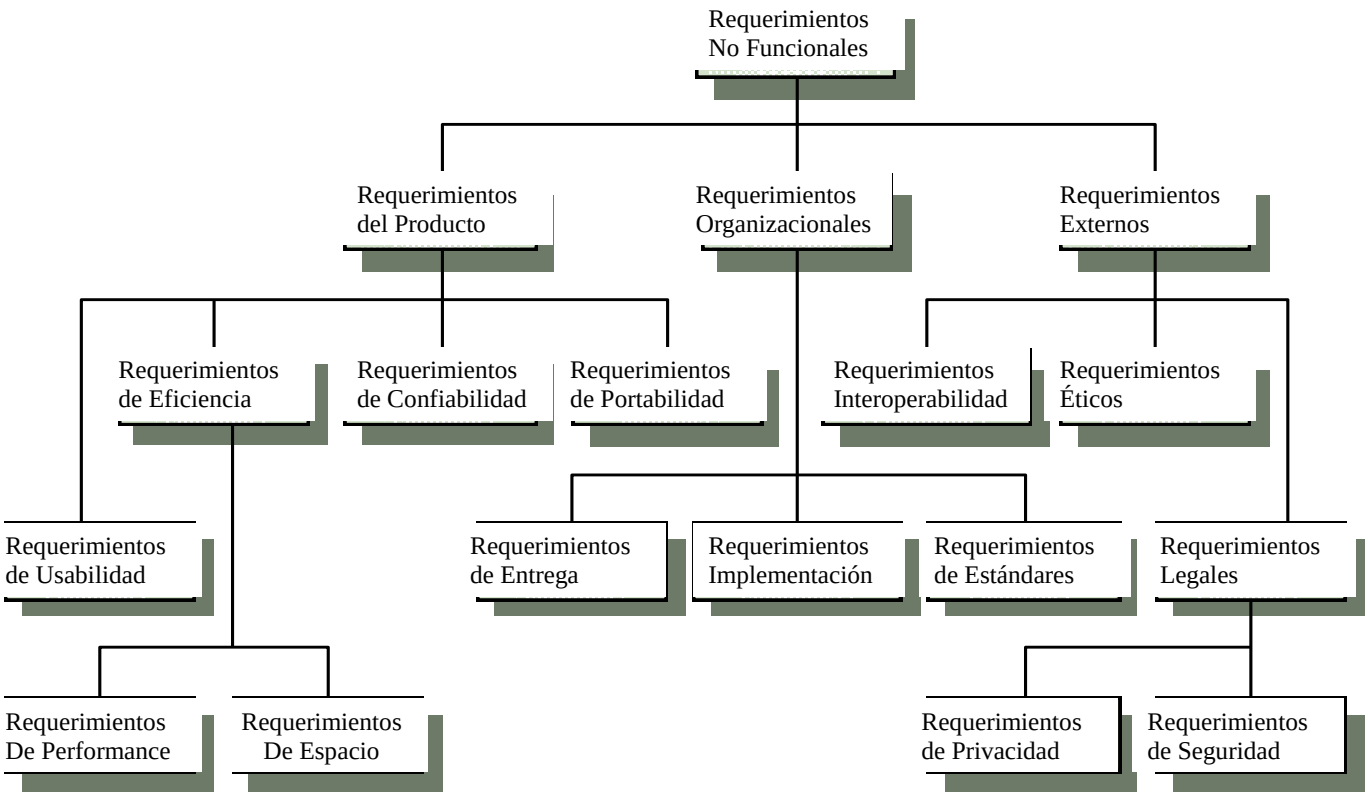

Figura 61: Requerimientos No Funcionales (Fuente: GobFactory – Documento ERS: Especificación de Requerimientos de Software)

Los requerimientos NO funcionales definidos por la empresa y a tener en cuenta son:

Del Producto

Usabilidad	Algunas consideraciones para medir la usabilidad de un producto de software son: Especificar el tiempo de capacitación requerido para usuarios normales y expertos para convertirse en productivos en operaciones particulares. Especificar tiempos de tareas mensurables para tareas típicas, alternativamente, Requerimientos de usabilidad básica del nuevo sistema sobre otros sistemas que los usuarios conocen y les agradan. Especificar requerimientos para conformidad con los estándares comunes de usabilidad, tales como estándares de GUI.
Confiabilidad	La confiabilidad podría expresarse en término de alguno de estos aspectos: Disponibilidad: Especificar el porcentaje de disponibilidad de tiempo, horas de uso, acceso de mantenimiento, etc.

	Tiempo Mínimo entre fallas: Especificado usualmente en horas, pero también puede especificarse en días, meses y años. Tiempo Mínimo de Reparación: ¿Cuánto tiempo está permitido que el sistema esté fuera de operación después de una falla? Certeza: Precisión Específica (resolución) y certeza (sobre un estándar) que es requerida para las salidas del sistema. Errores (bugs) Máximos o ratios de defecto: usualmente expresados en términos de BUGS/KLOC (miles de líneas de código) o bugs por casos de uso Errores (Bugs) o ratios de defectos: Categorizados en términos de bugs menores, significativos y críticos. Los requerimientos deberán definir lo que quiere decir bug "crítico" (tal como datos completamente perdidos, inhabilitación completa para usar ciertas partes de la funcionalidad del sistema).
Performance	Incluye tiempos de respuesta específicos. Tiempo de respuesta para una transacción (promedio, máximo), de principio al fin Capacidad (el número de clientes o transacciones que el sistema puede soportar). Modos de Degradación (modo aceptable de operación cuando el sistema ha sido degradado). Utilización de Recursos (memoria, disco, comunicaciones)
Portabilidad	Debe expresar las necesidades de crecimiento del producto hacia otras tecnologías de desarrollo, sistemas operativos y/o plataformas de hardware.

De la Organización

Entrega	Si la organización tiene requisitos explícitos respecto a la entrega del producto, entre los cuales podemos mencionar, fechas, épocas del año, días u horas específicos para por hacer el despliegue del producto, instalaciones on site distribuidas o remotas, etc., deberán especificarse en este apartado.
Implementación	Este apartado deberá especificar cualquier consideración que impacte en la construcción del producto que sea un requisito planteado por el cliente y el producto debe respetar.
Estándares	Si la organización contratante (cliente) desea que el producto respete ciertos estándares asociados al producto en sí mismo o a su proceso de desarrollo, los mismos deberán especificarse en este apartado.

Del Exterior

Éticos	Si existen requerimientos que deben considerarse en el contexto del producto que si bien no están legislados, responde a factores morales o pautas de conducta, deberán especificarse aquí.

Legales	Identificar si existen legislaciones nacionales, internacionales, provinciales, etc., aplicables y vigentes, que el software deba considerar. También incluya aspectos relacionados a los derechos de copia (copyright).
Interoperabilidad	Este aspecto implica requisitos vinculados con la necesidad de que el producto de software se comunique con otros productos de software del exterior, para intercambiar datos o algún otro aspecto.

3.5 Proceso de Desarrollo de Software Actual de WGob v7.0

Como base para el proyecto actual y con vistas de lograr a largo tiempo una mejora continua, GobFactory implementó el siguiente proceso de desarrollo.

Gráficamente las etapas del proyecto consideradas en el proceso de este producto son las siguientes:

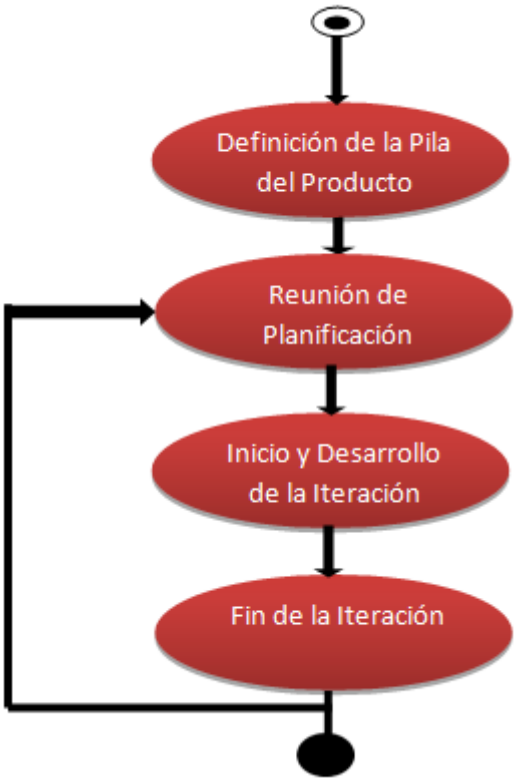

Figura 62: Etapas del Proceso de Desarrollo de Software Actual (Fuente: GobFactory – documento "Descripción del Proceso Implementado")

3.6 Diagrama del Proceso Actual del Producto WGob v7.0

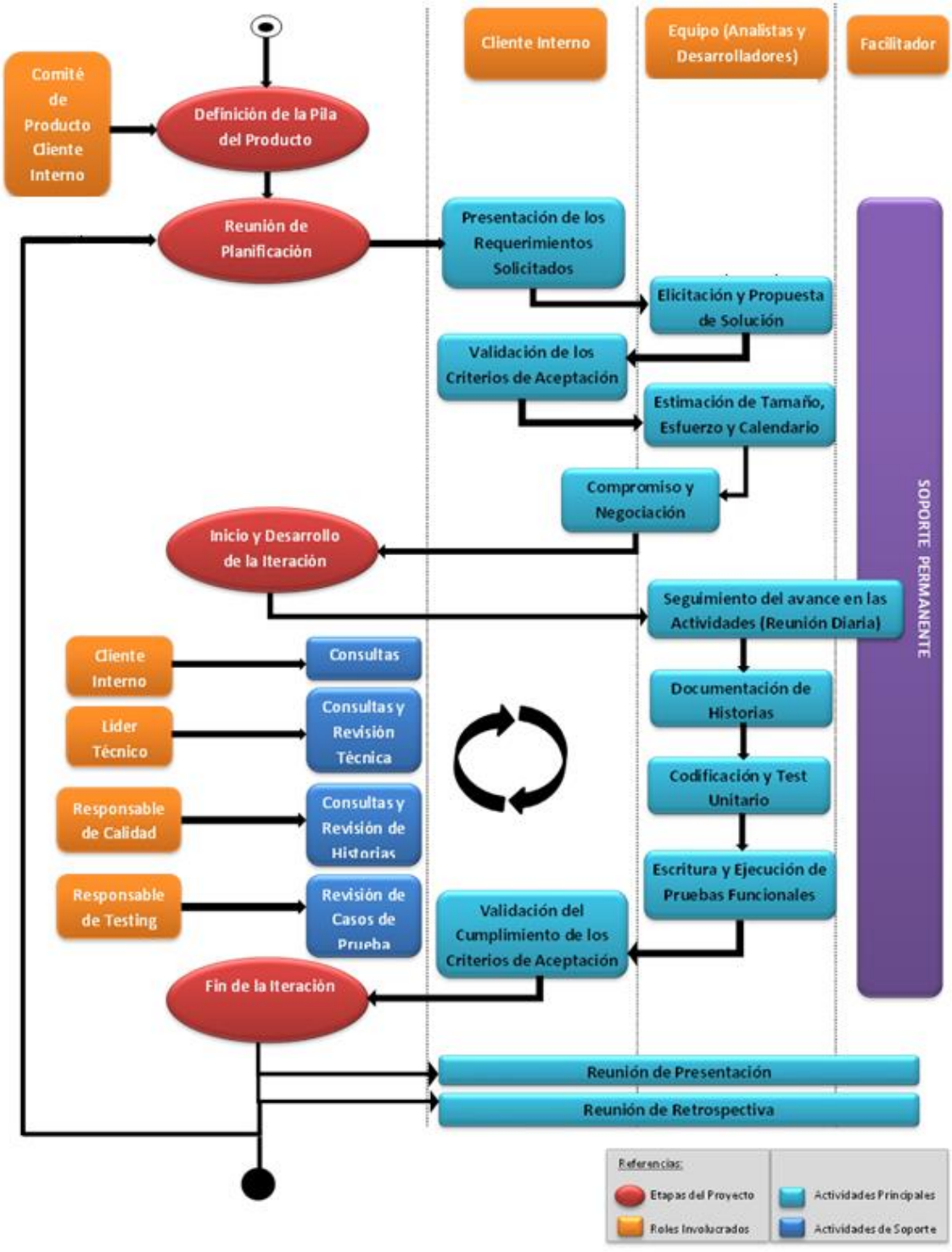

Figura 63: Proceso de Desarrollo de Software Actual (Fuente: GobFactory – documento "Descripción del Proceso Implementado")

A continuación se explica el Diagrama del Proceso Actual del Producto WGob v7.0. En él se observa cómo se lleva adelante la actividad y que atraviesa cuatro etapas claves (Fuente: GobFactory):

3.6.1 Etapa A: Definición de la pila del producto:

Objetivo

Definición de la pila de requerimientos ordenados y priorizados, a los que se les desea dar tratamiento a lo largo del proyecto/iteración. Como así también la identificación de los criterios de aceptación para cada uno de ellos.

Aclaración: Esta pila puede ser definida no solamente para una iteración sino para todo el proyecto, la misma es dinámica y puede variar con el tiempo.

Roles involucrados

- Comité de Producto
- Cliente Interno
- Adicionalmente se pueden hacer consultas puntuales al: Líder Técnico, Desarrollador, Analista, entre otros.

Actividades

Una vez identificados los requerimientos que formarán parte de la pila del producto serán **priorizados** basándose en la siguiente categorización:

1- Extremadamente Alta
2- Alta
3- Media
4- Baja

Cada uno de estos requerimientos deberá estar ordenado por el valor que tienen para el cliente externo/interno y las restricciones en cuanto a la funcionalidad, esto es lo que se conoce como **Rango en la Pila**.

El Rango en la Pila está representado por un valor numérico, entero positivo. Inicialmente se deben asignar valores que aumenten de 10 en 10, para brindar la posibilidad de usar valores intermedios. Tiene más peso que la prioridad determinada por el cliente.

Los requerimientos más importantes estarán al comienzo de la Pila.

A continuación se presentan dos tablas con los conceptos mencionados con el objeto de ejemplificar el uso de prioridad y el rango en la pila.

En la primera tabla (A) se puede observar que para distintos requerimientos con la misma prioridad pueden representar distintos valores para el negocio, que corresponde al Rango en la Pila tales como el Req.1 y Req.2. Como así también para prioridades altas un menor Rango de Pila ejemplo: Req.6.

Al ordenar la Pila del Producto surge la 2da tabla (B) donde se pueden observar la escala de precedencia entre requerimientos de acuerdo al Rango en la Pila. Esto es necesario al momento de definir cuáles requerimientos pueden postergarse para futuras iteraciones y cuáles no.
Como se indicó la Pila del Producto es dinámica por lo que varía con el tiempo no solamente con la introducción de nuevos requerimientos sino también la revaloración del Rango de los requerimientos incluidos con anterioridad.

A

RANGO EN LA PILA	PRIORIDAD	REQUERIMIENTO	TEMA
100	1	Req.1	Tema 1
90	1	Req.2	
80	2	Req.3	
50	4	Req.4	Tema 2
70	2	Req.5	
60	1	Req.6	
...	...	...	...

B

RANGO EN LA PILA	PRIORIDAD	REQUERIMIENTO	TEMA
100	1	Req.1	Tema 1
90	1	Req.2	
80	2	Req.3	
70	2	Req.5	Tema 2
60	1	Req.6	
50	4	Req.4	
...	...	...	...

Figura 64: Ejemplo del cálculo de la pila de producto de WGob (Fuente: GobFactory – documento "Descripción del Proceso Implementado")

Salidas del Backlog – Entregables resultantes

- Listado de los requerimientos priorizados y ordenados. Los mismos deben estar registrados en la Herramienta Gestión de Solicitudes para luego asociar las historias que surjan, a los mismos.
- Identificación de las condiciones mínimas de aceptación para cada uno de estos o por funcionalidad.
- Estimación de esfuerzo y calendario (cantidad de días) sugerida, solamente para los requerimientos a tratar en la iteración que esta por dar comienzo.

3.6.2 Etapa B: Reunión de planificación

Objetivo

Trasladar los requerimientos al Equipo de Trabajo y obtener el compromiso del mismo con respecto a las funcionalidades que podrán entregar.

Roles involucrados

- Cliente Interno (Dueño del Producto y representantes)
- Equipo de Trabajo (Desarrolladores y Analistas)
- Facilitador
- Adicionalmente se pueden hacer consultas puntuales al: Líder Técnico, Gestor de Configuración y Responsable de Calidad.

Actividades

Las actividades que se van a llevar a cabo en la misma son:

- **Presentación de los Requerimientos Solicitados:**

  Presentación de los requerimientos por parte del Cliente Interno con sus criterios de aceptación al Equipo de Trabajo. También deben incluirse los Nº de pedidos de requerimientos asociados.

- **Elicitación (reunión para ampliación de requerimientos) y Propuesta de Solución:**

  Esta actividad se centra en que el Equipo de Trabajo tenga un conocimiento de lo que se tiene que hacer, para esto pueden crear prototipos de interfaz, diagramas de estados, identificación de los requerimientos no funcionales, reglas de negocio, tablas, entre otros.

  A partir de la información recabada se identifican las Historias asociadas. Por lo menos debe contarse con la Descripción General escrita. La base para la misma es el estándar:

  > *Como <rol de usuario> solicito que <detalle del requerimiento> para <uso que se le va a dar a lo requerido dentro del ámbito operativo>.*

  Este proceso tiene como objetivo la identificación de que es lo hay que hacer y una aproximación inicial de la solución a implementar. Como así también un entendimiento uniforme para todos los miembros del equipo de trabajo.

- **Validación de los Criterios de Aceptación:**

  En la misma el Cliente Interno deberá validar si los criterios de aceptación presentados en la actividad Nº1 han sufrido o no modificaciones. Como así también si se identificaron nuevos.

- **Estimación:**

  En esta etapa el Equipo ya tiene el conocimiento suficiente para poder estimar el tamaño de las historias identificadas, la técnica de estimación a implementar es Póker Planning.

  El tamaño se mide en Puntos de Historias.

  A partir de esta estimación el equipo de trabajo debe identificar todas las tareas y estimar el esfuerzo de cada una (medido en horas/hombre) necesario para completar las historias comprometidas.

  Luego, el mismo deberá calendarizar las tareas para establecer la cantidad de días que les llevará realizar el total de funcionalidades presentadas.

  Todas las tareas identificadas formaran parte de las actividades planificadas para realizar dentro de la iteración. Por esto es necesario que las mismas cuenten con un esfuerzo reducido, se sugiere que no supere dos días ideales de trabajo. El **esfuerzo ideal diario** planteado es de 6 horas/hombre por día.

  Esto se plantea porque a lo largo de la iteración cada miembro del equipo deberá registrar el progreso en cada tarea para poder medir el avance de la iteración en su totalidad y controlar si se presentan o no desvíos de acuerdo a lo planificado.

  Por otra parte durante esta etapa se presentan adicionales al Facilitador otros roles de soporte que deben estar disponibles:

  - Líder Técnico: para resolver dudas técnicas.
  - Cliente Interno: para responder consultas de los requerimientos solicitados.

- **Compromiso y Negociación:**

  En base a la estimación anterior el Equipo de Trabajo puede identificar la cantidad de días que les llevará cumplimentar con lo solicitado y en caso de que se les presente una fecha límite de la iteración, es decir la cantidad de días que dura la misma, el Equipo deberá especificar si va a poder comprometerse en la realización de toda o parte de la funcionalidad solicitada.

  En esta actividad también surge la negociación debido a que el Comité de Producto junto con el Cliente Interno realizó previamente una estimación sugerida, que puede o no coincidir con lo planteado por el Equipo de Trabajo.

Salidas de la reunión de planificación – Entregables resultantes

Pila de la Iteración: representa a todas las historias que el Equipo de Trabajo se compromete a cerrar. Junto con las tareas identificadas para la completitud de las mismas.

Por cada Historia de la Pila se cuenta con:

- Descripción General de la misma junto con toda la información recabada que formaran parte de la solución. Adicionalmente se identifican las tablas, reglas de negocio, requerimientos no funcionales y prototipos de interfaz, entre otros.
- Identificación de las condiciones mínimas de aceptación e incorporación de nuevos criterios identificados.
- Validación de los criterios de aceptación por parte del Cliente Interno. Los mismos deberán estar documentados en una minuta de validación donde por cada Historia se documentará la Descripción General y los criterios de aceptación identificados.
- Identificación de las tareas a realizar para cada historia con su estimación en horas y prioridad.
- A nivel informal (solo porque no se plasma y entrega esa documentación) el equipo divide las tareas a realizar y las calendariza de modo tal de organizar su trabajo.

3.6.3 Etapa C: Inicio y desarrollo de la iteración

A lo largo de la Iteración se llevarán a cabo distintas actividades:

- Reuniones Diarias: reunión de seguimiento del progreso en las tareas.
  Cada miembro del equipo nombra las tareas realizadas, las que va a realizar durante el transcurso del día y las incidencias (problemas) que le pudieron surgir. Para poder cumplir con esta actividad, cada miembro del equipo, deberá registrar diariamente el avance en la herramienta TFS de cada una de las tareas planificadas y la creación de nuevas tareas en el caso que se presenten.

  Por otra parte el facilitador centralizará el tratamiento de las distintas incidencias y en caso de ser necesario re direccionará a los roles pertinentes para la solución de los mismos.

- Documentación de las Historias identificadas en la Reunión de Planificación plasmando en la herramienta TFS toda la información obtenida junto con las tareas que se deberán llevar a cabo. Además de la descripción especifica que surge durante la reunión de planificación, las historias incluyen una

descripción general, la cual se subdivide en las secciones solicitud, datos técnicos, criterios de aceptación y discusiones. Cabe resaltar que estas secciones son opcionales.

- Codificación y Test Unitario: Actividad realizada por los desarrolladores, la consideración es que deben ejecutar el test unitario teniendo en cuenta los criterios mínimos de aceptación.

- Escritura y Ejecución de Pruebas Funcionales: Actividad realizada por los Analistas de Pruebas, donde se deben registrar los Casos de Prueba en la Herramienta TFS. Como mínimo estos deben abarcar los criterios de aceptación planteados.

- Validación del Cumplimiento de los Criterios de Aceptación: Actividad en donde interviene el Cliente Interno y Analista de Prueba. El objetivo es que el Analista de prueba presente los Casos de Prueba donde se probaron los Criterios de Aceptación Especificados. Se deben plasmar la Validación de los Casos de Prueba por parte del Cliente Interno en la Sección Descripción de cada Caso de Prueba validado.

Las siguientes actividades se realizarán a medida que se necesiten:

- Consultas al Cliente Interno.
- Consultas y Revisión Técnica por parte del Líder Técnico.
- Consultas y Revisión de Historias: Revisión de las Historias escritas que cuenten con la estructura y completitud determinada. Revisión a cargo del Área de Calidad.
- Revisión de Casos de Prueba: se comprobará la cobertura, prioridad, coherencia y consistencia con los requerimientos solicitados que presentan los Casos de Prueba. Revisión a cargo del Responsable de Testing.

3.6.4 Etapa D: Fin de la iteración

Objetivo

Dar un cierre a la iteración y poder visualizar el incremento en la funcionalidad introducido, como así también evaluar las actividades resultantes en la misma.

Hay dos reuniones en esta etapa:

a. **Reunión de Presentación**- Demo del Producto: donde el Equipo presentan todas las historias cerradas que forman parte del incremento de la funcionalidad, al Dueño de Producto. Y se identifican todas las que no pudieron cerrarse.

   Una historia está cerrada si:

- Se encuentra documentada en la herramienta TFS y pasó la Revisión de Calidad.
- Se codifico.
- Realizaron en un 100 % las pruebas unitarias y funcionales diseñadas.
- Realizó la revisión de los Casos de Prueba.
- Realizó la validación del cumplimiento de los criterios de aceptación.

Roles involucrados

- Dueño del Producto y Cliente Interno
- Facilitador
- Equipo
- Y toda persona interesada en la demo.

b. **Reunión de Retrospectiva:** el objetivo de la reunión es la mejora continua. Es por esto que en el transcurso de la misma se deben identificar todas las actividades que se hicieron bien y las que se hicieron mal con el fin de encontrar oportunidades de mejoras.

Como datos de entrada se cuenta con las incidencias reportadas, las experiencias en la iteración identificadas por parte del equipo y facilitador. Como así también se pone bajo consideración el proceso aquí descripto y los documentos y entregables asociados con el fin de incorporar mejoras en estos.

Roles involucrados

- Dueño del Producto y Cliente Interno
- Facilitador
- Equipo
- Y toda persona interesada.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Capítulo 4: Fase Diagnóstica

Para conocer la realidad sobre la que hay que trabajar debimos abarcar tanto las manifestaciones de los problemas, como sus consecuencias y repercusiones. Se delimitó el alcance del proyecto, se tomaron datos, indicadores y estadísticas confiables (ya existentes en la empresa) y se delimitó el espacio y tiempo del proyecto.
Determinar el funcionamiento, las dificultades, las debilidades, las necesidades y las fortalezas, implicó analizar detalladamente el entorno de trabajo del equipo de desarrollo. Para ello, fue necesaria la participación de todos los miembros que lo integran, generando un espacio de debate y de intercambio de ideas que enriqueció la fase diagnóstica.

A continuación se reflejan algunos de los registros con los que ya cuenta la empresa y que permiten a simple vista ver el impacto en el proceso.

4.1 Horas estimadas vs horas completadas por sprint

En este gráfico podemos visualizar la diferencia importante entre las horas estimadas y las horas realmente ejecutadas.

Sprint	Horas estimadas al inicio	Horas completadas al final del Sprint
16	112	287
17	99	112
18	229	329
19	66	479
20	150	479
21	630	829
22	299	694
23	342	1844
24	206	1369
25	293	185
26	59	419
27	532	2414

Figura 65: Datos de Sprints (Fuente: Resultado estudio del grupo)

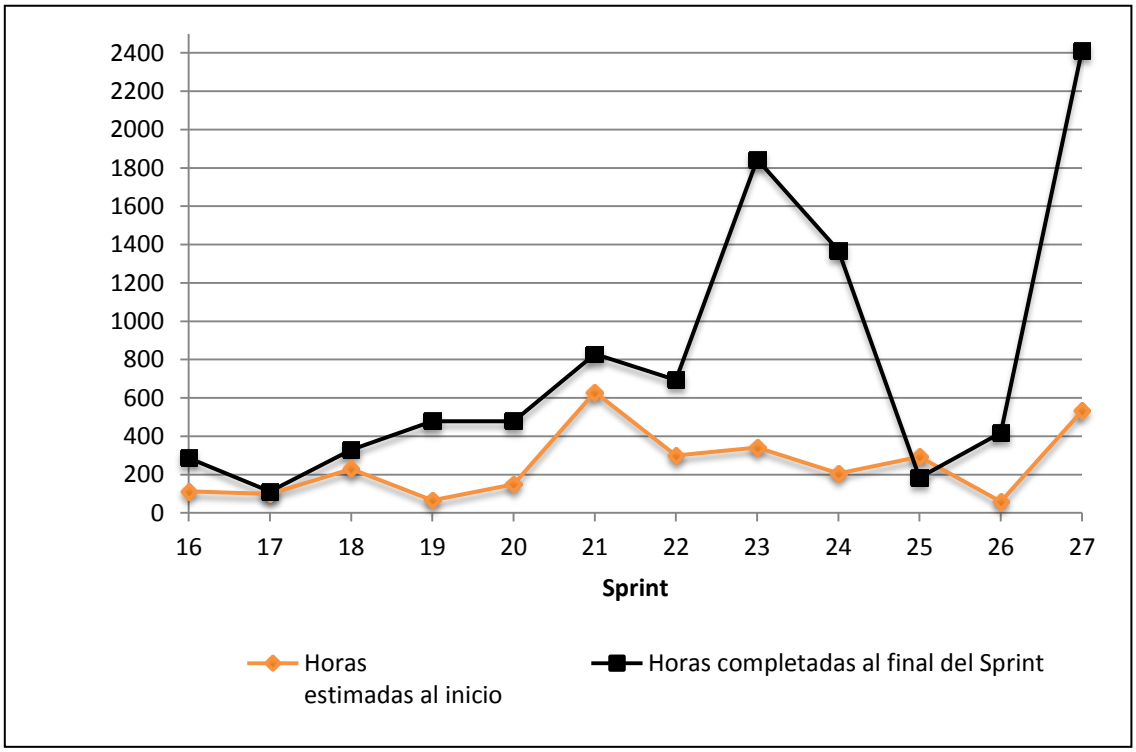

Figura 66: Gráfico de relación de horas estimadas vs horas completadas por sprint (Fuente: Resultado estudio del grupo)

Se observan iteraciones, como por ejemplo la 19, 23, 24, 26 y 27, con diferencias muy importantes respecto de lo planificado. En general se infiere una dificultad al momento de estimar las tareas a realizar durante el sprint.

4.2 Cantidad de solicitudes por tipo por sprint

A través de este gráfico se puede observar que la mayoría de las solicitudes que ingresan a un sprint corresponden a modificaciones al producto. En segundo lugar, se encuentran los pedidos por errores detectados en el cliente y que pasan a ser considerados prioritarios.

Cabe aclarar que en esta tabla sólo se consideran las solicitudes por error planificadas (ingresadas por el Comité). Las solicitudes por error no planificadas no están representadas en este gráfico dado que no se encuentran registradas en la herramienta "Solicitudes de Requerimientos". De todos modos se puede apreciar que los pedidos por error ocupan un porcentaje importante.

Los pedidos nuevos referidos a mejoras sobre el producto, actualizaciones, nuevos diseños, etc. ocupan el tercer lugar. Esto revela que se dedica poco tiempo a nuevas investigaciones y desarrollos que agreguen valor al producto.

	Sprint											
Tipo de Pedido	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
Modificaciones	15	7	10	0	64	49	58	39	66	11	6	247
Error	8	3	9	11	50	21	46	45	49	16	32	183
Nuevos	2	2	0	0	31	27	16	15	24	5	17	80
Total de Pedidos	25	12	19	11	145	97	120	99	139	32	55	510

Figura 67: Datos capturados por Tipo de Pedidos (Fuente: Resultado estudio del grupo)

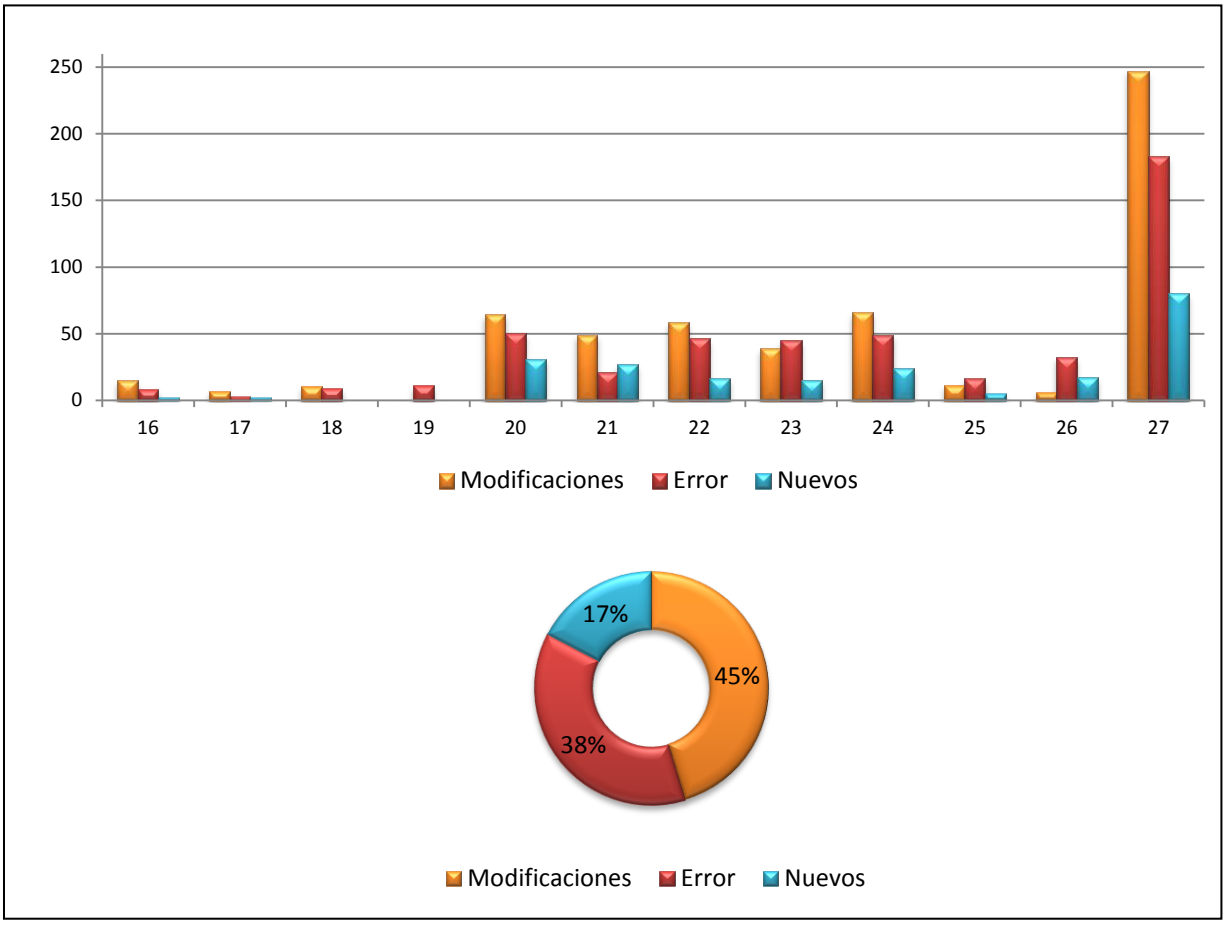

Figura 68: Gráfico de cantidad de solicitudes por tipo por sprint (Fuente: Resultado estudio del grupo)

4.3 Origen de las solicitudes de tipo error

En este gráfico de Pareto podemos ver que el factor "poco vital" y más importante generador de problemas es el área de Prueba. Correspondería prestarle atención y dedicarle recursos y esfuerzos para llevar a cabo acciones correctivas. Las otras áreas estarían dentro de los "muchos triviales".

Origen de pedidos de errores	Cantidad	Total	Porcentaje	Porcentaje acumulado
Pruebas	256	256	54%	54%
Análisis	75	331	16%	70%
Desarrollo	73	404	15%	85%
Consultores	45	449	10%	95%
Comité de Producto	24	473	5%	100%

Figura 69: Datos capturados sobre origen de solicitudes de tipo error (Fuente: Resultado estudio del grupo)

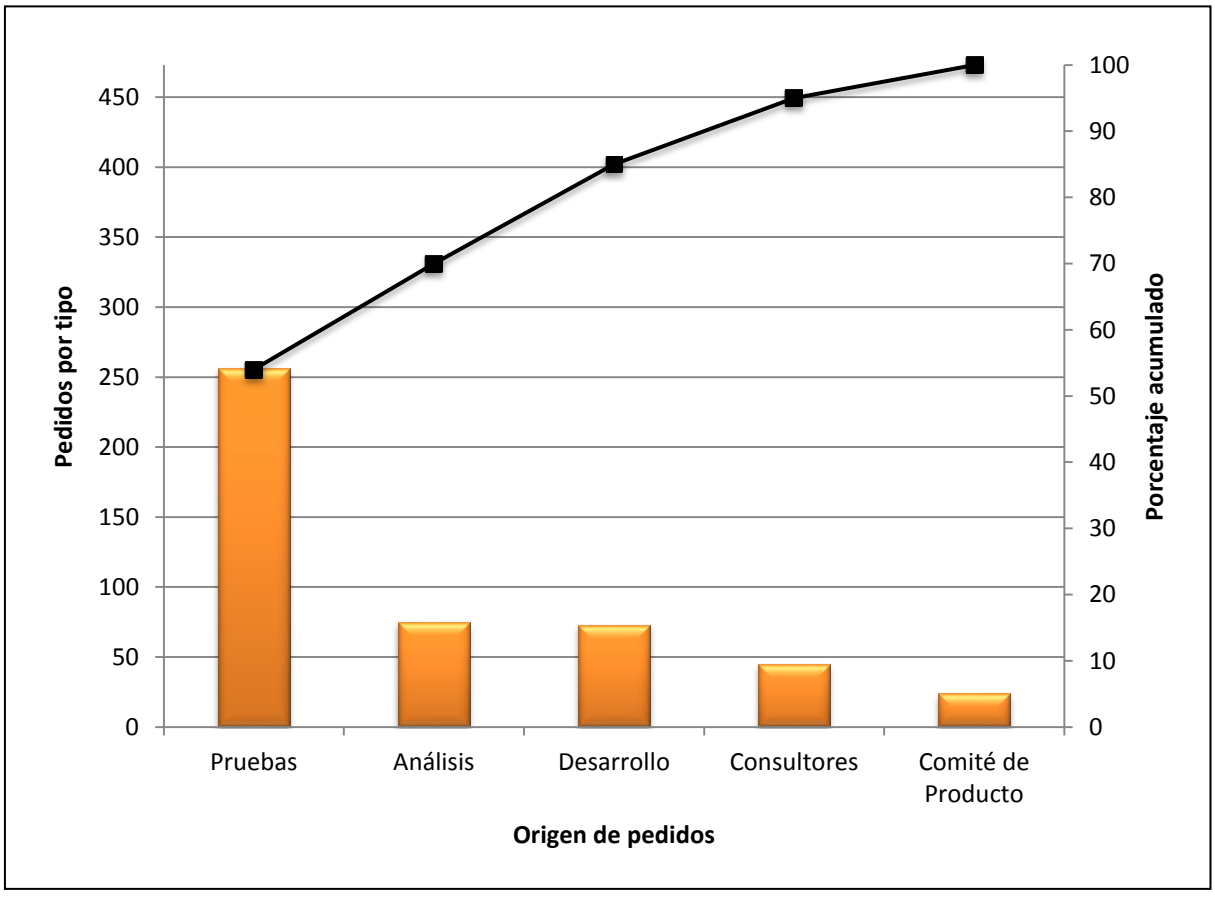

Figura 70: Gráfico de origen de solicitudes de error (Fuente: Resultado estudio del grupo)

4.4 Trabajo planificado vs no planificado

El trabajo planificado es aquel que se encuentra asociado a un pedido y que se encuentra registrado en la herramienta "Solicitudes de Requerimiento". En cambio, el trabajo no planificado es aquel que es solicitado directamente por un Cliente Interno o Externo e ingresa directamente al sprint.
En este gráfico se observa que un 33% aproximado del total de las tareas realizadas corresponden a solicitudes no planificadas, lo cual impacta negativamente en la duración estipulada del sprint, ya que los recursos humanos afectados son los mismos.

Sprint	Planificado	No Planificado
16	25	18
17	12	10
18	19	6
19	11	4
20	145	54
21	97	59
22	120	92
23	99	63
24	139	70
25	32	19
26	55	17
27	510	214

Figura 71: Datos sobre solicitudes por sprint (Fuente: Resultado estudio del grupo)

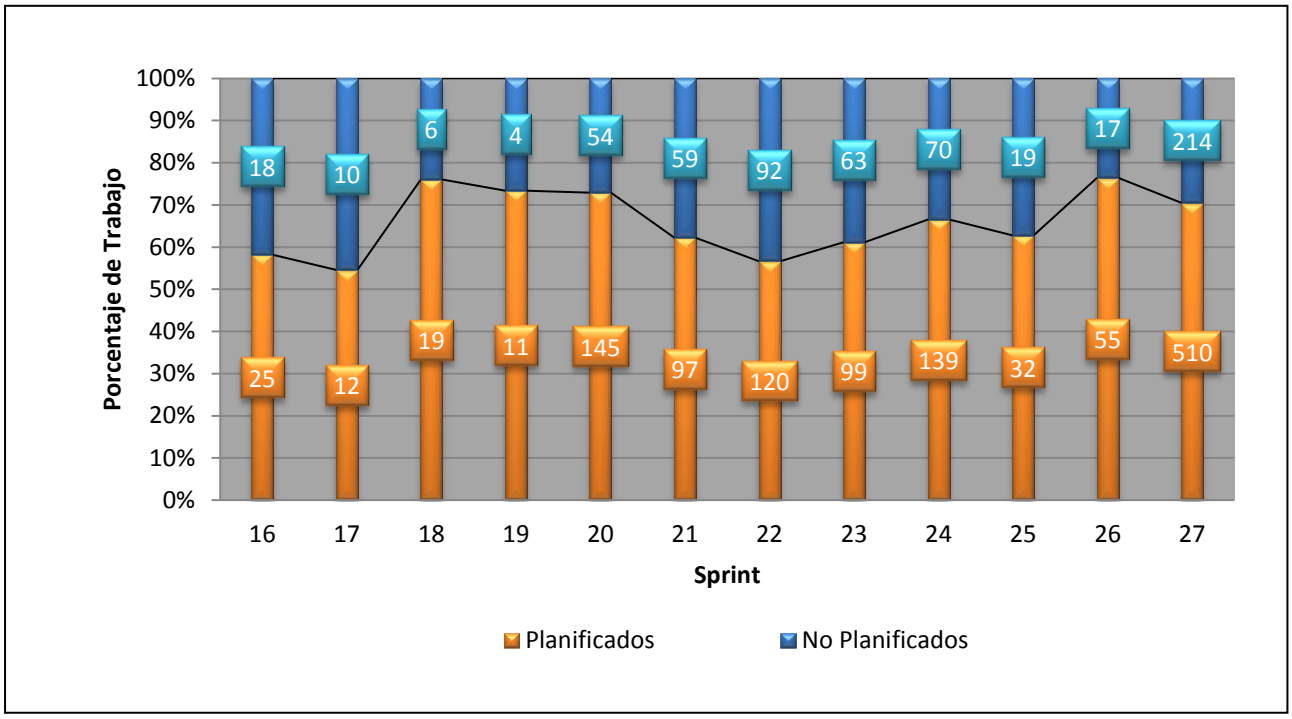

Figura 72: Gráfico solicitudes por sprint (Fuente: Resultado estudio del grupo)

4.5 Diagrama Causa – Efecto: Utilizamos este diagrama para identificar las posibles causas de nuestro problema específico (deficiencias en el proceso de desarrollo). Representamos los elementos (causas) del sistema que pueden contribuir al problema (efecto). Esta herramienta ayuda a lograr una concepción común del problema complejo, con todos sus elementos y relaciones claramente visibles.

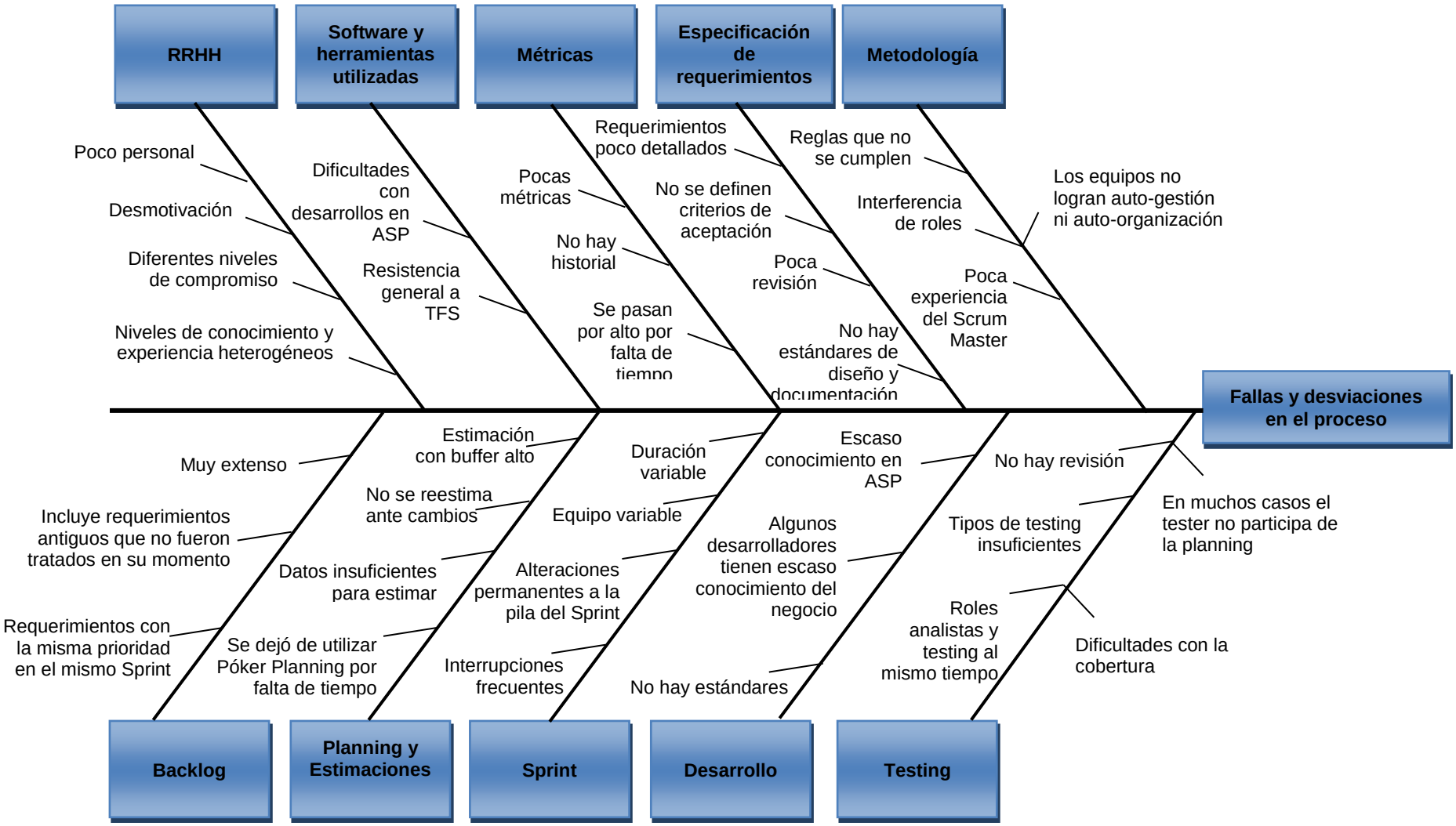

Figura 73: Diagrama Causa - Efecto (Fuente: Resultado estudio del grupo)

4.6 Matriz de Kelly: Esta técnica tiene su origen en la teoría de los constructos personales de Kelly, quién la diseña para el estudio de las relaciones interpersonales. A pesar de que su origen se vincula a este tipo de relaciones, con posterioridad ha sido utilizado con otros fines (estudios de mercado, elecciones de diferentes tipos, etc.). Nos permite explorar los sistema de constructos personales y constituye una forma de "ponerse en la piel de otras personas, ver sus mundos tal y como ellas los perciben, comprender su situación, sus preocupaciones e intereses" (Fransella y Bannister - 1977).

Constructos	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Constructos
Enfocado	5	7	7	5	5	5	4	2	5	5	4	1	1	2	No enfocado
Comprometido	3	4	4	2	4	3	4	6	2	5	2	1	1	1	No comprometido
Muy responsable	2	2	2	2	3	3	3	4	3	4	4	3	2	2	Poco responsable
Mucho conocimiento del negocio	1	2	1	4	3	4	5	6	3	5	6	7	7	6	Poco conocimiento del negocio
Mucho conocimiento del producto	1	2	2	6	3	6	5	7	3	5	6	7	7	7	Poco conocimiento del producto
Alta capacidad y experiencia en lo técnico	6	4	1	5	4	4	5	6	4	3	4	7	5	6	Poca capacidad y experiencia en lo técnico
Realiza múltiples tareas	1	2	1	2	2	2	4	6	2	2	4	6	6	5	No realiza múltiples tareas
Multidisciplinar	6	6	4	4	4	4	5	7	3	6	6	7	5	5	No multidisciplinar
Amplia disponibilidad horaria	7	6	7	5	6	6	4	3	6	5	5	4	4	4	Limitada disponibilidad horaria
Motivado	2	4	2	2	4	4	4	3	4	4	4	2	2	2	Desmotivado
Perceptible de interrupciones externas	1	2	2	2	2	3	5	6	1	2	4	4	6	5	No Perceptible de interrupciones externas
Atento a las necesidades del grupo	6	7	5	4	5	7	6	7	5	4	5	4	3	3	No tan atento a las necesidades del grupo
Estimaciones muy precisas	5	5	3	5	3	4	6	6	3	7	5	4	5	5	Estimaciones pocas precisas
Respetan los valores y principios ágiles	5	6	4	2	2	2	3	4	6	4	4	3	4	3	No respetan los valores y principios ágiles
PUNTUACIONES TOTALES	**51**	**59**	**45**	**50**	**50**	**57**	**63**	**73**	**50**	**61**	**63**	**60**	**58**	**56**	
RANGOS	**5**	**9**	**1**	**3**	**3**	**7**	**12,5**	**14**	**3**	**11**	**12,5**	**10**	**8**	**6**	

Figura 74: Matriz de Kelly (Fuente: Resultado estudio del grupo)

Escala de Puntuación:
1. Muy como el polo izquierdo
2. Bastante como el polo izquierdo
3. Un poco como el polo izquierdo
4. Punto medio
5. Un poco como el polo derecho
6. Bastante como el polo derecho
7. Muy como el polo derecho

Personal Área de Desarrollo:

A Product Owner
B Consultores
C Líder Técnico
D Scrum Master
E Analista Funcional Semi Sr. Avanzado
F Analista Funcional Semi Sr.
G Analista Funcional Junior 1
H Analista Funcional Junior 2
I Desarrollador Senior 1
J Desarrollador Senior 2
K Desarrollador Semi Sr. Avanzado
L Desarrollador Junior
M Desarrollador Inicial 1
N Desarrollador Inicial 2

Las Puntuaciones Totales consisten en la suma de los valores numéricos de cada columna (valoraciones recibidas por cada elemento).
Como las características o valores que se buscan satisfacer (particularidades de SCRUM) aparecen en los constructos de la izquierda, los valores próximos a 1 en la fila de rangos, indican que los profesionales se acercan al comportamiento deseado, es decir, son más compatibles con las características buscadas. Dado que en nuestra rejilla hay un total de 14 constructos, los valores más bajos (próximos a 14), indicarán los roles que mejor satisfacen los criterios que cumplen con la correcta implementación de SCRUM.

De acuerdo a la puntuación obtenida podemos observar que los roles que más se están adecuando a SCRUM son:

1. C - Líder Técnico
3. D – Scrum Master
3. E - Analista Funcional Semi Sr. Avanzado
3. I - Desarrollador Senior 1
5. A - Producto Owner
6. N - Desarrollador Inicial 2

Los que más se alejan de las características deseadas:

14. H - Analista Funcional Junior 2
12,5 G - Analista Funcional Junior 1
12,5 K - Desarrollador Semi Sr. Avanzado
11 J - Desarrollador Semi 2
10 L - Desarrollador Junior

4.7 Trabajo de Campo

Durante las reuniones en GobFactory, fue posible tomar contacto personalmente con algunos empleados y a través de entrevistas y encuestas, se pudo indagar sobre el trabajo que se realiza y el modo en que éste se lleva adelante. También se pudo acceder a documentos digitales en los cuales se registra el seguimiento del proyecto (Documento de inicio, Historias, defectos y tareas planeadas, Configuración, Interrupciones, Capacidad, Evolución, Notas de seguimiento diario, etc.).

4.7.1 Ambiente físico

Las instalaciones de la empresa son amplias, con buena iluminación y cómodas. Tiene dos pisos, en el inferior se encuentra el Área de Desarrollo y Soporte Técnico, en la cual trabajan aproximadamente 14 personas. En el piso superior, se encuentra Administración y Atención al Cliente (Consultores) y trabajan 18 personas. Hay dos cocinas – una arriba y otra abajo - que a la vez están ambientadas para espacios de reunión/capacitación. Se está proyectando una sala de dispersión/entretenimiento.

4.7.2 Clima de trabajo

En el Área de Desarrollo el ambiente de trabajo es sumamente silencioso, tranquilo, la mayoría de las personas usan auriculares, cada uno sentado en su escritorio y concentrados en su propio trabajo. Hay poca comunicación verbal durante las horas de trabajo. Muchas de las comunicaciones se desarrollan vía Skype, aun cuando las personas están muy cercanas en el espacio físico. Son escasas las reuniones de análisis y validaciones internas. En general, el equipo no tiene una comunicación fluida y constante durante la iteración, en lugar de avanzar en equipo, el desarrollador espera que el analista se acerque a su escritorio con consultas y/o con la historia que debe desarrollar (Estas son consideradas, en general, secuelas del ciclo de vida cascada bajo el modelo CCMI). Se ha hecho evidente que el ingreso de nuevos RRHH, mejora notablemente el clima del equipo, aportando nuevas ideas y formas de trabajo, estas personas son más dinámicas, más comunicativas e interactivas, promueven el intercambio fluido y la integración del equipo.
No obstante, la empresa se encuentra en un proceso de cambio e integración en el Área de Desarrollo, que involucra un aumento y mejora en la comunicación interna, otras dinámicas de grupo, entretenimientos para los momentos libres, y patio con mesas para trabajar o para charlas formales e informales. Por último, se han comenzado a implementar reuniones donde se analizan aspectos que puedan mejorar la motivación.

En las áreas del piso superior, el clima es más relajado, dinámico, de más movimiento. Quizá relacionado al tipo de actividades que se realizan y al perfil de las personas que componen estas áreas. Las tareas son de atención al cliente, administración y gerencia. Están continuamente en contacto intercambiando información, dando soporte a clientes y siempre movilizados, distendidos.

4.7.3 Producto

- Se trabaja sobre un único producto que gestiona la mayoría de las áreas administrativas municipales. Es adaptable a particularidades. Escalable, simple, desarrollado en ASP y .Net.

- Los nuevos empleados reciben, en un principio, una introducción al producto. Luego se capacitan y adquieren mayor conocimiento a medida que van abordando cada nuevo requerimiento que ingresa. Hay referentes sobre temáticas específicas (consultores) para áreas como por ejemplo contable, sueldos y rentas.

- El jefe de producto se queja sobre la alta tasa de defectos que regresan del cliente. Esto evidencia deficiencias tanto del área de testing, análisis y del cliente interno o consultores. Esta alta tasa limita nuevos desarrollos.

- Hay módulos del sistema que tienen escasa o nula documentación. La única forma de conocer el funcionamiento es leyendo el código software (ej. configuración de la generación de cedulones).

4.7.4 Proceso y Metodología

- Escaso conocimiento del personal sobre la misión y visión de la empresa impactando en personal no identificado con la misma.

- Si bien la metodología SCRUM no exige parámetros estructurados, el modo en que se implementa no está permitiendo alcanzar los objetivos establecidos. Existen aspectos claros que no se cumplimentan.

- Algunas personas entrevistadas manifiestan la necesidad de realizar ajustes/mejoras sobre el proceso actual (definiciones sobre ciclos de validación, duración de reuniones diarias, ausencia de los testers en las planning, etc.) que logren mejorar el desempeño del equipo y el resultado del sprint.

- En relación al proceso definido, se identifican etapas que en la práctica no se cumplen o son deficientes (presentación de los requerimientos solicitados, validación de los criterios de aceptación, validación del cumplimiento de los criterios de aceptación, reunión de presentación y reunión de retrospectiva). Algunas de ellas se dejaron de realizar por falta de tiempo, por falta de control o por considerarse no tan imprescindibles.

  Este proceso fue definido ante la necesidad de cumplir con un alto nivel de calidad a un cliente determinado. En la actualidad, si bien se intenta

mantener la calidad del proceso, esto queda supeditado a los criterios propios del analista funcional.

- La gestión de post it deberían tener actualización permanente (cada vez que se toma o termina una tarea), pero la realidad indica que se realiza individualmente y a última hora o durante la reunión diaria. El tablero no refleja el trabajo real, lo que impacta negativamente en el conocimiento que todo el equipo debería tener sobre dónde están parados.

- En la reunión de planificación no siempre participan todos los miembros dado que algunas personas pueden encontrarse realizando otras actividades. Como consecuencia de esto, se considera que la persona ausente comenzará la iteración con un nivel de conocimiento más acotado que el resto del equipo. Esta aclaración se realiza, ya que si la persona es, por ejemplo, el tester, las pruebas se realizarán bajo la perspectiva de la historia, no tiene una visión crítica que pueda incrementar su cobertura de pruebas con aspectos de negocio adicionales.

- Durante el transcurso del sprint ingresan solicitudes de urgencia que suelen agregarse o reemplazar aquellas que fueron tomadas (y comprometidas) por el equipo en la reunión de planificación. Como consecuencia de esto, el equipo debe adecuar el sprint planeado a las nuevas tareas, descartando aquellas que son imposibles de hacer en este tiempo. Si el Product Owner lo considera necesario, se postergan las fechas de cierre, o bien, la iteración se ajusta acotándola solo a los requerimientos de urgencias (independientemente de la planificación realizada previamente).

- Las prácticas de revisión y validación de historias con el Product Owner suelen no realizarse ya que éste se encuentra afectado por numerosas actividades y ante la demora, se procede a validar aspectos puntuales con el Responsable del Área de Desarrollo, u alguna persona indicada por el Product Owner para tal fin. Esto en ocasiones trae como consecuencia un desarrollo inadecuado o insuficiente de algunos aspectos de la solución. Tampoco se están realizando revisiones de a pares o del Responsable de Calidad sobre las historias.

- El código no se realiza bajo estándares, dejando la programación a criterio y costumbres del desarrollador. En ocasiones, los desarrolladores de mayor seniority advierten sobre prácticas de poca performance u errores básicos de programación.

- Las pruebas son escritas por el tester sin ser validadas por el resto del equipo. Algunos analistas exponen en la historia aspectos importantes que

deben ser cubiertos obligatoriamente por las pruebas (algo parecido, pero sin serlo, un criterio de aceptación).

- Los criterios de aceptación del requerimiento no están siendo incluidos en la historia. Inicialmente, estos criterios eran definidos por el Product Owner. Luego se dispuso que fueran identificados por el analista funcional y validado con el Product Owner. Con el tiempo, esta práctica fue desapareciendo dado que no se validaban, eran insuficientes o no se disponía de tiempo para su documentación.

- Al finalizar el sprint, sólo algunas soluciones son presentadas a los Consultores y/o Product Owner. Generalmente, se presentan aquellas que se consideran de gran impacto en el cliente, que presentan cierta complejidad o bien, es delicada su implementación (requieren de una alta participación del consultor en el despliegue por tener muchas configuraciones). El Product Owner indica si las observaciones realizadas durante esta reunión se resolverán antes o después del despliegue (en este último caso, ingresando como nuevas solicitudes de requerimientos).

- Las tareas de refinamiento de backlog la realiza el comité de forma semanal.

- Se hace evidente la preocupación del equipo por la capacidad de liderazgo de los mandos medios. Hay dificultades en la comunicación entre el jefe de desarrollo y su equipo, por lo cual permanentemente delega esta tarea en el Scrum Master. Las reuniones de área solo se llevan a cabo cuando la gerencia quiere transmitir requerimientos, reclamos y ajustes de procedimientos/reglamentaciones internas, que la gente modere su productividad, etc. Generalmente no se tratan temas de interés del personal (necesidades de capacitación, sugerencias de mejoras, etc.)

- De las entrevistas/observaciones surge que hay en el área de desarrollo, una disparidad entre los miembros del equipo en cuanto al nivel de conocimiento técnico y de negocio. Estas personas con mayor conocimiento, tienen que realizar tareas de entrenamiento de nuevo personal, por lo que tienen una mayor carga de trabajo generando en ocasiones, cuellos de botella. Esto, implica un impacto negativo en su capacidad ya que suele no registrarse en las estimaciones, de este modo también se afecta la precisión de las fechas de entrega.

- El personal del Área de Desarrollo no tiene afinidad y muestran descontento con la herramienta de trabajo TFS, utilizada para la carga de horas, carga de historias o defectos y como repositorio del producto.

4.7.5 Análisis de documentación:

Como se mencionó anteriormente el área de desarrollo trabaja bajo un marco SCRUM.

El seguimiento del proyecto se realiza a través de un tablero en el que se encuentran detalladas las actividades en curso. Cada miembro del equipo es el encargado de mover sus tareas de acuerdo a su avance diario. Al final de cada reunión diaria, el Scrum Master actualiza el gráfico Burn-Down en el cual se refleja el avance total del equipo a los fines de dejar evidencia clara de los desvíos o éxitos del sprint. Finalmente, toma una foto de la situación del tablero en ese momento para su posterior análisis, y lo guarda en un directorio del proyecto en el repositorio del producto (a través de la herramienta TFS).

Antes de cada reunión diaria, el Scrum Master realiza la impresión del documento “Evolución y tasa de evolución <fecha> - Avance del día xx de yy - zz Semana” en el cual se observa el progreso del sprint a la fecha actual. Adicionalmente, se presenta un resumen de las horas realmente estimadas por cada miembro y el avance real de los mismos.

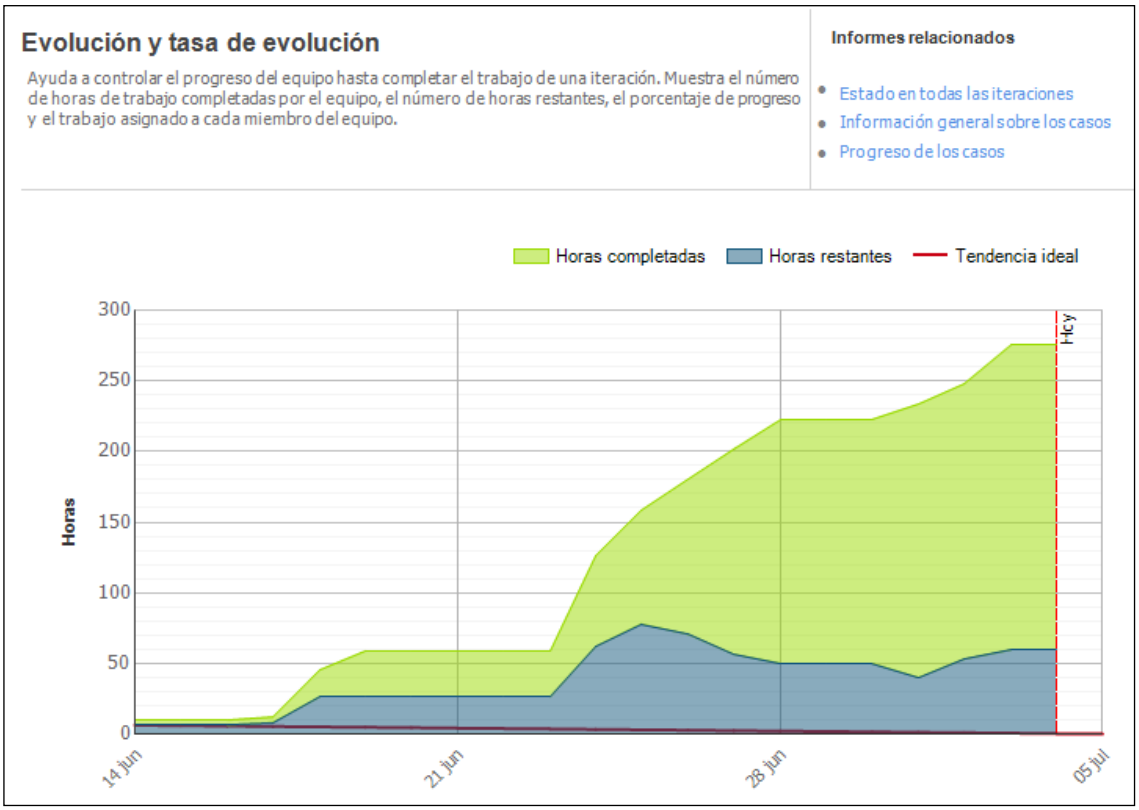

Figura 75: Ejemplo Evolución y tasa de evolución (Fuente: GobFactory - Herramienta TFS)

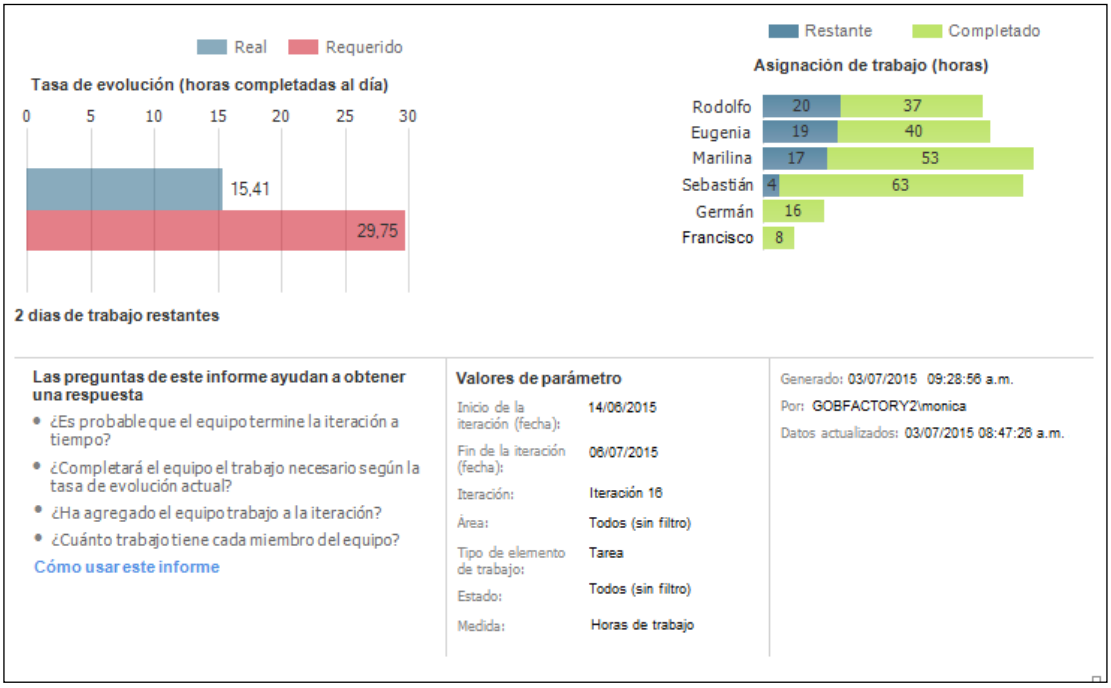

Figura 76: Ejemplo Tasa de evolución total y por persona (Fuente: GobFactory - Herramienta TFS)

Adicionalmente, el Scrum Master mantiene un Excel con el Seguimiento del Proyecto por cada sprint que se realiza, en él tiene:

- Un detalle de cada uno de los requerimientos que se llevarán a cabo con sus tareas, los responsables, el tiempo estimado de trabajo y el porcentaje de avance-faltante.

- Un detalle de las capacidades y disponibilidades de horas por cada persona del equipo. En este apartado, se cuenta también con información sobre las ausencias por licencias o retiros anticipados. Adicionalmente, contiene un gráfico en el cual se refleja claramente la situación de cada miembro del equipo (si se encuentra sobrecargado de tareas y necesita una re planificación de tareas, o bien, se encuentra con tiempo disponible para nuevas tareas o para dar soporte a miembros con sobrecarga).

- Un resumen de cada una de las reuniones diarias, discriminando lo expresado por cada miembro: que tareas realizó el día anterior, que va hacer hoy y sus interrupciones.

Proyecto: MUNI16 **Servidor:** Rack\WebGobCollection **Consulta:** 09. Trabajo Pendiente IT. 16 **Tipo de lista:** Árbol

ID	Rango en la pila	Tipo de elemento de trabajo	Título 1	Estado de tarea	Motivo	Asignado a	Estimación original	Trabajo restant	Trabajo completad
1242	1010	Historia de usuario	It14.H05. Página de LOGIN del Sistema	Resuelto	Pruebas de código y unitarias superadas	Eugenia			
1258	975	Historia de usuario	It15.H01. Débito Automático Banco Patagonia	Resuelto	Pruebas de código y unitarias superadas	Eugenia			
1266		Historia de usuario	It15.H05 Impuesto a las Ganancias - Tipos Gastos Deducibles	Resuelto	Pruebas de c	Marilina			

Figura 77: Ejemplo Seguimiento de Tareas (Fuente: GobFactory - Herramienta TFS)

Ejemplo Configuración Inicio y Fin de Iteración:

Configuración Ocultar instrucciones

Use esta hoja de cálculo para comenzar la planeación de la iteración. Para obtener instrucciones detalladas sobre cómo usar este libro, visite: http://go.microsoft.com/fwlink/?LinkId=145659
Paso 1: Seleccione el **Área** y la **Iteración** adecuadas.
Paso 2: Especifique la **Fecha de inicio** y la **Fecha de finalización** de la iteración.
Paso 3: Cambie a la hoja de cálculo **Interrupciones** para especificar las interrupciones en la iteración.

Área e iteración

Paso 1	
Área	\
Iteración	\ Iteración 16

Fechas

Paso 2	
Fecha de inicio	14/06/2015
Fecha de finalización	06/07/2015
Días*	16

* Indica las vacaciones especificadas para el equipo en la pestaña Interrupciones.

Figura 78: Ejemplo configuración inicio y fin de iteración (Fuente: GobFactory - Herramienta TFS)

Ejemplo Interrupciones:

Interrupciones Ocultar instrucciones

Use esta hoja de cálculo para describir las interrupciones planeadas en la programación del equipo para esta iteración.
Paso 1: Agregue **Días** que las personas piensan no trabajar en la tabla **Interrupciones planeadas**.
Paso 2: Agregue las fiestas o los días cuando todo el equipo no trabajará durante esta iteración en la tabla **Vacaciones**. La columna **Días** de la hoja de cálculo **Configuración** no tendrá en cuenta estos días en el número total de días de la iteración.
Paso 3: Cambie a la hoja de cálculo **Capacidad** para continuar.

Interrupciones planeadas

Miembro del equipo	Descripción	Fecha de inicio	Fecha de finalización	Días	Días restantes

Vacaciones

Descripción	Fecha

Figura 79: Ejemplo interrupciones (Fuente: GobFactory - Herramienta TFS)

Ejemplo Capacidad del Equipo:

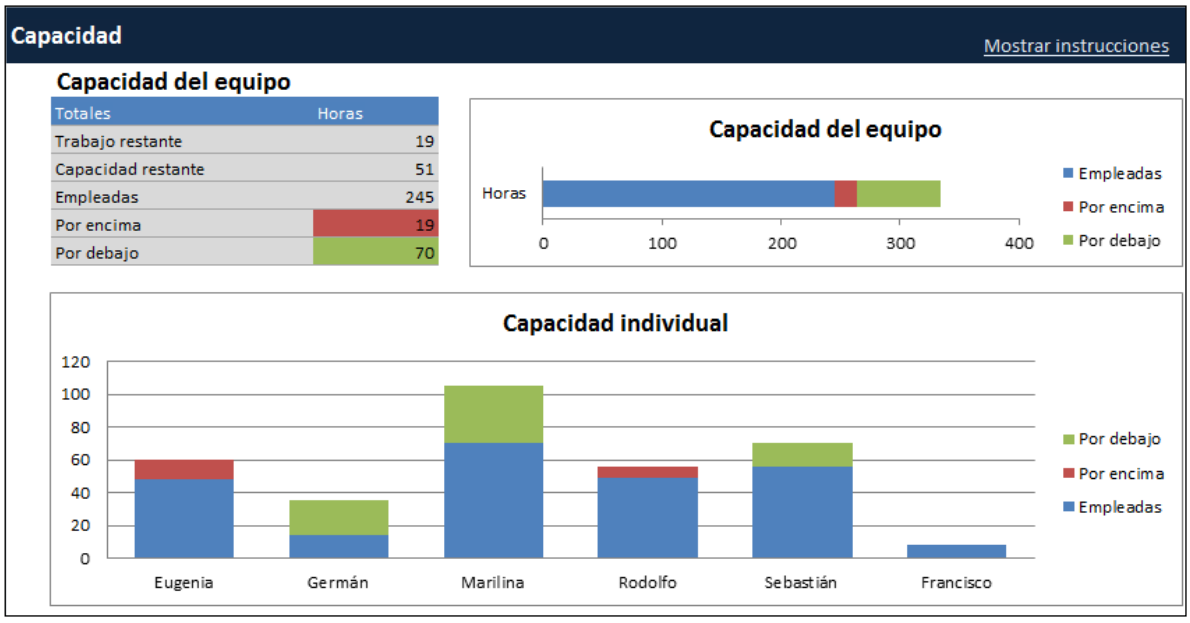

Figura 80: Ejemplo gráfico de capacidad del equipo y por persona (Fuente: GobFactory - Herramienta TFS)

Capacidad individual

Miembro del equipo	Horas/día	Días	Capacidad	Asignadas	Empleadas	Por encima	Por debajo
Eugenia	3	16	48	60	48	12	0
Germán	7	5	35	14	14	0	21
Marilina	7	15	105	70	70	0	35
Rodolfo	7	7	49	56	49	7	0
Sebastián	7	10	70	56	56	0	14
Francisco	2	4	8	8	8	0	0

Figura 81: Ejemplo capacidad del equipo y por persona (Fuente: GobFactory - Herramienta TFS)

Por último, el Scrum Master confecciona al comienzo de cada iteración un documento de inicio, en el que se identifican las personas que participaran en el sprint, los requerimientos (con sus solicitudes asociados) que fueron tomados y sobre los cuales existe compromiso por parte del equipo, los encargados de cada requerimiento, información sobre los entornos de desarrollo, prueba y reléase. Las bases de datos de clientes que es necesario utilizar mínimamente para las pruebas (esto es, las pruebas deben realizarse mínimamente sobre la BD del cliente que solicito el requerimiento). Por último, cualquier otro tipo de información u hito que sea relevante para la iteración.

4.8 Informe de Situación FODA

Para entender los principales problemas de un proyecto así como sus potencialidades se realizará un análisis FODA que permitirá conocer la situación de la empresa analizando sus características internas (debilidades y fortalezas) y su situación externa (amenazas y oportunidades). Facilita la identificación de factores tanto internos como externos que puedan condicionar el éxito del /los proyectos.
La información volcada en el cuadro fue obtenida de las entrevistas con las Gerencias y la Scrum Master.

	Positivos Para alcanzar el objetivo	**Negativos** Para alcanzar el objetivo
Origen interno (atributos de la empresa)	**Fortalezas** • Experiencia • Conocimiento del negocio • Solidez empresarial • Cobertura del mercado • Recursos humanos • Buena relación con los clientes • Producto parametrizable, se adapta a distintos tipos de municipios • El sistema tiene amplia gama de prestaciones. • El sistema cubre y relaciona la mayoría de las áreas administrativas. • Asesoramiento sobre gestión municipal. • Asesoramiento sobre hardware • Instalación y mantenimiento de servidores de clientes • Estable, resistente a fallos.	**Debilidades** • Dificultades con la implementación de la metodología de trabajo • Lenguaje de programación del sistema • No se persigue la innovación en la estética y/o tecnología del producto • Es compleja la adaptación del producto según las particularidades de cada municipio • Poco personal y mucha demanda de los clientes. • Más atención a la comercialización que al desarrollo. • Alta tasa de defectos desplegados en el cliente.
Origen externo (atributos del ambiente)	**Oportunidades** • Desarrollo del mercado • Ampliación de los servicios que deben ofrecer las municipalidades • Subsidios para financiamiento de proyectos de software • Recomendaciones entre municipios	**Amenazas** • Crecientes desarrollos orientados a web • Nuevas tecnologías y servicios • Nuevas empresas ampliamente calificadas • Cambios de gobierno pueden implicar que los municipios deban cambiar de proveedor • Recortes de presupuesto a las municipalidades

Figura 82: FODA (Fuente: Resultado estudio del grupo)

4.9 Aplicación de SCRUM en GobFactory

El siguiente cuadro compara las características de SCRUM con las prácticas implementadas por GobFactory a los fines de identificar las desviaciones/diferencias entre la práctica actual y lo recomendado. Esta comparación fue posible tras haber entrevistado los diferentes roles del equipo de desarrollo.

		Características de SCRUM	Aplicación en GobFactory
Contexto		Genera un contexto relacional e iterativo, de inspección y adaptación constante para que los involucrados vayan creando su propio proceso.	Se cumple medianamente. El área definió un proceso, el cual fue evolucionando de acuerdo a la realidad del momento. Al día de hoy, se han dejado de aplicar actualizaciones por lo que éste no refleja el proceso real.
Principios de SCRUM	Individuos e interacciones por sobre procesos y herramientas	Se apoya en la confianza hacia las personas, sus interacciones y los equipos.	Se cumple medianamente, se está trabajando en potenciar al equipo.
	Software funcionando por sobre documentación exhaustiva	La documentación es un documento intermedio sin valor de negocio.	Se cumple medianamente. La documentación que se utiliza incluye Historias con detalle excesivo (similar a un CU), Casos de Pruebas, Manual de Configuración e Instructivos Internos Técnicos para los Consultores. A la vez, se exige que se incluya en todos los casos aspectos de negocio y soluciones operativas.

	Colaboración con el cliente por sobre la negociación de contratos	Para el desarrollo ágil el valor del resultado no es consecuencia de haber controlado una ejecución conforme a procesos, sino de haber sido implementado directamente sobre el producto. Un contrato no aporta valor al producto. Es una formalidad que establece líneas divisorias entre responsabilidades, que fijan los referentes para posibles disputas contractuales entre cliente y proveedor. En el desarrollo ágil el cliente es un miembro más del equipo, que se integra y colabora en el grupo de trabajo. Los modelos de contrato por obra no encajan.	Debido a la cantidad de clientes, la empresa cuenta con un área de consultoría (Clientes Internos). El servicio que le ofrece la empresa al cliente se compromete a brindar una solución de gestión incluyendo software y circuitos operativos. El contrato que se realiza es consultoría y mantenimiento. En la empresa existe el rol de Cliente Interno (Consultor) que es el intermediario o interprete entre el equipo y el cliente externo. Los consultores colaboran con el equipo de desarrollo.
	Respuesta al cambio por sobre el seguimiento de un plan	La gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.	El producto tiene una evolución constante, durante la cual se realiza mantenimiento y se agregan nuevas prestaciones que agregan valor al producto.

		Características de SCRUM	Aplicación en GobFactory
Roles	**Product Owner**	Trabaja colaborativamente con el resto del equipo para asegurarse que el producto final tenga la mayor cantidad posible de valor.	Posee un alto nivel de conocimiento del negocio y del producto, lo que le permite visualizar y proponer permanentemente mejoras. La dificultad, es que tiene una limitada disponibilidad de horarios. Esto provoca retardos en la velocidad del equipo dado que ante la necesidad de revisiones y/o toma de decisiones, debido a su ausencia, se producen interrupciones en el ciclo.
	Equipo de Desarrollo	Es autoorganizado y autogestionado. Determina la forma en que se realizará el trabajo y cómo resolverá cada problemática.	Los equipos no logran autogestionarse ni autoorganizarse. Necesitan de la constante presencia del Scrum Master.
		Cada persona del equipo debe ser multifuncional y multidisciplinar. No existen especialistas exclusivos, sino más bien, individuos generalistas con capacidades especiales.	Se cumple medianamente en los roles de Análisis y Testing. Desarrollo no logra realizar otras funciones (Ej. Falencias en las pruebas unitarias). Se está trabajando para mejorar su crecimiento como tester.
	Scrum Master	Velar por que todos los participantes del proyecto sigan los valores y principios ágiles, las reglas y proceso de SCRUM y guiar la colaboración intraequipo y con el cliente de manera que las sinergias sean máximas	Se cumple medianamente, ya que no se logra cumplimentar todos los valores.
		Quitar los impedimentos que el equipo tiene en su camino para conseguir el objetivo de cada iteración (proporcionar un resultado útil al cliente de la manera más efectiva) y poder finalizar el proyecto con éxito	Se cumple medianamente, aún no logra que el Product Owner esté disponible para las revisiones y validaciones, o que éste asigne a otro referente.
		Proteger y aislar al equipo de interrupciones externas durante la ejecución de la iteración (introducción de nuevos requisitos, asignaciones imprevista de otras tareas a un miembro del equipo, etc.).	Se cumple medianamente. La necesidades del área de Consultoría suelen requerir prioridad máxima.

		Características de SCRUM	Aplicación en GobFactory
Elementos de Scrum	**Product Backlog**	Es el conjunto de todos los requisitos de proyecto, incluye funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de las sucesivas iteraciones de desarrollo. Se encuentran priorizados según su retorno sobre la inversión.	El Backlog se encuentra priorizado, en primer lugar, por urgencia (errores críticos), por tipo de solicitud (error, modificación, nuevo) y por cliente. Posteriormente, se atienden aquellas solicitudes que implican nuevas y mejores prestaciones y que van a poder impactar en todos los clientes.
	Sprint Backlog	Es la lista que descompone las funcionalidades del Product Backlog en las tareas necesarias para construir un incremento: una parte completa y operativa del producto. Se asigna a cada tarea el personal que la ejecutará, y se indica el tiempo de trabajo que resta para terminarla.	Se cumple.
	Sprint	Requiere que al final de cada Sprint se entregue un producto funcionando.	Se cumple.
		Sprint de duración fija preestablecida entre 1 a 4 semanas como máximo.	Se realizan iteraciones que varían entre 2 y 8 semanas. Esto dificulta tomar métricas que puedan ser comparables.
		Todo el equipo tiene toda la información necesaria para poder tomar decisiones informadas sobre el proyecto en cualquier momento.	Suele realizarse la planning y el comienzo de la iteración sin que todo el equipo esté presente.
	Sprint Planning Meeting	Al momento de la planning el Product Owner debe tener preparado el backlog del producto.	Se cumple, pero se suelen priorizar requerimientos (solicitudes) que surgen de imprevistos y que son ingresados por el Dueño del Producto alterando los requerimientos a los que se comprometió el equipo.
		Los equipos identifican lo que hay que hacer y toman la responsabilidad de hacerlo.	Se cumple.

	Sprint Planning Meeting	Se requiere compromiso de todo el equipo al comienzo de cada Sprint.	Se cumple medianamente. Existen casos en los que no participa el desarrollador o tester en esta reunión (dado que se encuentran en otros desarrollos u actividades). Esto produce que el tester deba planificar y ejecutar sus pruebas en base a la historia, no recibiendo información adicional para una visión más crítica de la solución o del negocio. El desarrollador se limita a lo que indique el analista.
		Es recomendable que las actividades duren menos de un día. Esto permitirá detectar bloqueos o retrasos durante las reuniones diarias.	Las actividades en ocasiones duran más de un día. Las tareas no son correctamente desglosadas debido a que el equipo no utiliza la herramienta disponible de gestión no la consideran apropiada. En cuanto al proceso de estimación se agrupan las tareas, no desmenuzándose correctamente porque implica más esfuerzo en el proceso. Se estima aplicando el criterio de juicio de experto y siempre se estima con buffer. Durante varios sprint se utilizó Póker Planning para estimar, con lo cual se logró disminuir la brecha de estimaciones errores, pero aún se producían desviaciones. Por cuestiones de tiempo, esta técnica dejó de utilizarse.
		El equipo de desarrollo debe proveer las estimaciones de cuanto esfuerzo será requerido para cada una de las características del producto. Para ello utiliza su capacidad productiva (velocidad) obtenida de los Sprint pasados.	El Dueño del Producto suele establecer fechas de finalización menores a las que sugiere el Equipo de Desarrollo. Se inician las tareas tratando de cumplir con los tiempos impuestos. Los equipos son variables en su composición y cantidad, lo que impide tomar como referencia la velocidad.

	Sprint Planning Meeting	Al momento de la planning el Equipo tiene un conocimiento de las tecnologías empleadas y, del negocio del producto suficiente para realizar estimaciones basadas en "juicio de experto", y para comprender los conceptos de negocio que expone el propietario del producto.	Se estableció que al momento de la planning el equipo debe adquirir al menos el 60% del conocimiento (técnico y de negocio) para poder estimar utilizando "juicio de experto" o "Póker Planning".
	Scrum Diario	En la reunión diaria todos y cada uno de los miembros toman turnos para responder las preguntas: - Que hice desde la última reunión diaria hasta ahora. - Que voy hacer desde ahora hasta la próxima reunión diaria. - Que impedimentos o problemas tengo.	El equipo no se siente cómodo en estas reuniones en las que debe hablar y exponer su avance. El Scrum Master repite las preguntas por cada persona y éstas se limitan a responder lo mínimo. En muchos casos se omite informar si existen impedimentos y esto compromete la finalización del sprint. En otros casos, algunas personas se extienden con su explicación técnica (que no involucra a todos), lo que genera distracciones y falta de atención en lo que se está diciendo. El lugar físico tampoco es el adecuado para este tipo de reuniones.
		Se recomienda que la reunión diaria tenga una duración aproximada de 15 minutos.	La duración de la reunión suele ser entre 20-40 minutos para un equipo de aproximadamente 8 personas. Se tratan temas técnicos que no son relevantes a todo el equipo y que deberían ser tratados en una reunión específica.
		El progreso del proyecto se mide en base al producto terminado.	Se cumple medianamente, dado que en las reuniones diarias, no todos los miembros del equipo informan lo que les falta para cumplir el objetivo del sprint. Como soporte, la registración de las tareas no se realiza, por lo tanto, no es posible ver el esfuerzo pendiente ni medir el avance real.

	Revisión de Sprint	Al final del sprint se realiza una reunión de revisión del producto construido durante el sprint. Participan el Product Owner y otros involucrados (Stakesholder). Los interesados utilizan y evalúan el producto durante la reunión. De la reunión de revisión (presentación), se obtiene retroalimentación, la cual debe ingresar como PBIs. Los nuevos PBIs deben ingresar estimados, los ya existentes con la misma estimación (o similar) deben ser eliminados para no incurrir en el incremento desmedido del alcance (scape creeping). Si se agrega trabajo, debe quitarse trabajo del otro lado.	No se realizan reuniones de revisión del producto. Esto ocasiona que los consultores al momento de implementar no cuenten con el conocimiento suficiente para el despliegue y para transmitir al cliente el potencial de la solución. Además existan soluciones que vuelven a ingresar por no cumplimentar 100% las expectativas del cliente. Esta revisión no se realiza ya que la gerencia exige presentaciones formales, lo que llevaría a asignar al sprint más tiempo para la preparación de las mismas.
	Retrospectiva	Deben realizar reunión de retrospectiva inmediatamente después de la revisión. La revisión es el "qué", la retrospectiva el "cómo".	Se cumple medianamente. Hay casos en que una retrospectiva incluye más de dos iteraciones. Se trata el "qué se hizo bien", "qué se hizo mal", "el cómo" y "cómo mejorar". En algunos casos, las mejoras identificadas durante esta reunión no son implementadas porque fueron detectadas fuera de tiempo, generando falta de interés en aportar ideas y mejoras en futuras reuniones.
	Refinamiento del Product Backlog	El refinamiento del Backlog es una actividad constante a lo largo del Sprint. Pero algunos equipos prefieren una reunión durante el Sprint. El Product Owner puede establecer más reuniones de esta durante el Sprint.	El refinamiento lo realiza el Comité y Dueño del Producto en base a la información entregada por el Scrum Master una vez a la semana, y eventualmente el equipo de puede realizarlo en cualquier momento del sprint.

Prácticas Recomendadas	**Todo el equipo es responsable por la calidad del producto**	Se cumple medianamente. La responsabilidad de la calidad del producto está a cargo del tester. Éste, según el caso, responde ante defectos que se desplegaron en los reléase. Aún no existe como filosofía de trabajo que todo el equipo debe responder ante esta situación. Cabe aclarar que, en la práctica, existen analistas funcionales que terminan el análisis actual y salen del equipo para encarar nuevos desarrollos. El tester -y el desarrollador- cuentan con tiempos de soporte del analista pero no trabajan en conjunto. Por ello, se interpreta que los defectos son responsabilidad del tester (salvo que el defecto corresponda claramente a error de análisis.
	Automatización de pruebas unitarias	No se cumple. Las pruebas las realiza el desarrollador aplicando su propio criterio. Existe un alto porcentaje de errores o fallas resultantes del deficiente y acotado test unitario (en muchos casos no se llegan a cubrir los caminos de éxito). Esto trae como inconveniente la mala calidad de los test funcionales, dado que parte del tiempo destinado a estas pruebas, se pierde por defectos en las interfaces, en las funciones básicas, de comunicaciones, etc. que podrían evitarse con un buen test unitario.
	Automatización de pruebas de sistemas	No se cumple. Las pruebas las realiza el tester en forma manual. La cobertura es establecida por el tester sin ninguna participación del Product Owner o del resto del equipo, existiendo la posibilidad de que no se cubran todos los escenarios que deberían. Las pruebas automáticas no son una posibilidad en la actualidad, dado que el equipo es reducido y existe gran demanda de trabajo, no restando tiempo para encarar tareas de investigación y/o de implementación sobre éstas pruebas.
	Integración continua	Se cumple de forma no automatizada.
	Aprendizaje implícito	Se cumple.
	Los equipos deben mantenerse fijos a los fines de contribuir a la obtención de métricas	Se cumple en parte. Generalmente, los equipos se deshacen al finalizar los sprint o se producen entradas y salidas de éstos durante el sprint. Todas estas situaciones contribuyen a que no se pueda controlar bien la velocidad de trabajo del equipo.

4.10 Problemas detectados en el Área de Desarrollo

A los fines de poder identificar aquellos aspectos que puedan estar interfiriendo en el óptimo desarrollo del proceso, se llevó a cabo una investigación orientada a la valoración de las etapas del proceso implementado. La misma se realizó sobre una muestra representativa y participaron miembros de todas las áreas involucradas. Se recopilaron testimonios orales a través de entrevistas y encuestas en soporte papel con el propósito de averiguar sobre las prácticas, opiniones y actitudes.

4.10.1 Área de Análisis:

Dificultad para realizar tareas de exploración ante un requerimiento: Como una buena práctica del Área de Análisis se definió que al momento de recibir y especificar un requerimiento de cualquier tipo (nuevo, modificación o error), los analistas deben verificar el comportamiento actual del sistema (si aplica) y la documentación anterior existente (casos de usos, lectura de código, instructivos o manuales de configuración). Esta práctica se implementa a los fines de facilitar la especificación y la comprensión del requerimiento, permite determinar si el mismo no ha sido bien definido o si es necesario desestimarlo por alguna razón. Por ejemplo, suelen ingresar requerimientos que han sido solucionados anteriormente, y por desconocimiento de consultores o del comité, se presentan como requerimientos nuevos, o también, suelen presentarse pedidos de solución de errores sobre funcionalidades que fueron definidos exactamente de la manera en que se encuentran funcionando en la actualidad.
No obstante, existen analistas que comienzan sus tareas sin realizar esta práctica dado que la forma actual de llevarla a cabo es engorrosa. Actualmente, no hay trazabilidad directa con los casos de uso ni documentación de los proyectos anteriores a SCRUM, por ello, localizarlos es una tarea que exige un esfuerzo extra.

Escrituras de historia no estandarizadas: De acuerdo al requerimiento, a la experiencia y conocimiento del analista funcional, las historias son detalladas en mayor o menor grado. En general, cuentan con bastantes aspectos funcionales, técnicos y de negocio, dado que responde a un requerimiento organizacional del Dueño de Producto. Las historias son consultadas, ante eventos de errores en las soluciones desplegadas, como evidencia de lo desarrollado. El consumo de tiempo para hacer historias con alto nivel de detalle es excesivo, incluso la documentación suele finalizarse con posterioridad al código y las pruebas. En algunas situaciones, el pedido asociado a la historia se homologa (cierra) aunque la documentación (historia y documentación asociada) no haya sido terminada. Lo correcto es que la historia se encuentre cerrada y toda la documentación asociada terminada.

Cuando el requerimiento es puntual y su especificación es relativamente breve, los analistas cargan defectos de mejora en lugar de historias de usuarios, simplemente porque el defecto involucra una descripción corta del problema y de la solución, sin mayores datos.

Para asegurar que las historias de usuarios incluyen todos los aspectos técnicos y funcionales requeridos por el cliente, y que alcanzan la estructura mínima definida para este componente, el Área de Calidad debe realizar una revisión antes del cierre

de la misma. La persona encargada de esta práctica es el Scrum Master, quien tiene a cargo ambos roles. Su escasa disponibilidad complica el cumplimiento de esta revisión.

No se indican los criterios de aceptación: estos deben ser acordados con el PO o los consultores. En la práctica, son definidos por el analista, y luego validados y confirmados por el PO. Esta tarea depende exclusivamente del analista, y general no se hace, primero porque existen analistas con poco conocimiento de negocio y/o de producto como para plantear los criterios, y segundo, porque al momento de validar con el PO la historia se enfoca en lo que hay que hacer y no en lo que se debe testear. Los analistas que indican criterios lo hacen de forma global con poca precisión, por lo cual, es factible que se cometan errores u omisiones.

Manual de configuración y reglas de negocio desactualizadas: al igual que las historias, estos documentos también son escritos según el criterio de cada analista. Actualmente se encuentran desactualizados o mal especificados, motivo por el cual, existen numerosos reclamos del área de atención al cliente. No hay una persona específica asignada para auditar estos documentos.

Dificultades con la disponibilidad del Product Owner: A medida que se avanza con la historia, los miembros del equipo deben contar con la colaboración del Dueño del Producto (PO) a los fines de elicitar y definir aspectos de negocio, definir cuestiones del desarrollo, obtener reglas de negocio, evacuar dudas y revisar los avances, pero dado que el PO cumple diferentes funciones dentro de la organización, suele ser complicado pactar una reunión con él. Esto provoca un retardo en las tareas del equipo.

Afinidad con la Herramienta de Gestión: Team Fundation Server (TFS) resulta una herramienta poco amigable para los analistas funcionales. Por ello, suelen crear las historias en un documento Word y luego las asocian a la herramienta, considerando a la herramienta como un repositorio y no facilitadora del desarrollo de las tareas.

4.10.2 Área de Testing:

Inclusión del tester desde el inicio: Como una buena práctica de la metodología se indica que todos los miembros del equipo deben participar de las reuniones de planificación, así todos tendrán el mismo conocimiento de la solución a desarrollar. No obstante, suele suceder que el tester no participa de dicha reunión dado que encuentra afectado a tareas de análisis y/o testing en otro equipo. Al momento de escribir y ejecutar las pruebas, éste tester se basa únicamente de lo indicado en la historia sin aportar un mayor valor agregado a su tarea o una mirada crítica a lo desarrollado.

Respaldo de pruebas: Según el proceso, los testers deben documentar los escenarios de prueba en la herramienta TFS. Algunos testers ejecutan pruebas sin escribir estos escenarios y sin dejar evidencia de la cobertura y los resultados. Ante eventuales defectos en los despliegues, el equipo de desarrollo no puede asegurar que la prueba se realizó de acuerdo a las expectativas de la historia, en qué base de

datos se ejecutó y pasó la prueba, que defectos se detectaron y cuáles se resolvieron.

Test unitarios insuficientes: En cuanto a los defectos, son cargados como tales y estimados en la herramienta TFS, a los fines de hacer un seguimiento de su avance. Durante la ejecución de las pruebas, los testers detectan un alto grado de defectos o fallas resultantes de una reducida prueba de test unitaria realizada por los programadores. Esta situación trae como consecuencia que exista una menor cobertura de pruebas funcionales, dado que gran parte del tiempo se dedicó a la realización de ciclos básicos de prueba.

Falta de revisiones: Los tipos de pruebas definidos son: Exploratorias, Operacionales, de Carga, de Estrés e Integrales. Se realizan una reducida cantidad de pruebas de Regresión (sólo en aplicaciones críticas o complejas). Existe un entorno exclusivo para pruebas (distinto al de desarrollo y de producción). Una vez realizado el reléase del sprint, se ejecuta una prueba a los fines de detectar problemas en el despliegue.
Dado que no hay una validación de los casos de prueba, suelen obviarse algunos de los tipos, que son esenciales para el tipo de funcionalidad que se ejecuta, trayendo como consecuencias, entre otros, que los procesos masivos bloqueen tablas de la base de datos y no permitan la operatividad normal del municipio, que las impresiones masivas se interrumpan por problemas de tiempo del servidor, que la comunicación entre las interfaces presenten error, inconsistencias de datos en funcionalidades relacionadas, etc.

Cobertura de prueba insuficiente: al finalizar la escritura de casos de prueba, el tester debe validar la cobertura, coherencia y prioridades de los escenarios con el PO. Esta práctica se ve obstaculizada por la escasa disponibilidad del PO.

4.10.3 Área de Programación:

Definición de estándares: no se implementan buenas prácticas de desarrollo de software que permitan un código estandarizado que otorgue una mayor facilidad de lectura y mantenimiento. Cada desarrollador escribe el código de acuerdo a su conocimiento y experiencia propia, lo que lleva a que existan diferentes formas de implementar un mismo procedimiento que puede estar presente en diferentes funcionalidades.

Dificultades de mantenimiento: al convivir tan diferentes modos de realizar un mismo proceso es difícil y lento realizar actualizaciones/mantenimiento del código por el tiempo que se demora en interpretarlo. Cuando la persona que va a realizar este mantenimiento está muy en desacuerdo con la forma en que el desarrollador anterior programó, suele decidir rehacerlo de cero a su forma, con los tiempos extras que esto conlleva. Además, se corre el riesgo de que se impacte en otras funcionalidades relacionadas, sobre las cuales el desarrollador no tiene un conocimiento tan preciso por la falta de documentación y comentarios en el código que faciliten su lectura e interpretación.
Es probable que en la cobertura del testing no se contemple este cambio y su impacto total, dado que no forma parte de la solicitud de requerimiento original,

generando un despliegue de software defectuoso. Es posible que esto retrase los tiempos de entrega del requerimiento.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Capítulo 5: Calidad y Mejoras al Proceso

Una de las respuestas para dinamizar los cambios necesarios es la calidad, entendida como cultura, metodología y herramienta de gestión, que introduce el concepto de mejora continua en cualquier organización y a todos los niveles de la misma, lo que significa que tiene la voluntad de llegar a todas las personas y a todos los procesos
Considerando que cuando se habla de Calidad, se pueden distinguir dos enfoques complementarios: el estructural, vinculado a las normas, procedimientos, etc…, para que las acciones se realicen correctamente y considerando el sistema de información necesario para las mismas; y el que se ha denominado “procesos de mejora continua de la calidad”, que necesita del esfuerzo constante de todos los agentes implicados para mejorar lo existente, con los cambios y adaptaciones que sean necesarios en cada circunstancia (Francisco Javier Gómez Piñeiro).

Teniendo en cuenta este marco se presenta este capítulo, en el cual se toman en consideración los conceptos de calidad y de mejora continua, y luego, se detallan las propuestas con las cuales se intenta aportar ideas para mejorar el trabajo del equipo de desarrollo.

5.1 Calidad

- La Real Academia Española define calidad como: “Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor”.

- La International Standards Organization, ISO en la norma 8402:1994, la define como la “Totalidad de propiedades y características de un producto, proceso o servicio que le confiere su aptitud para satisfacer unas necesidades expresadas o implícitas.” En la actualización de la Norma ISO, la 9000:2000, la definición se actualizó a “Grado en el que un conjunto de características inherentes cumple con los requisitos”.

- La IEEE, Std. 610-1990, la define como “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”. Pressman, Roger, Ingeniería de Software 3ª Ed., McGraw Hill, 1993

Teniendo en cuenta estas definiciones, podemos decir que un software es de calidad si cumple con un conjunto de cualidades como las siguientes:

- **Corrección:** grado en que el software cumple con su especificación y satisface los objetivos que propuso el cliente.

- **Confiabilidad:** grado en que el software desempeña su función con la precisión requerida.

- **Eficiencia:** cantidad de código y de recursos necesarios para que un software realice su función.

- **Integridad:** control sobre el acceso al software y a los datos por parte de las personas no autorizadas.

- **Facilidad de uso:** esfuerzo necesario para aprender, operar e interactuar con el software.

- **Facilidad de mantenimiento:** esfuerzo necesario para localizar y corregir errores en el sistema.

- **Flexibilidad:** respecto a la calidad de un programa, la flexibilidad hace referencia a establecer en qué medida el programa es susceptible de ser cambiado

- **Portabilidad:** esfuerzo necesario para transferir el programa de un entorno de hardware o software a otro.

- **Facilidad de reutilización de software:** uso repetido de producto, componentes, procesos, conocimientos, etc.

- **Interoperabilidad:** esfuerzo necesario para acoplar un sistema con otro.

La obtención de un software de calidad implica utilizar metodologías o procedimientos estándares para análisis, diseño, programación y prueba del software, que permitan enfocar la filosofía de trabajo elevando la productividad y calidad.

Para mejorar la calidad, las metodologías ágiles priorizan los objetivos del proyecto en función del valor que aportan al cliente.

5.2 Dimensiones de la calidad de software desde el punto de vista Ágil

1. **Cumplimiento de expectativas del Cliente (Product Owner)**.
   Implica que se deben satisfacer no solo las expectativas del cliente, sino que también se deben cumplir con las expectativas del consumidor o usuario final del producto.

2. **Calidad externa, funcionamiento correcto**. El producto debe comportarse de la manera esperada, tanto a nivel funcional como no funcional (rendimiento, usabilidad, etc.).

3. **Calidad interna**. El producto debe ser mantenible, debe poder evolucionar a un ritmo sostenido.

4. **Satisfacción del equipo**, esté debe estar orgulloso de su trabajo para mantener su motivación y compromiso con el proyecto y la empresa.

5.3 Mejora continua

La mejora continua del proceso es un factor fundamental de cualquier equipo de trabajo ágil. Esto se conoce como "kai-zen" del japonés: cambio bueno.
Para aumentar progresivamente la productividad y elevar la calidad de software, se deben identificar y eliminar sistemáticamente los impedimentos. Entendiendo como impedimento a cualquier situación que disminuye la productividad y/o atente contra la calidad.

Para poder aplicar **Kaizen** hay que pasar por cuatro pasos definidos por Eduards Deming en su teoría del Circulo **PDCA** (Plan, Do, Act, Check), o **circulo de Deming**:

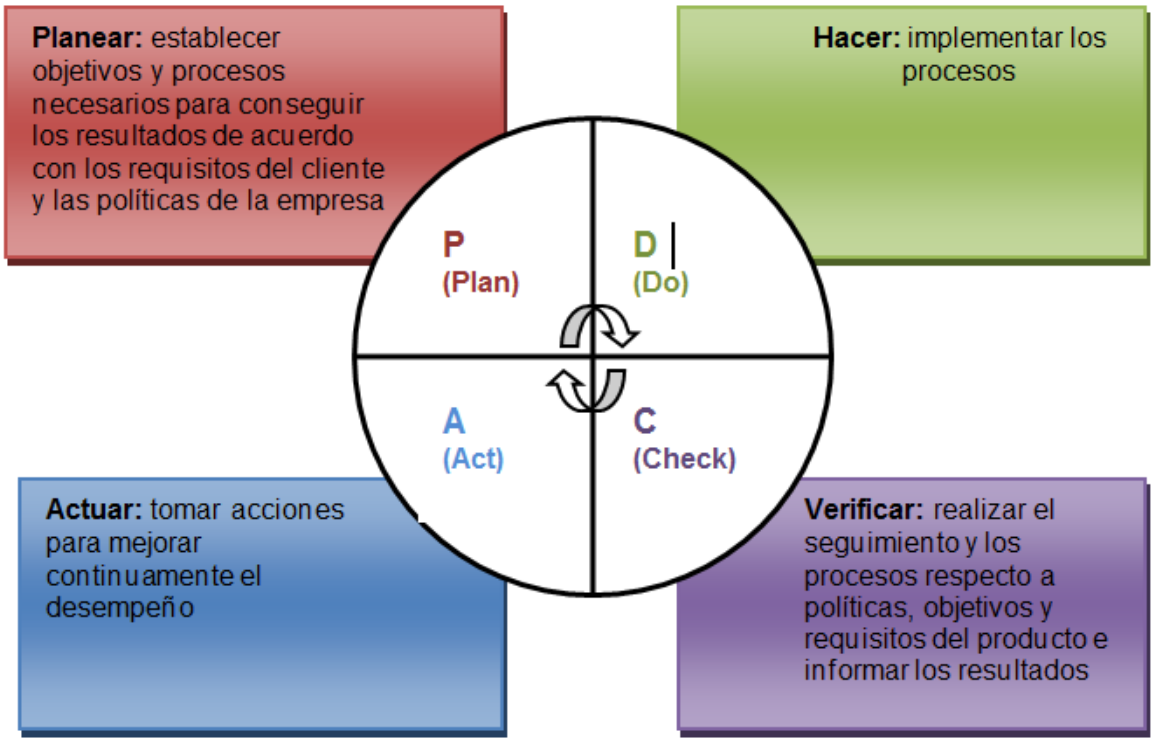

Figura 83: Círculo PDCA (Fuente: http://www.pdcahome.com/5202/ciclo-pdca/)

La utilización del círculo y su repetición continua e iterativa de los pasos mencionados, es propio del tipo de mejora que nunca alcanza su fin sino que siempre encuentra nuevas oportunidades de mejora.

5.4 Objetivos de la mejora continua

- Incrementar la ventaja competitiva a través de la mejora de las capacidades.
- Alinear las actividades de mejora de todos los niveles con la estrategia organizativa propuesta.
- Lograr flexibilidad para reaccionar rápidamente a las oportunidades que se presenten.

5.5 Procesos y mejora continua

Un proceso es un conjunto de actividades o tareas que se relacionan entre sí, y que se ejecutan siguiendo un orden lógico con el propósito de alcanzar un resultado específico a partir de las entradas de recursos e información. Cuando los procesos no están bien diseñados se transforman en un problema dentro de las organizaciones, impidiendo el incremento de la productividad y la calidad, de ahí la necesidad de trazar estrategias para mejorarlos.

Ante esto, las organizaciones de software han enfocado los esfuerzos a mejorar sus procesos para lograr productos de mayor calidad y ofrecer mejores servicios que satisfagan las necesidades de clientes cada vez más exigentes. La mejora del Proceso Software implica la mejora continua de la calidad y puede tener como metas elevar la capacidad del equipo, auditar desarrollos de software interno, planificar estrategias de ingeniería del software de la empresa, etc.
Al iniciar las tareas de mejora del Proceso de Desarrollo de Software, es necesario tener en cuenta los siguientes objetivos principales de la mejora de procesos: Comprender el estado actual de las prácticas de gestión y de ingeniería de software en la empresa. Seleccionar las áreas de mejora donde los cambios puedan producir los máximos a medio y largo plazo. Centrarse en añadir valor al negocio y no en alcanzar un proceso utópico. Combinar procesos eficaces con personas con habilidades, motivadas y creativas.

La Mejora del Proceso consiste en aplicar prácticas que proporcionan buenos resultados y cambiar o eliminar las prácticas que causan problemas. El valor que tiene la Mejora del Proceso se ve reflejado en reducciones de costos de producción, mejoras en la calidad del producto y cumplimiento de las necesidades y expectativas del cliente. Además de estos beneficios, se logra un entorno de trabajo más estable, una reducción de la tasa de rotación del personal, mayor compromiso de los equipos y una mejora en las relaciones de trabajo con los clientes.

5.6 SCRUM y mejora continua

De acuerdo a las características de SCRUM mencionadas en los capítulos anteriores, recordamos:

- Ciclos de duración fija de no más de 4 semanas
- Entregas de incrementos del producto final
- Reuniones diarias
- Retrospectivas, etc.

Se puede decir que SCRUM ejecuta el ciclo de mejora continua de Deming en los siguientes aspectos:

- Teniendo en cuenta las **expectativas del cliente**, las cuales se van depurando iteración a iteración mediante las siguientes actividades:

- **Reunión de planificación**: al inicio de la iteración el equipo se reúne con el cliente, quien ha expresado sus necesidades y expectativas mediante la lista priorizada de objetivos del proyecto (product backlog). El equipo selecciona tantos objetivos como crea capaz de cumplir en la iteración. Los refina con preguntas al cliente y hace una primera descomposición en forma de tareas necesarias para completar cada uno de estos objetivos, creando la definición de completado y la lista de tareas de la iteración (sprint backlog). Es importante que el cliente prepare iteraciones con metas claras y objetivos cohesionados que produzcan mayor sinergia.

- La **iteración se desarrolla orientada a objetivos**, minimizando el "número de objetivos en curso" (WIP, Work In Progress). Es deseable tener el objetivo completado cuando acabe la iteración y no muchos iniciados sin completar.

- **Presentación de los resultados** completados al cliente al final de la iteración. El feedback recibido del cliente permite ir acercándose a sus expectativas mediante ajustes que podrán realizarse en siguientes iteraciones. Este refinamiento se incorporará al product backlog y el cliente lo repriorizará.

Además de este ciclo de calidad, encontramos dentro de la propia iteración otro ciclo de Deming: la reunión diaria de sincronización del equipo, permite inspeccionar lo que está sucediendo y realizar las adaptaciones necesarias para poder conseguir los objetivos de la iteración, añadiendo o quitando tareas del sprint backlog durante la iteración.
Se puede observar que uno de los principales beneficios de SCRUM es la flexibilidad frente a cambios en el contexto, dado que permite modificar y repriorizar objetivos y/o tareas del proyecto iteración a iteración para poder conseguir un objetivo.

- Mejorando los **procesos de trabajo del equipo**, mediante:

  - La replanificación de las tareas a ejecutar.
  - La retrospectiva sobre la ejecución de la iteración, donde se detallan errores, impedimentos y aprendizaje de cada iteración para conseguir que los errores se transformen en aprendizaje, garantizando efectos a corto plazo. Esto permite interiorizar lo aprendido y ponerlo en práctica en un plazo de tiempo muy breve.

- Buscando la **motivación y satisfacción del equipo**:

  - Pudiendo entregar al cliente software funcionando con la mejor calidad y en forma regular.
  - Dándole la posibilidad de conocer y entender muy bien las expectativas del cliente, lo que impactará en mejores resultados en las siguientes iteraciones.
  - A través de la mejora continua del proceso entre el equipo y el cliente se permite ser más productivo, reducir retrabajo y tareas de poco valor.

5.7 Propuestas de mejoramiento para GobFactory

En todo proyecto de software existe la necesidad de una correcta gestión, para lo cual, al definir estas propuestas de mejora, se tomaron como referencia las 4P del desarrollo de software. Las mismas, hacen referencia a personal capacitado y motivado, a la selección de un proceso adecuado y una gestión acorde al proyecto enfocado en obtener un producto de calidad.

5.7.1 - Personas

Los recursos humanos siempre son lo más importante en el desarrollo de software. Para que una empresa de este tipo alcance el éxito, no debe valorar principalmente las herramientas que utiliza, sino, considerar en primer lugar, a las personas y su rol dentro del equipo. El reclutamiento y la selección de personas son fundamentales, ya que es la oportunidad para observar las capacidades que los postulantes tienen para aportar a la organización y detectar quienes tienen capacidad de trabajar en equipo y/o bajo presión.
Es importante observar el desempeño del equipo continuamente para mejorar la toma de decisiones. Una forma de medir este desempeño, es a través de la participación activa de las personas en todas las actividades incluidas en el proyecto, analizar su productividad y su interacción con el resto del equipo. Esto ayudará a percibir si realmente el equipo cuenta con la suficiente capacidad y responsabilidad para llevar adelante sus tareas, conocer el nivel de motivación y detectar la necesidad de estímulos, capacitación, etc. para mejorar el rendimiento.
Teniendo en cuenta las características de GobFactory, su problemática y las opiniones relevadas durante el trabajo de campo se sugieren las siguientes mejoras:

- Redefinir tareas dentro del equipo en base a la experiencia, capacidad e interés de cada persona.

- Concientizar en la necesidad de que los equipos sean estables (si el equipo tuvo éxito, que se mantenga en el tiempo).

- Refinar el proceso de reclutamiento y selección de personal incorporando etapas de presentación e intercambio de los postulantes preseleccionados con el equipo de desarrollo, a los fines de poder contar con la opinión del grupo.

- Definir equipos multifuncionales más pequeños con capacidades y características diferentes y elegir a los líderes de entre los miembros del grupo. Esto permite establecer una estrategia para afrontar urgencias que requieran atención inmediata con un equipo altamente capacitado.

- Motivar a la autogestión y autoorganización

Impacto: equipos más potentes.

- Incentivar al trabajo en equipo, la comunicación interna y la motivación del grupo a través de talleres dinámicos, en los que se trabaje sobre estos puntos y que estén a cargo de personas con experiencia en este tipo de capacitación.

- Desarrollar el conocimiento, la experiencia y habilidades de las personas como parte de las actividades generales de gestión de la calidad de la organización.

Impacto: promover, motivar y mejorar el trabajo del equipo y el desempeño individual para afrontar los diferentes tipos de situaciones que se presenten. Promover y generar capacitaciones internas o externas. Generar un clima apropiado para la comunicación, el intercambio de ideas y el debate, intentando alcanzar soluciones consensuadas, más apropiadas técnicamente y que agreguen valor al producto.

5.7.2 - Proyecto

Gestionar un proyecto exitoso consiste en entender la problemática y dar una solución adecuada. Un proyecto es el elemento organizativo por el cual se gestionará el desarrollo de las actividades en la implementación del producto. Para la correcta implementación de un proyecto tecnológico se debe entender la importancia de los siguientes aspectos: recursos, calidad, finanzas, contratos, comunicaciones, riesgo, alcance y planificación.

Algunos de los factores que conducen al éxito:

- Conformar el equipo correcto
- Incentivos ⟺ Calidad
- Seguimiento del procesos ⟺ Controlar actividades
- Decisiones Inteligentes
- Análisis de resultados

A continuación se indican propuestas que intentan aportar y mejorar la gestión del proyecto:

- Incorporar información de utilidad en el tablero para ayudar al equipo en su tarea de autogestión y autoorganización. Se podrían considerar los siguientes datos:

  - **Capacidad de cada persona**: ayuda a conocer la disponibilidad diaria, en horas, en el proyecto. Es la base para estimar la fecha de finalización de la iteración. Los equipos conocen con exactitud el tiempo en el que debería avanzar en cada proyecto.

  - **Detalle de horas por día**: en recuadro se indica la cantidad de horas estimadas total para la iteración (por cada uno de los miembros), el

avance diario y las horas restantes de la iteración. El total de horas pendientes se reflejan en el gráfico Burndown Chart.

- **Reestimación en el post it**: diariamente cada miembro debe actualizar en el post it el pendiente por tareas o su reestimación en caso de haberse efectuado.

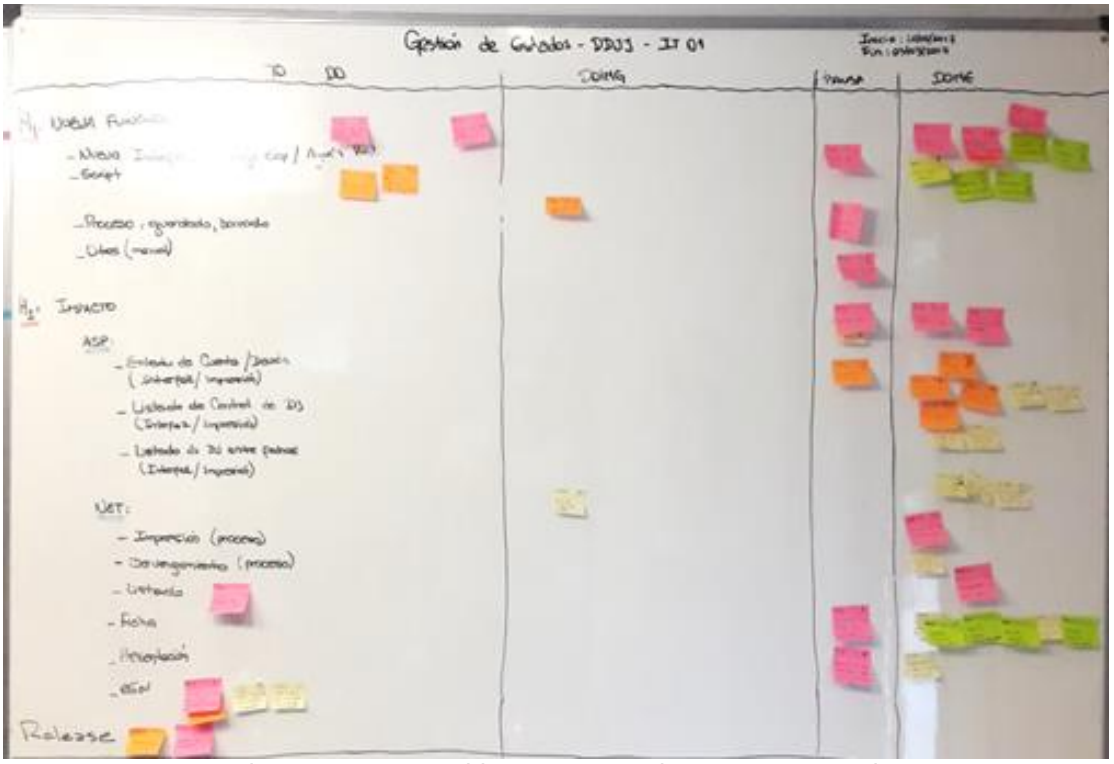

Figura 84 – Foto: tableros SCRUM de un proyecto real

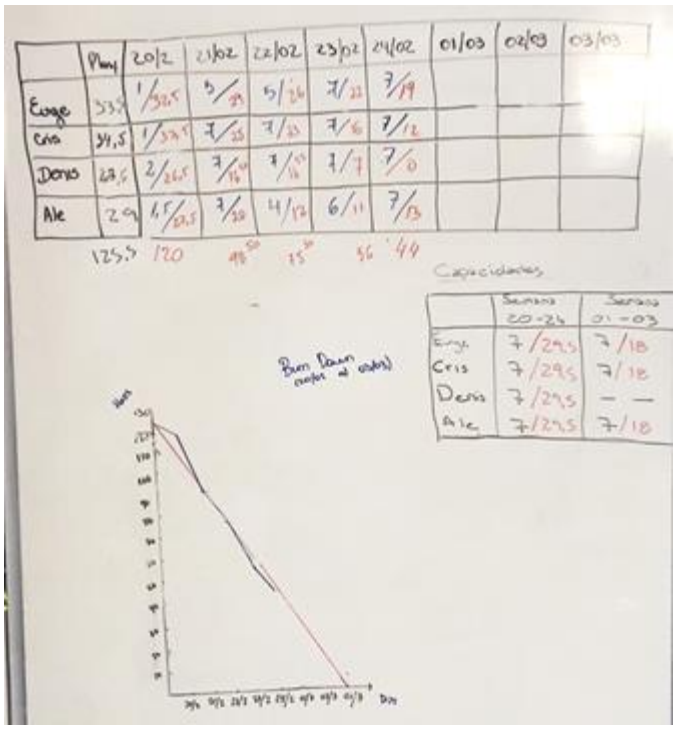

Figura 85 – Foto: prueba y validación con el equipo de desarrollo de la propuesta

Figura 86 – Foto: propuesta de confección del post it de tarea

Impacto: información más detallada sobre el avance de la iteración y la disponibilidad de cada miembro del equipo.

- Definir al Scrum Master como único receptor de requerimientos no planificados provenientes de atención al cliente (solicitudes recibidas a través de llamadas telefónicas, tareas de soporte de atención o técnica, capacitaciones puntuales, etc.), para evitar que el equipo de desarrollo se desvíe de su tarea planificada sin autorización y sin que quede registro de lo realizado.

  Por cada interrupción concretada, el Scrum Master debe pedir una solicitud de requerimiento no planificado permitiendo de este modo incluirlo en las métricas (Trabajo planificado vs no planificado) al final del sprint y poder medir el impacto de estos pedidos en la productividad.

Impacto: gestionar las interrupciones al equipo de desarrollo. Si el Scrum Master conoce el origen, frecuencia y alcance de cada solicitud podrá realizar una mejor reasignación de tiempos y recursos.

5.7.3 - Producto

Todo producto requiere una estimación cuantitativa y una adecuada planificación. Un correcto relevamiento de información y un análisis detallado de requerimientos, proporciona los datos necesarios para realizar una estimación, no sólo del costo del producto, sino también del esfuerzo que será requerido.
Aspectos que tienen ver directamente con el producto son:

- El software que se desarrolla durante la vida del proyecto
- Modelos, códigos, ejecutables, documentación, diagramas
- UML, bocetos de la interfaz de usuario, prototipos, componentes, planes de prueba
- Ingeniería y gestión
- Colección de modelos
- Historias de usuarios, análisis, diseño, despliegue, implementación y prueba.
- Reuniones del equipo de desarrollo con el cliente las veces que sea necesaria para definir el dominio y los objetivos del producto.

A partir de estos aspectos, se incluyen en este apartado las siguientes propuestas de mejora:

- **Reemplazar la etapa de validación de los criterios de aceptación**, ya que no se cumple, **por una etapa de Revisión (Review)** en la que se contraste la historia de usuarios y todos sus componentes (datos técnicos, alcance, reglas de negocio, criterios de aceptación, prototipos, ejemplos, etc.) con el cliente.

Impacto: esto permitiría asegurar que el requerimiento fue correctamente interpretado y por ende, reduciría los defectos relacionados con el análisis. El cliente, alcanzaría un grado de conocimiento mayor sobre lo solicitado y lo que será desarrollado. Aumentaría su compromiso y participación al interactuar fluidamente con el equipo, y éste a su vez, tendría la tranquilidad de trabajar sobre información certera.

- **Establecer estándares para la escritura de historias.** Deben ser breves, claras y contener número de pedido, descripción general (siguiendo el esquema como – quiero - para), descripción específica, datos técnicos, criterios de aceptación especificados al mínimo detalle por el cliente interno y discusiones (validaciones). Las discusiones deben contener fecha, hora, nombre de los participantes en la reunión y temas tratados y conclusiones.

  La historia debe ser escrita focalizada en el requerimiento y pensando en la posibilidad de reutilizarlas en nuevas historias o defectos de mejora.
  Actualmente una historia describe una funcionalidad completa, similar a un caso de uso. Son complejas, dado que abarcan todas las características y especificaciones funcionales y técnicas de la solución. Deberían subdividirse en historias diferentes y reutilizables como por ejemplo encabezado, cuerpo, proceso de validación (si fuera necesario) y reporte, de este modo cada una de ellas podría ser instanciada desde nuevos desarrollos sin necesidad de reescribir algo que ya existe.

Impacto: una correcta escritura de historia facilitará su lectura, comprensión y su reutilización, reduciendo los tiempos del analista y favoreciendo la consistencia y coherencia de las funcionalidades. Este cambio, sumado a la revisión constante con el cliente, permitiría liberar al Scrum Master de tener que controlar la calidad de la historia.

- **Establecer estándares para la escritura de defectos.** Deberían ser formulados de manera breve indicando tipo de defecto (funcional, no funcional o de interfaz), categoría (invalidante, severo, leve o mejora), número de pedido asociado, funcionalidad, descripción general, resultado esperado y cliente externo donde se detectó el defecto.

  Una vez finalizada la corrección o modificación solicitada, el programador tendría que escribir cuál fue la solución realizada y el tester referenciar la prueba realizada (adjuntando evidencia en caso de corresponder).

  Debería ser más ágil que la historia sin entrar en especificaciones técnicas o de negocio.

  Impacto: permitiría contar con evidencia de modificaciones puntuales, errores o fallas presentadas y solucionadas. Agilizará el tiempo del equipo al momento abordar un defecto, dado que involucra únicamente una tarea de verificación, escritura del defecto, programación y una prueba rápida.

- **Establecer estándares para la especificación de parámetros y permisos especiales en Manual de Configuración.** Por cada parámetro impactado en la solución que se esté desarrollando, se debería registrar una descripción breve del funcionamiento y las configuraciones posibles. La descripción deberá indicar la/s funcionalidad/es que lo utilizan, el comportamiento de la misma (con y sin el parámetro) y las variantes del mismo.

  El manual debería estar ubicado en el repositorio del producto y distribuido de acuerdo a la estructura de módulos. En cada documento, se debe respetar la distribución de submódulos, temas y subtemas asociado a la solicitud de requerimiento que dio origen al parámetro o permiso. Esta ubicación se deberá representar por un código de tema compuesto por cuatro niveles, el primer nivel indicará módulo, el segundo submódulo, el tercero tema y el cuarto subtema.

  Ejemplo:
  002.000.000.000 Rentas
  002.015.000.000 Financiación
  002.015.005.000 Carga Tipo de Talonario

  Impacto: facilitaría la trazabilidad y comprensión de cada parámetro registrado. Este manual sería de mucha utilidad para el consultor a la hora de implementar el producto en los municipios.
  Tener los parámetros centrados en un documento fuera de la historia de usuario, permitirá tener una historia más limpia, enfocada en el requerimiento y no en la configuración.

- **Establecer procedimientos de trabajo según el tipo de requerimiento ingresado.** Cuando ingresa un requerimiento al área, el analista que lo toma, comienza su desarrollo de acuerdo a su propio criterio y experiencia, no sigue

reglas ni procedimientos previamente definidos. Esto puede ocasionar análisis deficientes o pérdidas de tiempo y de esfuerzo sobre todo cuando son abordados por personal sin experiencia o conocimiento en el tema o rol. Por ejemplo, cuando un desarrollador debe ejercer tareas de analista, éste comienza primero por el código y luego con la escritura de la historia, avanzando y retrocediendo de acuerdo a las dudas que le van surgiendo sobre la marcha.

Se sugiere la definición de un marco de trabajo que permita a todos por igual conocer el modo de llevar adelante el análisis de un requerimiento. En este marco se debería tener en cuenta:

Dependiendo del tipo de requerimiento (nuevo, modificación o error) el analista debería recorrer una serie de pasos diferentes (Anexo 1).

- **Implementar el testing durante todo el proceso, no al final**: en esta propuesta los tres roles del equipo (analista funcional, programador y tester) trabajarían en simultáneo para acordar criterios y coberturas desde el momento en que se comienza el análisis.

  Los tres roles aportarán aspectos importantes tanto para el análisis como para el testing y la programación. Los casos de prueba que se ejecutarán en el testing se plantearán conjuntamente con análisis. Los escenarios más críticos servirán como base para el test unitario realizado por el programador. Este último tendrá la colaboración del analista y del tester.

  Impacto: beneficiaría en que todos aportarían aspectos en cuanto a lo técnico y a lo funcional propios de su rol, mejorando la calidad de la solución desde el inicio y no esperando la instancia de testing a posteriori para detectar que la solución en desarrollo se encuentra incompleta o con errores básicos.
  La participación del tester y del analista en el test unitario aseguraría un software funcionando (sin errores de comunicación o de interpretación) y de acuerdo a las especificaciones, dando lugar a que el tester se enfoque en las pruebas funcionales y de negocio sin errores técnicos. El tester conocería en detalle la funcionalidad, lo que le da valor agregado a la calidad de su trabajo.

- **Establecer pautas para testing, que tiendan a eliminar los defectos en el cliente final.** Actualmente, tester encara la escritura de prueba en su propio criterio y experiencia, no sigue reglas ni procedimientos previamente definidos. Ante las dificultades observadas se proponen las siguientes estrategias que podrían impactar positivamente en el desempeño del grupo:

  - Se sugiere que el equipo de testing también participe en las acciones sugeridas anteriormente (etapa de revisión, reunión de presentación de requerimientos y reunión de presentación de la solución). De esta manera, se refinarían los requerimientos, se revisarían aspectos técnicos y limitaciones, se reafirmarían aspectos de negocio y funcionales. La participación de todos los miembros del equipo nivelaría el conocimiento

del equipo, ampliando los límites de la visión antigua y acotada del tester. La reunión de presentación de la solución es la última instancia en la que el equipo se asegura que lo desarrollado cumple con los esperado por el consultor (cliente interno).

- Desde el área propia de testing se sugiere elaborar un checklist con aspectos mínimos que debe contemplar el tester a la hora de plantear los casos de prueba, la cobertura y las técnicas a utilizar según el tipo de pedido que se encuentra en desarrollo. Estos ítems podrían ser sometidos a revisión y actualización en reuniones preestablecidas entre los testers a los fines de adecuarla y mejorarla.

- Otra opción, sería generar un documento que contenga procedimientos a seguir, técnicas de testing apropiadas, entornos propicios y sugerencias sobre cobertura de la prueba, según diferentes tipos de pedidos. Ayudaría a diseñar pruebas que sean más eficientes para detectar errores con la menor cantidad de esfuerzo y tiempo posibles.

Impacto: un marco de trabajo ordenado, participativo, consensuado y validado, otorgaría seguridad al tester a la hora de abordar un proceso, lograría coberturas mejor definidas, escenarios y técnicas de testing adecuados y centraría prácticas apropiadas según cada tipo de requerimiento (tratando siempre de detectar la totalidad de errores existentes). Estas mejoras se verían traducidas en un aumento de la calidad del producto entregado.

- **Incorporar el versionado de componentes a la plantilla de prueba como un elemento obligatorio.** Esto significa que al momento de ejecutar un ciclo de prueba, el tester obtenga a través de la herramienta TFS la versión lógica del componente que se está verificando y debería dejarlo indicado en el caso de prueba.

Impacto: otorga al tester la posibilidad de justificar un ciclo indicado como correctamente ejecutado (ESTADO = PASÓ) ante la aparición de un defecto en el cliente final.

- **Diseñar un checklist para escritura y ejecución de pruebas.** Los analistas pueden omitir especificar o tener en cuenta aspectos importantes a la hora de crear los escenarios de prueba.

Impacto: ayuda a recordar al tester aquellos ítems que no debe descuidar en ninguna instancia de prueba. Le servirá de guía antes, durante y al finalizarlas. Así podrá revisar que la cobertura contemple todos los aspectos previstos.

- **Incorporar como práctica la realización de pruebas de regresión.** Ante desarrollos que impacten en la misma o en otras funcionalidades o procesos, se deberían llevar a cabo actividades de testing de regresión a los fines de

asegurar que no se haya visto afectado el normal funcionamiento de dichos componentes. Deberían ejecutarse los escenarios de prueba utilizados en desarrollos anteriores.

Impacto: permite asegurar que posteriormente a algún cambio, lo que estaba funcionando correctamente, continúa de la misma manera.

- **Establecer un equipo de desarrollo destinado a investigar, proponer y proyectar la automatización de pruebas unitarias.** El producto ofrecido por la empresa es complejo, por ello, al incorporar nuevas funcionalidades o modificar/corregir las existentes, el programador puede, debido al desconocimiento sobre el sistema total, introducir cambios que afecten involuntariamente otras funcionalidades. Esto termina impactando en el trabajo del tester, ya que lo sobrecarga de tareas al tener que ejecutar más ciclos para depurar el test unitario. La automatización de las pruebas básicas de test unitario ayudaría a asegurar la calidad del código entregable permitiendo al tester enfocarse en las pruebas de funcionales.

  Para comenzar con la automatización, este equipo podría generar scripts que testeen funcionalidades que ya se encuentran estables y que no deben ser impactados con cambios en otra parte del software. También se podrían crear scripts que permitan testear automáticamente cuestiones comunes a todas las páginas como por ejemplo: conectividad, autenticación y transferencia de datos.

  Impacto: asegurar la calidad del código entregable y reducir el tiempo de pruebas dado que los ciclos no fallan por cuestiones técnicas básicas y evitables de modo sencillo.

- **Establecer trabajo colaborativo entre los programadores para realizar el test unitario**. Deberían recibir apoyo del tester y del analista funcional para definir ciclos de pruebas más efectivos.

  Impacto: minimizar los defectos en el código entregado al test.

- **Considerar tips que aporten al trabajo de los programadores**.

  - Analizar claramente la historia de usuario o el defecto solicitado y el negocio. En caso de ser necesario, solicitar información adicional al analista y/o líder técnico.

  - Trabajar conjuntamente con el analista funcional y demás programadores que participan en el proyecto para tomar la mejor decisión de cómo diseñar y cuál debería ser el comportamiento de la solución.

  - Comentar el código que puede necesitar explicación. Cada función debería ser precedida por 1 o 2 líneas que describan los argumentos y que es lo

que la misma devuelve. Los comentarios deberían explicar el ¿Por qué? y el ¿Qué? (deben actualizarse cuando se actualiza el código).

- Los programadores y líder técnico deben acordar convenciones de nomenclatura para las variables y bases de datos.

- Organizar el código utilizando estructuras visuales que ayuden a entender la estructura, lógica y ciclos del código.

Impacto: estos tips lograrían un código más fácil de entender, depurar y mantener, permitiendo un seguimiento más sencillo de las variables, de las funciones y del flujo del programa.

- **Incorporar la programación de a pares** cuando se trata de desarrollos nuevos, de problemas complejos, de gran impacto, que incluyan tecnologías nuevas no probadas anteriormente o cuando se exija rapidez en la solución.

Impacto: esta metodología de trabajo aporta las siguientes ventajas:

Calidad de diseño: se logran en general programas más cortos, mejores diseños y pocos defectos. Al trabajar en parejas se consideran más alternativas de diseño que al trabajar individualmente. Se obtienen diseños más simples, más fáciles de mantener y se detectan rápidamente los defectos, lo que impacta reduciendo el costo de desarrollo.

Aprendizaje y formación continua: se produce un intercambio de conocimiento entre programadores.

Permite superar problemas complejos: el trabajo en pares ayuda a que problemas “imposibles” se convierten en más sencillos o incluso se resuelvan más rápidamente que cuando se trabaja por separado.

Aumenta la motivación: los programadores sienten mayor confianza y seguridad en que su trabajo será correcto.

Disminuye el riesgo ante el eventual abandono de un miembro del equipo ya que el conocimiento del sistema se comparte entre varios programadores.

Incremento de la disciplina y mejor gestión del tiempo: se logra mayor concentración y responsabilidad evitándose distracciones en cuestiones personales cuando trabajan con un compañero. También es flexible desde el punto de vista que permite que ante interrupciones, uno se ocupe de la misma y el otro puede seguir trabajando.

Menos interrupciones: las personas son más reacias a interrumpir a una pareja que a una persona que trabaja sola.

- **Tomar como criterio obligatorio el agregar funciones que validen todos los campos** indicados en la historia de usuario para asegurar que los datos que se ingresen a las páginas son del tipo y formato adecuado.

Impacto: evitaría que fallen los ciclos de prueba por aspectos técnicos básicos y evitables desde el punto de vista del software.

- **Realizar acciones de auditoria de código** (instrumentos y plan de auditoria) utilizando como referencia alguna metodología o guía estandarizada para que sea más efectiva.

  - En esta auditoria deberían participar todos los programadores y el líder técnico. Ellos conocen la estructura del software, el contexto de la aplicación y por lo tanto pueden indicar los componentes a tener en cuenta en la revisión de código optimizando la cobertura de la auditoria.
  - Se sugiere realizar estas auditorías al menos una vez por mes. Luego el auditor asignado deberá, a través de una reunión, informar los resultados de la misma al resto del equipo para que puedan trabajar en las correcciones. Este informe debe quedar en el repositorio del producto.

  - Al final de cada reunión se determinaran las funcionalidades (únicamente en tecnología .Net) que fueron desplegadas en el último mes y que requieren ser auditados. Se consideran prioritarias aquellas que pertenecen a nuevos desarrollos o que son de alto impacto. Las funcionalidades ASP no deben ser tenidas en cuenta dado que no habrá nuevos desarrollos en esta tecnología.

  - La auditoría debería cubrir al menos los siguientes aspectos:

    Autenticación: auditar que todas las conexiones internas y externas pasen a través del sistema de autenticación y asegurar que éstos controles no sean vulnerados; todas las páginas del sistema deben requerir autenticación (controlar que en los puntos donde se transmiten credenciales o información sensible se emplean POST, que las credenciales se transmitan cifradas, etc.).

    Autorización: validar que se hayan implementado los mecanismos adecuados de autorización y que se hayan definido los tipos o perfiles de grupos y los permisos de dichos usuarios a cada parte del sistema.

    Validación de Datos de Entrada: verificar que incluyan un mecanismo robusto que compruebe que los datos no puedan ser accedidos y/o modificados por un usuario malicioso (contemplar cabeceras HTTP, campos de entrada, campos ocultos, datos de listas, cabeceras/datos HTTP, que todas las comprobaciones de validación de datos en el servidor y no en el lado del cliente, etc.)

    Gestión de Errores/Fugas de Información: revisar que todos los métodos y funciones que devuelven valores tengan una correcta

gestión de errores, y devuelven valores comprobados y esperados cuando se produce una falla. Gestionar excepciones y situaciones de error, controlar los mensajes que se entregan al usuario, asegurar que la aplicación falla de “forma segura”, etc.

Registro/Auditoría: controlar que ningún tipo de información sensible se almacena en los registros de la aplicación: cookies, información en métodos “GET”, credenciales de autenticación, etc. La aplicación debe registrar las acciones que se producen por parte de los usuarios y especialmente en casos de acciones potencialmente peligrosas, todos los eventos de autenticación, fallidos o no, se registran, etc.

Criptografía: revisar que no se transmite ningún tipo de datos sensible sin cifrar, sea internamente o externamente y que los algoritmos o métodos de cifrado son reconocidos y poseen la robustez necesaria.

Entorno Seguro de Código: asegurar que ningún recurso (ficheros, directorios y su estructura)pueda ser accesible por el usuario, que las peticiones SQL dinámicas no sean vulnerables a ataques de Inyección SQL, asegurar que todas las decisiones lógicas por defecto tienen una cláusula preestablecida, etc.

Gestión de Sesiones: verificar cómo y cuándo se crea la sesión para el usuario, si debe estar autenticado o no, la robustez del ID de sesión, cómo se almacenan las sesiones, el tratamiento de la caducidad de las sesiones, cómo se gestiona la inactividad HTTP, cómo se gestionan las funciones de logout y su efecto en la gestión de sesiones, etc.

Impacto: ayudar al área de desarrollo a conocer el nivel de seguridad y la vulnerabilidad de las aplicaciones utilizadas en su producto, a conocer el nivel de seguridad del código y del software de base. Permite mejorar la seguridad del sistema y del área de desarrollo.

5.7.4 - Proceso

Proporciona un marco de trabajo desde el cual se establece un plan detallado para la construcción del producto. El equipo de desarrollo debe elegir el proceso adecuado que satisfaga las necesidades o requerimientos del cliente.

Se sugieren a continuación aspectos que tienen que ver con redefinir el ciclo de vida de desarrollo para adecuarlo y optimizarlo según las necesidades actuales de la empresa y adecuaciones que tienen que ver con la metodología:

- **Definir iteraciones de duración estable de dos semanas**, a los fines de cumplir con tres objetivos:
  - Facilitar la toma de métricas
  - Minimizar el tiempo desperdiciado y los costos asociados.

- Estresar el proceso en desarrollo para detectar sus debilidades y fallas.

Impacto: permite la rápida detección de errores de proceso y su pronta resolución.

- **Eliminar la etapa validación del cumplimiento de los criterios de aceptación,** tarea que actualmente realiza el tester junto al cliente interno.
  Se considera una etapa obsoleta ya que en la práctica no se está llevando a cabo.
  Si se implementa la etapa de revisión (mencionada en el punto anterior), el trabajo conjunto de los tres roles del equipo aseguraría una buena calidad de los requerimientos desde el inicio. Los mismos debieran llegar suficientemente depurados y detallados a la etapa de prueba, y con tantas revisiones como haya sido necesario durante la etapa de análisis.

  El tester, por su parte, conoce bien los aspectos técnicos y funcionales del negocio. Se asume que han sido tenidos en cuenta todos los escenarios de prueba posibles, ya que ha trabajado conjuntamente con el analista y el programador en la especificación de requerimientos.

Impacto: Depurar el proceso, eliminando etapas que no se efectúan y no son acordes a las necesidades del equipo. Se gana tiempo para la ejecución de pruebas.

Al aplicar esta modificación, la estructura del proceso quedaría conformada de la siguiente manera:

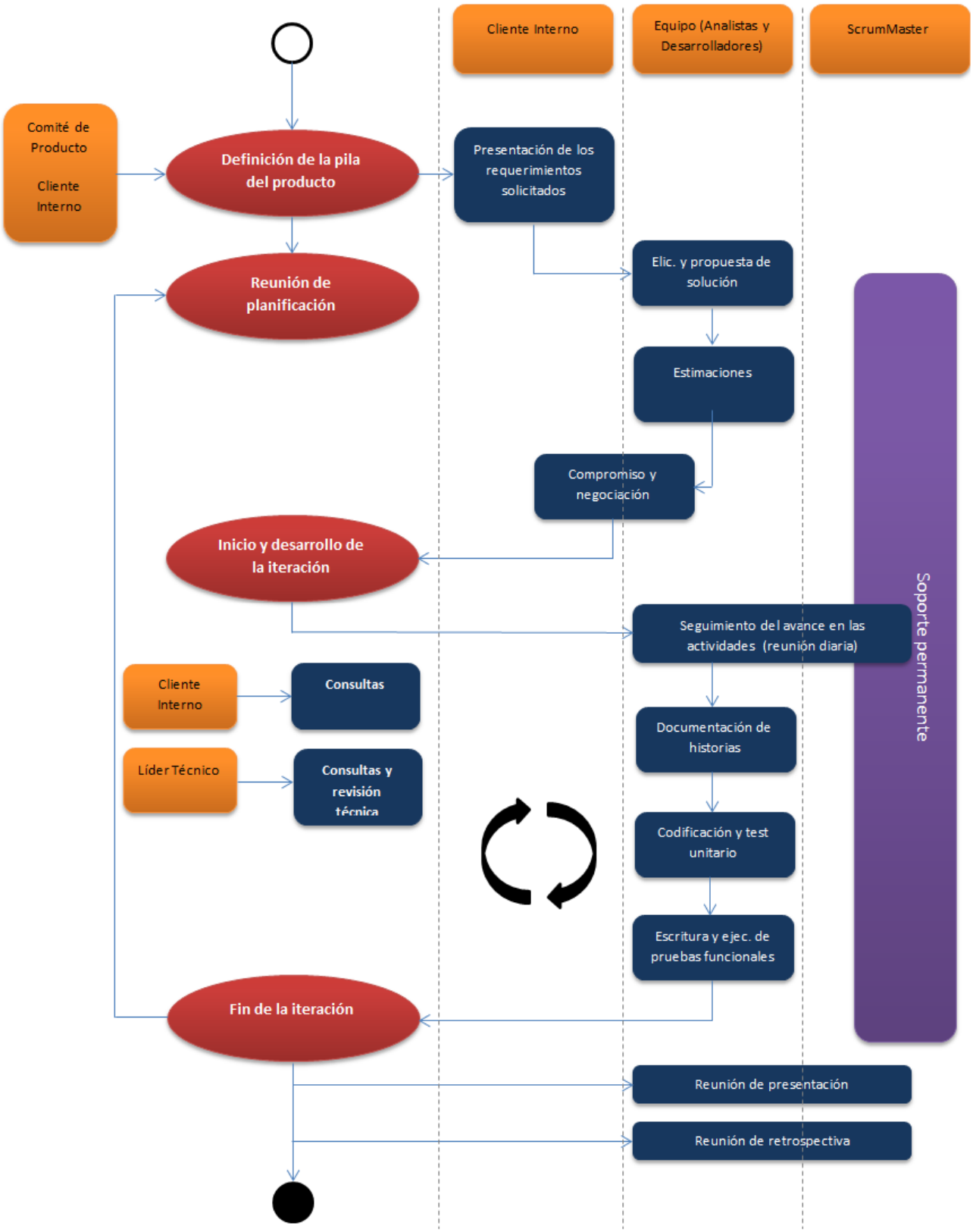

Figura 87: Proceso de desarrollo propuesto para el producto WGob (Fuente: Confeccionado por el grupo)

- **Optimizar la planning del sprint**:

  - Proponer que todos los roles del equipo participen de la reunión. De esta forma se logra que todos alcancen el mismo nivel de conocimiento.

  - El dueño del producto debe tener un conocimiento claro de los requerimientos que plantea. Su participación debe lograr unificar criterios.

  - Destinar tiempo suficiente que permita asegurar que el requerimiento fue comprendido y puede ser estimado. Si bien, la duración, según la metodología SCRUM, no debería superar las 8 hs., la forma en que son recibidos los requerimientos (con escasa información o representan funcionalidades grandes), suele requerir más tiempo para comprender el alcance del tema o para desglosar las tareas.

  - Incentivar el intercambio de ideas y opiniones para que surjan mejores propuestas y depurar los requerimientos.

  - Desglosar correctamente las tareas ya que actualmente no se puede visualizar claramente el avance diario. Esto es necesario como paso previo para lograr mejorar las técnicas de estimación.

    Al momento de abordar nuevos desarrollos y objetivos de gran magnitud, lo mejor es desglosarlos en tareas más pequeñas y manejables. La EDT (Estructura de Desglose del Trabajo) organiza y establece el alcance total del proyecto y representa el trabajo especificado en la definición del alcance. Permite identificar y definir todos los esfuerzos requeridos, asignar las responsabilidades a los elementos de la organización. A partir de la EDT se establece un cronograma y presupuesto adecuado para la realización de los trabajos.

    Impacto: la EDT es importante en los demás procesos del proyecto, porque es la base para el control de costos, la asignación de recursos, el cronograma, el análisis de riesgos del proyecto y el seguimiento.

  - Se podría considerar eliminar la etapa de análisis de la iteración y aplicarla de modo previo. Esto significa que los analistas toman los temas que se encuentran en el backlog y comienzan su especificación mientras que el equipo de desarrollo se encuentra ejecutando una iteración independiente (como equipo paralelo).

    Una vez que el analista finaliza la historia de usuario, ésta queda disponible para ingresar en el siguiente backlog del sprint. Antes de la reunión de planificación del nuevo sprint, el equipo de desarrollo se reúne con los analistas para tomar conocimiento de los requerimientos que se encuentran disponibles e introducirse en los temas (se transmiten los objetivos, alcances, negocio, aspectos técnicos, etc).

Cuando el equipo de desarrollo (programadores y testers) adquirió el conocimiento necesario, se realizaría la reunión de planificación en la cual se finalizaría la revisión de la historia de usuario, se haría el desglose de tareas, la estimación y se acuerdan los requerimientos que se tratarán en la iteración.

Una de las tareas más difíciles de estimar es el análisis, dado que a medida que se avanza, pueden identificarse nuevos requerimientos que se suman a los iniciales y hacen que sea necesario replanificar.

La etapa de análisis puede ser gestionada y organizada aplicando una metodología diferente al SCRUM, tal como Kanban que daría la posibilidad de una administración más fácil y dinámica teniendo en cuenta que:

- Las iteraciones no necesariamente deben tener un tiempo fijo. Su duración puede depender de la pila de producto y la mejora del proceso.
- El compromiso es opcional. El analista debería ajustarse a las fechas de entrega que establezca el Comité de Producto.
- No requiere estimación.
- En cualquier momento se podría ingresar requerimientos a la pila y reorganizar las prioridades.
- Si bien no es obligatorio. Se podría hacer una reunión de planificación semanal para realizar ajustes sobre el tablero, sobre la disponibilidad de los miembros que participan o de mejora de proceso.
- Participan sólo los analistas funcionales, lo que lleva a reuniones diarias cortas y ágiles.

Impacto: realizar el análisis por fuera del sprint permite que el analista tenga mayor margen de tiempo para elaborar las historias de usuario, reduciendo la posibilidad, que ante demoras, se impacte negativamente en la fecha de finalización estimada. Separarlo lograría una iteración más precisa en cuanto a tiempos, porque el equipo de desarrollo comenzaría con requerimientos ya refinados y ante eventuales dudas contaría con el soporte del analista funcional.

- **Mejorar la reunión diaria:**
  - Mejorar la dinámica. Se sugiere establecer como secuencia de participación, el orden en que los miembros se encuentran ubicados, mejorando la fluidez de los relatos y evitando tiempos de silencio

contraproducentes para este tipo de reunión. Asimismo se lograría que se escuchen las voces de todos los participantes.

- Reducir la duración de la reunión considerando que cada miembro no debe exponer más de un minuto.

- Adecuar el área de desarrollo para lograr un clima privado y más apropiado para trabajar. Evitar todo tipo de interrupciones (Por ejemplo, los telefónicos o poniendo carteles con horarios de reuniones). Actualmente se llevan a cabo en la cocina, el cual es un lugar público de acceso permanente de personas que generan, sin quererlo, distracciones y/o interrupciones.

Impacto: generar un clima de trabajo óptimo para poder manejar mejor los tiempos y lograr la mayor participación.

- **Retomar la realización de las reuniones de presentación** que fueron dejadas de lado por las exigencias para la exposición (horarios, adecuación del lugar físico, armado de power point, preparación de datos y de sitio, evaluación del orador por parte de la gerencia, tarea de estimación de la presentación, etc.).

  Deberían ser dinámicas, rápidas e informales. Esto sería conveniente ya que en esta reunión los miembros del equipo conjuntamente con los involucrados, repasan los requerimientos iniciales, resaltan los beneficios de la solución y de las tecnologías seleccionadas, etc. Se debe propiciar un clima de diálogo e intercambio entre los presentes muy fluido.
  Es necesario que entre los participantes se encuentre presente el consultor interno que generó el requerimiento.

Impacto: recuperar una práctica importante que permite dar a conocer todo el potencial de la solución, evacuar todas las dudas y discutir sugerencias.

- **Ampliar la convocatoria a la reunión de presentación de requerimientos solicitados**, incorporando, además de al dueño del producto y al equipo completo de desarrollo, a toda aquella persona que conozca con precisión las necesidades y operatividad de los municipios. El PO debe brindar la mayor cantidad de detalles del requerimiento, y capacitar o delegar su función en otro consultor que pueda asumir este rol cuando él no se encuentre disponible.

Impacto: la presencia y participación de los clientes internos con más experiencia permitirá reforzar y definir con mayor certeza los requerimientos correctos, conocer aspectos generales y particulares del negocio como también aspectos técnicos. Se lograría un mayor conocimiento y un inicio de ciclo de desarrollo más sólido, que aquel que comienza habiendo relevado requerimientos sólo de parte del dueño de producto.

- **Replantear el modo de llevar adelante las reuniones de presentación** ya que actualmente exigen una diversidad de requisitos de exposición que dificulta la práctica. Este tipo de reuniones, tan vitales para el proceso, debieran ser menos estructuradas y con mayor interacción entre los involucrados para aprovechar al máximo la experiencia y el conocimiento de todos. La participación de los consultores es imprescindible ya que de otro modo solo reciben el software junto a los manuales de implementación de mano del encargado de despliegue.

  Se podría distribuir al final de la reunión una breve encuesta de satisfacción que involucre las siguientes preguntas:
  - Aspectos positivos a destacar de la reunión
  - Aspectos a mejorar

  Impacto: si esta última instancia previa al despliegue en todos los clientes finales es productiva se logrará asegurar que:

  Lo desarrollado cumpla las expectativas de los clientes internos y externos.

  Se eliminen los defectos en el producto.

  Los consultores participantes conozcan el alcance de la solución, las especificaciones técnicas y la operatividad del sistema. Estarán preparados para capacitara otros compañeros y podrán realizar las implementaciones e instrucción en los municipios.

  El gerente comercial conozca el alcance y los servicios que entrega el producto para poder ofrecerlo a los clientes e interesados.

  Tras haber implementado mejoras en etapas previas de validación y definición de requerimientos, habrá mayores posibilidades de una buena devolución en esta reunión de presentación, logrando de este modo, que el equipo de desarrollo aumente su confianza y motivación.

- **Realizar las reuniones de retrospectiva al final de cada iteración.** El objetivo de la reunión es identificar las fortalezas de un equipo para poder construir a partir de ellas. En la práctica actual no se realizan al final de la iteración sino que en general, abarcan más de una. La duración de una iteración puede variar entre uno y tres meses, por lo cual, la retroalimentación de información sobre aspectos positivos y negativos suele llegar desfasado. Recibir información no oportuna impide la rápida adecuación del proceso.

  La reunión debe realizarse al final de cada iteración con todos los miembros de los equipos participantes presentes. La metodología de la reunión debe ser dinámica y con objetivos concretos que permitan tratar la problemática o avanzar en temas de interés, buscando siempre lograr una participación más comprometida de todos los involucrados.

El Scrum Master podría ampliar las preguntas del checklist (¿que se hizo bien?, ¿Aspectos a mejorar?) incluyendo otras observaciones que desee aportar el equipo de desarrollo para mejorar las próximas reuniones de presentación.

Impacto: es la entrada al ciclo de mejora continua del proceso, por ello, los beneficios que se aportarían son:

Reemplazando el checklist, que se limita a indicar que se hizo bien y que se hizo mal, por diferentes herramientas de comunicación (Ej.: torbellino de ideas, espina de pescado, circulo de preguntas, debate, diferentes dinámicas que incentiven la creatividad, la integración y la construcción, etc.), se lograrán más y mejores aportes y seguramente un mayor compromiso de los participantes.

Las prácticas o mejoras que se consideren oportunas quedaran reflejadas en un ítem de acción con un responsable y fechas establecidas, el cual debe ser controlado en la siguiente retrospectiva

---

- **Mejorar la dinámica de la reunión retrospectiva** para incentivar la participación de los miembros del equipo motivando los aportes personales que permitan identificar buenas prácticas y aspectos del proceso que requieren ser ajustados. Entre las técnicas dinámicas conocidas se sugieren:

  - **Estrella de Mar**: se basa en crear un diagrama con forma de estrella de mar que permite marcar cinco cuestiones a tratar, evitando centrarse solo en lo bueno y lo malo:

    1. Qué hay que continuar haciendo: cosas que han dado buen resultado y que deberían seguir haciéndose.
    2. Menos de: prácticas que no están dando el resultado esperado.
    3. Más de: prácticas que se desean desarrollar más o que no se están aprovechando al máximo.
    4. Dejar de hacerlo: cosas que no son útiles o no agregar valor.
    5. Comenzar a hacerlo: sugerencias de cosas nuevas a probar.

  - **Barco de Vela**: ayuda a reflexionar sobre oportunidades, riesgos y problemas que ha habido en la iteración:

    1. Se comienza dibujando un barco en una pizarra y la tierra a donde debe llegar (meta del equipo, lo que se quiere lograr a nivel de procesos, de desarrollo o de calidad).Es importante que el equipo acuerde y refleje en la pizarra estos ítems.

    2. El equipo debe imaginar que viaja en el barco hacia esta tierra “prometida”. En la escena hay otros elementos que también hay que dibujar:
       - un sol: las cosas que nos iluminan y alegran el camino.

- un ancla (dibujada saliendo del barco hacia el suelo): las cosas que no nos permiten avanzar.
- rocas: los riesgos que nos encontramos en el camino.
- viento: aquellas cosas positivas que nos empujan y acercan a nuestra meta.

3. La idea es que una vez hecho el dibujo y explicado que significa cada elemento, se otorgan cinco minutos para que los miembros del equipo individualmente escriban en post-it qué elementos de cada tipo han encontrado en este sprint. Es recomendable que cada uno escriba como mínimo 3 cosas de cada tipo.

4. Después, por turnos, cada persona va colocando sus post-it alrededor de cada elemento (sol, ancla, etc.) y explica por qué ha decidido escribir eso en los post-it. Luego se agrupan los post-it similares.

5. Posteriormente se analiza el porqué de las cosas mencionadas y se proponen cuáles son los aspectos que se deberían priorizar para mejoraren el próximo sprint.

6. Por último toca definir qué vamos a hacer para resolver los asuntos que hemos priorizado. Tenemos que salir de la retrospectiva acordando qué vamos a hacer, cuándo va a estar hecho y quién/quiénes son los responsables de hacerlo.

Impacto: reuniones más dinámicas y participativas, permiten encontrar más opciones de mejora y compromiso.

- **Tomar métricas y realizar un seguimiento que permita retroalimentación para toma de decisiones:** Actualmente la empresa utiliza las métricas de velocidad del equipo y burn down chart para conocer el avance diario. Como las iteraciones son de tiempo variable, las métricas actuales no representan escenarios 100% comparables por lo que no se aprovechan los resultados obtenidos.
  Si bien SCRUM es flexible en cuanto a la cantidad y tipo de métricas a tomar o no, se sugiere, dada la cantidad de pedidos de error por despliegues defectuosos, incorporar la medición de “Cantidad de defectos identificados en el cliente final por iteración”.

Impacto: Esto permitiría identificar el porcentaje de defectos en relación a las soluciones desplegadas, encontrar los posibles orígenes y/o causas, y tomar acciones correctivas que permitan mejorar futuros sprint.

5.8 Propuesta de Implementación

A los fines de optimizar el proceso de desarrollo software se han agrupado las propuestas antes mencionadas, tratando de establecer una hoja de ruta factible, que pueda ser implementada de modo escalable para lograr el resultado deseado. Esto daría el espacio para una apropiación e internalización de los cambios antes de pasar a la siguiente etapa.

Si se consulta a la gerencia, sucede que desean aplicar todas las propuestas, pero si decidimos abordar todos estos cambios desde el primer día, las dimensiones del proyecto de mejora que acabaremos diseñando, serán tan grandes que tendrán altas probabilidades de interrupciones y a la vez, será tan complejo y abarcará tantas áreas, que tendrá pocas posibilidades de éxito.

La implementación debería abordar los problemas de forma ordenada, no todos a la vez. A medida que aumenta la complejidad del sistema, también lo hace la madurez de la organización al abordar dicha complejidad.

Este modo de implementación escalonado, facilitaría el cambio cultural dentro de la organización a lo largo del tiempo, permitiendo una mejor adaptación de los equipos. En general, todo cambio provoca resistencias, y es muy probable que ante cambios importantes en la gestión del proyecto y en los modos de realizar las tareas, se produzcan molestias. La implementación progresiva, por paquetes, de las fases permitiría que los usuarios se adapten y puedan apreciar las ventajas que traen aparejados estos cambios no solo para el proceso en sí, sino también para ellos mismos. Se debe propiciar una cultura que involucre a las personas de manera activa en la búsqueda de oportunidades de mejora del desempeño del proceso, las actividades y el producto.

Es necesario considerar esta implementación como un proyecto de gestión de cambios que se subdividirá en partes manejables que deberían tratarse como subproyectos con sus propios indicadores de éxito. De este modo se conseguirán rápidamente algunas de las ventajas esperadas y servirá para lograr motivación y apoyo en general.

Si bien el enfoque principal del cambio está puesto en mejorar el proceso, los beneficios del trabajo y tiempo invertido deben funcionar en cada nivel: directivos, jefes de proyectos, equipo de desarrollo y usuario municipal.

5.9 Paquetes de Mejora Propuestos

5.9.1 Paquete 1 - enfocado al producto, al proceso y a las personas

En primer lugar se espera que el equipo pueda identificar y trabajar con una mejor calidad de requerimientos y especificaciones, y en segundo lugar, establecer buenas prácticas y tips que mejoren el desempeño de cada rol. Por último, se pretende focalizar en las personas -piezas fundamentales del proceso- mejorando aspectos de motivación, del trabajo en equipo y de la metodología.

Un requerimiento es el punto de partida de todo el proceso, por lo cual, se considera un punto clave a resolver inicialmente. La incorporación de revisiones periódicas y mejoras en la reunión de planificación le permite al equipo lograr un desarrollo mejor sustentado, confiable y acorde a las expectativas del cliente. También reducirían los defectos originados por análisis defectuosos.

Con la definición de estándares en el análisis, se lograría que los analistas cuenten con un marco de trabajo organizado que defina con claridad sus tareas y mejore los tiempos de especificaciones y de análisis. Disponer de documentos escritos estandarizados reduce la posibilidad de duplicar contenido, amplía la visión de aspectos técnicos, funcionales y de negocio, aporta claridad en la lectura, facilita búsquedas y reduce pérdidas de tiempo.

Las mejoras indicadas para el equipo de programadores, permite lograr un código más limpio, de fácil lectura y comprensión, optimizado, mejor diseñado. Estos tips, también impactan en el trabajo del tester, dado que éste recibe un código depurado, validado y libre de defectos técnicos.

Los ajustes en la metodología y en aspectos relacionados sobre la gestión del proyecto le otorgarían al Scrum Master una plataforma de trabajo clara y organizada, que le servirían de base para coordinar las iteraciones y los equipos, a la vez le servirán de referencia para comenzar a toma de métricas y aplicar mejora de proceso.

En esta primera etapa se sugiere implementar las siguientes propuestas de mejora:

Proceso:

- Ampliar la convocatoria a la reunión de presentación de requerimientos solicitados.
- Eliminar la etapa validación del cumplimiento de los criterios de aceptación.
- Definir iteraciones de duración estable de dos semanas.
- Optimizar la planning del sprint.
- Mejorar la Reunión Diaria.

Producto:

- Reemplazar la etapa de validación de los criterios de aceptación por una etapa de Revisión (Review).
- Establecer estándares para la escritura de historias.
- Establecer estándares para la escritura de defectos.
- Establecer estándares para la especificación de parámetros y permisos especiales en Manual de Configuración.
- Tomar como criterio obligatorio el agregar funciones que validen todos los campos.
- Realizar acciones de auditoria de código.

Personas:

- Concientizar en la necesidad de que los equipos sean estables (si el equipo tuvo éxito, que se sostenga en el tiempo).
- Motivar la autogestión y autoorganización.
- Incentivar al trabajo en equipo, la comunicación interna y la motivación del grupo a través de talleres dinámicos, en los que se trabaje sobre estos puntos y que estén a cargo de personas con experiencia en este tipo de capacitación.

Proyecto:

- Establecer un único canal de comunicación entre los clientes internos y el equipo de desarrollo.

A continuación se presenta el plan de mejora de cada una de las propuestas incluidas en este paquete:

5.9.1.1 Plan de mejora - Paquete 1

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Ampliar la convocatoria a la reunión de presentación de requerimientos solicitados	Scrum Master	1 sem.	-	No requiere recursos financieros extras	Cant. de asistentes de las diferentes áreas que se desean involucrar	Scrum Master
Eliminar la etapa validación del cumplimiento de los criterios de aceptación	Testers	2 sem.	Tiempo de reuniones del equipo (analista funcional, tester y programador).	No requiere recursos financieros extras	Cant. de ciclos de prueba en escenarios similares	Testers Scrum Master
Definir iteraciones de duración estable de dos semanas	Scrum Master	3 sem.	Backlog apropiadamente analizado, depurado y priorizado. Equipo disponible.	No requiere recursos financieros extras	Duración de las distintas iteraciones. Cant. de métricas definidas y realizadas. Cant. de cambios de requerimientos durante las iteraciones. Cant. de trabajo planificado vs no planificado.	Scrum Master
Optimizar la planning del sprint	Scrum Master		Disponibilidad y tiempo del equipo designado, del PO y del Líder Técnico, TV, pizarra/s y sala de reunión	No requiere recursos financieros extras	Asistencia de todos los miembros del equipo designado.	Scrum Master Dueño del producto
Optimizar la reunión diaria	Scrum Master Equipo de Desarrollo	2 sem.	Disponibilidad y tiempo del Scrum Master para investigar, evaluar alternativas y planificar. Pizarra para el tablero SCRUM en el espacio de trabajo del equipo.	Costo de la pizarra y elementos de librería	Duración de exposición por miembro. Duración total de la reunión.	Scrum Master

Tareas	Resp. de tarea	Tiem	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimien
Reemplazar la etapa de validación de los criterios de aceptación por una etapa de revisión (Review)	Scrum Master	2 sem.	Tiempo de las áreas de desarrollo y de atención al cliente.	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por errores detectados en el cliente.	Scrum Master
Establecer estándares para la escritura de historias	Scrum Master Analista Funcional	4 sem.	TFS. Tiempo de un Analista Funcional y del Scrum Master	No requiere recursos financieros extras	Horas completadas en la carga de historias por tipo de pedido en el TFS.	Scrum Master
Establecer estándares para la escritura de defectos	Scrum Master Analista Funcional	4 sem.	TFS. Tiempo de un Analista Funcional y del Scrum Master	No requiere recursos financieros extras	Horas completadas en la carga de defectos en el TFS.	Scrum Master
Establecer estándares para la especificación de parámetros y permisos especiales en manual de configuración	Analista Funcional	1 sem.	TFS	No requiere recursos financieros extras	Cant. de parámetros a definir por página vs Cant. de parámetros definidos de acuerdo a los estándares.	Analista Funci
Tomar como criterio obligatorio el agregar funciones que validen todos los campos	Programador Scrum Master	1 sem.	TFS. Tiempo de los programadores	No requiere	Cant. de ciclos de prueba por solución	Scrum Master

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimien
Realizar acciones de auditoria de código	Programador Scrum Master	2días	TFS Tiempo de los programadores	No requiere	Tiempo de respuesta de cada página o proceso. Esfuerzo de programación en funcionalidades similares. Cant. de líneas de código en funcionalidades similares.	Scrum Master
Concientizar en la necesidad de que los equipos sean estables (si el equipo tuvo éxito, que se mantenga en el tiempo)	Scrum Master Líder Técnico (jefe de área)	3 sem.	Disponibilidad y tiempo del Scrum Master y Líder Técnico.	No requiere recursos financieros extras	Velocidad del equipo. Cant. de solicitudes por tipo y complejidad que ingresan a la iteración. Cant. de solicitudes de retrabajo.	Scrum Master
Motivar a la autogestión y autoorganización	Scrum Master	2 sem.	Disponibilidad y tiempo del Scrum Master.	No requiere recursos financieros extras	Velocidad del equipo. Cant. de solicitudes completadas vs ingresadas en la iteración.	Scrum Master
Incentivar al trabajo en equipo, la comunicación interna y la motivación del grupo a través de talleres dinámicos	Scrum Master	1 sem.	Disponibilidad y tiempo de los participantes. Especialista externo sobre la temática. Proyector y sala de reunión	Honorarios del Especialista	Niveles de satisfacción del personal a través de cuestionarios periódicos.	Scrum Master
Establecer un único canal de comunicación entre los clientes internos y el equipo de desarrollo	Scrum Master	3 sem.	N/A	No requiere recursos financieros extras	Cant. de requerimientos no planificados y no ingresados a través del Scrum Master. Cant. de trabajo planificado vs no planificado.	Scrum Master Responsable de Atención al Cliente. Líder Técnico

5.9.2 Paquete 2 - enfocado en mejorar la entrega del producto desarrollado

Se centra en la optimización de las reuniones de presentación al cliente y en el logro de mejores devoluciones en las reuniones de retrospectiva de modo que sirvan de base para un proceso eficiente y adecuado a las necesidades de negocio actuales. En cuanto a las áreas de programación y testing se busca implementar estrategias que minimicen al máximo los defectos en los productos entregables y mejoren su calidad. En este paquete, el Scrum Master se enfoca en tomar métricas que mejoren el proceso, la metodología y en lograr equipos multifuncionales pequeños que se adapten (según las capacidades) a las diferentes necesidades de desarrollo y/o urgencias.

En el paquete 2 se sugiere implementar las siguientes propuestas de mejora:

Proceso:

- Retomar las reuniones de presentación.
- Replantear el modo de llevar adelante las reuniones de presentación.
- Realizar las reuniones de Retrospectiva al final de cada iteración.
- Tomar métricas y realizar un seguimiento que permita retroalimentación para toma de decisiones.

Producto:

- Establecer procedimientos de trabajo según el tipo de requerimiento ingresado.
- Implementar el testing durante todo el proceso, no al final.
- Establecer pautas para testing, que tiendan a eliminar los defectos en el cliente final.
- Incorporar el versionado de componentes a la plantilla de prueba como un elemento obligatorio.
- Establecer trabajo colaborativo entre los programadores para realizar el test unitario.
- Aplicar tips que aporten al trabajo de los programadores.
- Incorporar la programación de a pares.

Persona:

- Redefinir tareas dentro del equipo en base a la experiencia, capacidad e interés de cada persona.
- Definir equipos multifuncionales más pequeños con capacidades y características diferentes. Esto permite establecer una estrategia para afrontar urgencias que requieran atención inmediata con un equipo altamente capacitado.

A continuación se presenta el plan de mejora de cada una de las propuestas incluidas en este paquete:

5.9.2.1 Plan de mejora - Paquete 2

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Retomar las reuniones de presentación	Scrum Master Equipo de Desarrollo	1 sem.	Disponibilidad y tiempo del Scrum Master y del equipo de desarrollo para organizar y hacer la convocatoria a los involucrados.	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por errores y/o faltantes detectados por el cliente interno.	Scrum Master
Replantear el modo de llevar adelante las reuniones de presentación	Scrum Master	1 sem.	Tiempo de las áreas de desarrollo, de atención al cliente y gerencia. Equipo tecnológico adecuado.	No requiere recursos financieros extras	Tiempo de duración de las reuniones. Control de asistencia de los consultores y gerencia.	Scrum Master
Realizar las reuniones de retrospectiva al final de cada iteración	Scrum Master	3 sem.	Disponibilidad y tiempo del Scrum Master para investigar, evaluar alternativas y planificar.	No requiere recursos financieros extras	N/A	Scrum Master
Tomar métricas y realizar un seguimiento que permita retroalimentación para toma de decisiones	Scrum Master	6 sem. (equivalente a 3 iter)	Disponibilidad y tiempo del Scrum Master para investigar, evaluar alternativas y planificar. Pizarra para exponer las métricas más representativas para el equipo.	No requiere recursos financieros extras	Cant. de defectos identificados en el cliente final por iteración. Cant. de métricas definidas y realizadas.	Scrum Master Dueño del Producto.

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Establecer procedimientos de trabajo según el tipo de requerimiento ingresado	Analista Funcional	1 sem.	TFS	No requiere recursos financieros extras	Horas dedicadas a completar una historia (incluye análisis, programación y testing) por tipo y complejidad del pedido en el TFS Horas dedicadas a completar un defecto (incluye verificación, escritura, corrección del código y testing) por tipo y complejidad del pedido en el TFS	Analista Funcional
Implementar el testing durante todo el proceso, no al final	Tester Scrum Master	2 sem	TFS Tiempo del Analista Funcional, Tester, Programador y Scrum Master.	No requiere recursos financieros extras	Cant. de ciclos de prueba en escenarios similares. Cant. de solicitudes de retrabajo por errores detectados en el cliente.	Scrum Master
Establecer pautas para testing, que tiendan a eliminar los defectos en el cliente final	Tester Scrum Master	2 sem.	Tiempo del Tester, Programador y Scrum Master.	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por errores detectados en el cliente.	Scrum Master
Incorporar el versionado de componentes a la plantilla de prueba como un elemento obligatorio	Tester Scrum Master	1 día	TFS Tiempo del Tester, Programador y Scrum Master.	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por errores detectados en el cliente cuyo versionado difiere del testeado y homologado.	Scrum Master
Establecer trabajo colaborativo entre los programadores para realizar el test unitario	Programador Scrum Master	1 sem.	TFS Tiempo de los programadores	No requiere	Cant. de ciclos de prueba por solución	Scrum Master

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Considerar tips que aporten al trabajo de los programadores	Programador Scrum Master	1 sem.	TFS Tiempo de los programadores	No requiere	Tiempo de desarrollo de las soluciones.	Scrum Master
Incorporar la programación de a pares	Programador Scrum Master	2 sem.	TFS Tiempo de los programadores	No requiere	Tiempo de desarrollo de las soluciones. Cant. de ciclos de prueba por solución.	Scrum Master
Redefinir tareas dentro del equipo en base a la experiencia, capacidad e interés de cada persona	Scrum Master Líder Técnico (jefe de área)	4 sem.	Disponibilidad y tiempo del Scrum Master, Líder Técnico y del Equipo para participar de reuniones de área.	No requiere recursos financieros extras	Velocidad del equipo. Cant. de solicitudes por tipo y complejidad que ingresan a la iteración.	Scrum Master
Definir equipos multifuncionales más pequeños con capacidades y características diferentes y elegir a los líderes de entre los miembros del grupo.	Scrum Master Dueño del Producto Líder Técnico (jefe de área)	3 sem.	Disponibilidad y tiempo del Scrum Master, Dueño del Producto y Líder Técnico.	No requiere recursos financieros extras	Velocidad del equipo. Cant. de solicitudes completadas vs ingresadas en la iteración. Cant. de solicitudes de defectos por retrabajo.	Scrum Master

5.9.3 Paquete 3 - enfocado en la optimización del proceso (diseño e implementación)

Incorpora mejoras a las prácticas habituales que permitan monitorear el avance de cada sprint y la retroalimentación del proceso, a la vez que fortalezcan y aumenten las capacidades de cada área.

En paquete 3 se sugiere implementar las siguientes propuestas de mejora:

Producto:

- Diseñar un checklist para escritura y ejecución de pruebas.
- Incorporar como práctica la realización de pruebas de regresión.
- Establecer un equipo de desarrollo destinado a investigar, proponer y encarar la automatización de pruebas unitarias.

Proceso:

- Eliminar la etapa de análisis de la iteración y de aplicarla de modo autónomo.
- Mejorar la dinámica de la reunión de retrospectiva.

Proyecto:

- Incorporar información de utilidad para ayudar al equipo en su tarea de autogestión y autoorganización. Se podrían considerar los siguientes datos:
    - Capacidad de cada persona: ayuda a conocer la disponibilidad diaria, en horas, en el proyecto. Es la base para estimar la fecha de finalización de la iteración. Los equipos conocen con exactitud el tiempo en el que debería avanzar en cada proyecto.
    - Detalle de horas por día: en recuadro se indica la cantidad de horas estimadas total para la iteración (por cada uno de los miembros), el avance diario y las horas restantes de la iteración. El total de horas pendientes se reflejan en el gráfico Burndown Chart.
    - Reestimación en el post it: diariamente cada miembro debe actualizar en el post-it el pendiente por tareas o su reestimación en caso de haberse efectuado.

Persona:

- Desarrollar el conocimiento, la experiencia y habilidades de las personas como parte de las actividades generales de gestión de la calidad de la organización.

A continuación se presenta el plan de mejora de cada una de las propuestas incluidas en este paquete:

5.9.3.1 Plan de mejora - Paquete 3

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Diseñar un checklist para escritura y ejecución de pruebas	Tester Scrum Master	3 días	TFS Tiempo del Tester y Scrum Master	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por omisión de testing.	Scrum Master
Incorporar como práctica la realización de pruebas de regresión	Tester Scrum Master	1 sem.	TFS Tiempo del Tester y Scrum Master	No requiere recursos financieros extras	Cant. de solicitudes de retrabajo por errores detectados en el cliente.	Scrum Master
Establecer un equipo de desarrollo destinado a investigar, proponer y proyectar la automatización de pruebas unitarias	Programador Scrum Master	4 sem.	TFS Tiempo de los programadores	No requiere	Grado de avance de la investigación.	Programadores
Eliminar la etapa de análisis de la iteración y aplicarla de modo previo	Scrum Master	2 sem.	N/A	No requiere recursos financieros extras	Cant. de estimaciones que arrojan resultados similares a los planificados.	Scrum Master
Optimizar la dinámica de las reuniones de retrospectiva	Scrum Master	2 sem.	Tiempo destinado a la reunión de retrospectiva. Sala de reunión.	No requiere recursos financieros extras	Cant. de iteraciones incluidas. Control de asistencia de todos los participantes.	Scrum Master

Tareas	Resp. de tarea	Tiempo	Recursos necesarios	Financiación	Indicador seguimiento	Resp. de seguimiento
Incorporar información de utilidad para ayudar al equipo en su tarea de autogestión y autoorganización	Scrum Master Equipo de Desarrollo	1 sem.	Disponibilidad y tiempo del Scrum Master para investigar, evaluar alternativas y planificar. Disponibilidad y tiempo del Scrum Master, Líder Técnico y Equipo. Pizarra y sala de reunión	No requiere recursos financieros extras	Niveles de satisfacción del personal a través de la reunión de retrospectiva.	Scrum Master
Desarrollar el conocimiento, la experiencia y habilidades de las personas como parte de las actividades generales de gestión de la calidad de la organización	Scrum Master	4 sem.	Disponibilidad y tiempo del Scrum Master, Líder Técnico y Equipo. Proyector y sala de reunión	No requiere recursos financieros extras	Niveles de satisfacción del personal a través de cuestionarios periódicos.	Scrum Master Líder Técnico

Conclusión

El objetivo principal de este trabajo fue detectar aquellos aspectos del proceso factibles de ser mejorados y aportar soluciones que puedan incrementar la calidad del producto. Estas propuestas, respetan la metodología de desarrollo SCRUM pero intentan adecuarla a las necesidades y particularidades de GobFactory.

Se presentan paquetes escalables de soluciones que abordan ajustes en diferentes etapas y funciones. Las adecuaciones pretenden alcanzar la mayor optimización posible del proceso, teniendo en cuenta aspectos del ciclo de vida, propuestas de trabajo para el equipo de Análisis, Testing y Programadores, redefinición de criterios en la aplicación de la metodología, y mejoras en la gestión de recursos y del proyecto.

Este trabajo no incluye demostración ni medición de resultados luego de la implementación de las propuestas. Esta fase, queda pendiente para un trabajo posterior dado que es necesario contar con al menos un semestre para poder aplicar consecutivamente los paquetes. Con posterioridad, se podrá medir o evaluar el rendimiento y la satisfacción del equipo con los ajustes realizados sobre el proceso, la metodología y las nuevas prácticas en general.

En este recorrido se identificaron aspectos necesarios a trabajar para poder cumplir con los requerimientos principales de la empresa. Para ello, se definieron objetivos específicos que tienden a cubrir estas necesidades. Se detalla el abordaje de éstos a continuación:

1. **Identificar posibles áreas de desarrollo a mejorar**

   Durante la etapa de relevamiento se detectaron aspectos en las diferentes áreas involucradas en el proceso que demoraban y obstaculizaban, en mayor o menor grado, el desarrollo de una solución. Por ejemplo, fallas en la comunicación, disparidad de criterios en análisis, programación y testing, entre otros, hicieron que fuera necesario buscar soluciones para cada área en particular y que impactaran positivamente en las demás áreas, etapas o actividades del proceso.

   Análisis: se detectan prácticas y procedimientos no estandarizados tales como la escritura de historias, defectos y parámetros, lo cual afecta a la calidad de la información que se desarrolla dentro del equipo. También se han observado inconvenientes en las etapas de planificación, elicitación de requerimientos y validación de historias.

   Programación: pruebas unitarias insuficientes y deficientes ya que se despliegan soluciones con elevado porcentaje de defectos que podrían haber sido prevenidos. Tampoco se utilizan procedimientos estandarizados en programación ni funciones de validación para todos los campos que lo requieren.

   Testing: esta tarea se realiza al final del proceso lo que implica que se llega a la etapa final con gran cantidad de defectos que podrían haberse eliminados en las etapas preliminares. No hay establecidos procedimientos y criterios comunes para definir la cobertura y la escritura de los casos de prueba.

Las áreas que llevan adelante las diferentes etapas del proceso actual, no logran los resultados esperados y no alcanzan los tiempos de respuesta estimados por el comité o por el propio equipo durante la reunión de planificación. Tampoco han podido aplicar mejoras al proceso o a sus propias actividades que realmente hayan impactado positivamente en el producto final.

La identificación y solución a estos problemas fortalecería el ciclo de vida, asegurando que todos los miembros conozcan los requerimientos comprometidos, la solución esperada y las limitaciones técnicas posibles, garantizando la calidad del producto.

Dentro de las propuestas de este trabajo, se incluyen mejoras a las prácticas del proceso que abarcan estas tres áreas principales, a los fines de fortalecer cada una de ellas y el trabajo del equipo en conjunto.

2. **Detectar prácticas relevantes del proceso de desarrollo actual de WGob, que necesiten ser revisados para su optimización.**

En cada una de las áreas que forman parte del ciclo de vida se han detectado actividades factibles de ser mejoradas. Estas mejoras tienen que ver con metodología, proceso, iteraciones, comunicación, actividades, trabajo en equipo, avance y seguimiento del proyecto.

Contar con un proceso en el que no se realiza la validación a término, que sufre reiteradas interrupciones por requerimientos no planificados, y que tiene equipos e iteraciones variables, hace que sea difícil estimar y medir. Esto impacta negativamente en el rendimiento del equipo y en la calidad del producto, generando costos, retrabajo, retraso en las entregas y un ambiente poco favorable para la mejora continua.

A partir de estas observaciones se proponen ideas que pueden ayudar a resolver o mejorar estas situaciones. Algunas de ellas son: retomar la realización de la reunión de presentación al cliente y de la retrospectiva al final de cada iteración, cambiar la dinámica de las reuniones en general, mejorar el tablero SCRUM, lograr iteraciones fijas y equipos estables, tomar métricas, etc.

3. **Disminuir reclamos de clientes.**

El gran número de reclamos que ingresan a la empresa vía telefónica o personalmente a través de los consultores y que son reflejados a través de las solicitudes de requerimientos de tipo error y en menor medida de modificación (por ampliación de requerimientos), representan un indicador que refleja que el proceso es insuficiente y/o ineficiente y que existen prácticas que requieren ajustes:

- En general, todas las especificaciones y decisiones sobre los requerimientos ingresados a la pila, son determinadas por el dueño del producto y su criterio es el que condiciona las características de la solución que se lleva adelante.

Los clientes internos (quienes representan y conocen a la perfección a cada municipio) no suelen participar de la definición de estos requerimientos. Es por esto que durante la reunión de planificación, el equipo no cuenta con todos los puntos de vistas suficientes como para lograr una solución adecuada en cuanto a tecnología y a negocio.

- Al finalizar un desarrollo, se despliega en los municipios sin previa presentación a los clientes internos. Al momento de implementar suelen detectarse escenarios operativos que no se tuvieron en cuenta, requerimientos que no fueron considerados y problemas con el entorno/tecnología. Estos inconvenientes ingresan a la empresa como pedidos de errores (considerado implícitamente como retrabajo). La comunicación entre los clientes internos, el PO y equipo de desarrollo, es clave para mejorar la calidad del producto y satisfacer las expectativas de clientes internos y externos. Debido a esto, se sugiere la participación de todos los involucrados en la planning y optimizar las reuniones de presentación como una manera de atenuar los despliegues defectuosos.

- En la mayoría de los casos, el analista, el programador y el tester trabajan en solitario, sin colaboración o interacción con el resto del equipo. El analista prepara la historia con la información recibida en la planning, mientras que programador y tester realizan su trabajo basándose sólo en lo indicado en la historia diseñada por el analista. Esta forma de trabajo impacta en la calidad del resultado dado que no permite aprovechar al máximo el conocimiento, la experiencia, la creatividad de cada rol, y las ventajas de un trabajo sinérgico.

- En cuanto al testing, éste se realiza en la etapa final del ciclo de vida, lo que implica que la empresa debe asumir un costo mayor a la hora de corregir defectos. Por otro lado, no se realiza testing de regresión, ampliando la posibilidad de despliegues defectuosos dado que los cambios realizados por el programador pueden haber introducidos defectos en funcionalidades estables.

Para reducir el impacto negativo de las prácticas anteriores, las cuales se reflejan en los entregables al cliente, se indicaron mejoras que incluyen la realización de testing a lo largo de todo el proceso, herramientas y algunas buenas prácticas factibles. También se propone un trabajo conjunto y sinérgico de los tres roles (analistas, tester y programadores) para establecer pautas de trabajo consensuadas que aseguren la calidad del producto final.

4. Lograr mayor frecuencia de los despliegues.

Mantener al cliente actualizado en cuanto al producto y brindarle respuestas a sus requerimientos/problemas de forma ágil debería ser una prioridad para la empresa. Por el contrario, se realizan iteraciones largas y variables entre sí en cuanto a tiempo, complicando la tarea del área de atención al cliente, encargada de negociar con los municipios una fecha probable de entrega. También se satura al equipo de trabajo con temas extensos que perjudican la dinámica ya que se agregan temas al backlog aumentando la duración del sprint. Estas

iteraciones largas e indefinidas no permiten al equipo realizar prácticas de mejora del proceso, no pudiendo el Scrum Master tomar métricas que le permitan conocer el estado de situación del proceso, metodología y/o prácticas.

La propuesta planteada de definir iteraciones fijas no mayores a dos semanas busca aumentar la satisfacción del cliente al recibir ajustes, mejoras y nuevas prestaciones del producto en los plazos concretos comprometidos.

5. Mejorar la comunicación interna.

La comunicación es uno de los principales pilares de la metodología ágil, por lo que debería ser fluida y oportuna en todas las direcciones. Trabajar sobre este aspecto es esencial para lograr que el equipo pueda autoorganizarse y autogestionarse y que la solución desarrollada sea la óptima.

Durante el proceso se han detectado aspectos que podrían ser mejorados y que tienen que ver con la información disponible en las reuniones, los perfiles participantes, las metodologías y la comunicación que se genera durante las mismas.

Como se mencionó anteriormente, casi todas las decisiones en general, son determinadas por el dueño del producto y su criterio es el mayor condicionante. Esto, sumado a su escasa disponibilidad de tiempo, y el hecho de que no participan todos los involucrados en las validaciones, impacta negativamente en las tareas de las áreas de desarrollo ya que no se logran validar con exactitud y oportunamente todos los requerimientos solicitados e ingresados a la pila de la iteración.

También se observa que durante las reuniones diarias, de planificación y retrospectiva, el equipo se muestra desinteresado y poco participativo, generándose un clima incómodo y poco propicio para que surjan buenas ideas y aportes.

La disparidad en el nivel de conocimiento que tienen los integrantes del equipo sobre funcionalidades, requerimientos y aspectos técnicos dificultan la definición y especificación de los requisitos y generan defectos en la solución planteada, falta de precisión y errores al determinar el impacto y en la planificación de las entregas.

La participación y el intercambio de opiniones en las reuniones, permitiría que los requerimientos sean correctamente interpretados y detallados, que las soluciones aporten valor al cliente, que se mejore el clima de trabajo, que exista mayor acercamiento y confianza entre las personas y sobre el producto.

Este trabajo propone redefinir los canales de comunicación para lograr precisión en la información y evitar interrupciones al equipo. También sugiere algunas dinámicas para aplicar en las reuniones que intentan lograr no solo una mayor participación, sino que, a la vez, buscan enmarcar el trabajo para obtener el mayor provecho de los encuentros. Un buen trabajo en equipo se logra a través

de una buena comunicación, coordinación, complementariedad, confianza y compromiso

6. **Fortalecer el trabajo en equipo.**

   Un equipo que trabaja motivado y comprometido logra que sus integrantes se sientan incentivados para aportar nuevas ideas y desarrollar sus capacidades y creatividad. Esto generará un efecto sinérgico, que sin duda redundará en beneficios para el equipo y el proyecto.

   Uno de los aspectos que se pudieron advertir fue la necesidad de mejorar el trabajo en equipo en el área de desarrollo. Cada rol enfocado únicamente en su tarea, con diferentes niveles de conocimiento sobre el negocio, no ayuda al trabajo colaborativo ni favorece el intercambio de ideas o el aporte de conocimientos, por el contrario, aísla cada vez más a las personas.
   Como estrategias para ayudar y fortalecer el desempeño se sugieren modos de trabajo que tiendan a reunir en las tareas a los miembros del equipo para mejorar el rendimiento en base a un trabajo colaborativo, consensuado y comprometido.

   Por otro lado, se sugiere implementar equipos multifuncionales. Una vez que éstos logren alcanzar la madurez suficiente para poder trabajar en ese ambiente colaborativo, tendrán mayor capacidad de autogestión y autoorganización, dado el nivel de confianza que se establece entre los integrantes y la diversidad de conocimientos y capacidades. En este caso, serian propicios realizar talleres que ayuden a mejorar la comunicación y respeto entre pares.

Para alcanzar los objetivos empresariales con éxito es fundamental disponer de un producto diferenciado, ágil, seguro, de calidad, y que abarque todos los aspectos que requiere una gestión municipal. Para lograr un producto de calidad es necesario contar con un proceso adecuado a sus características y que pueda ser adaptado a las necesidades particulares de cada cliente. Una gestión eficiente debe tener la capacidad de realizar reingeniería de proceso, a tiempo, a los fines de garantizar su correspondencia a las necesidades del momento. Todo proceso debe ir acompañado de una metodología de trabajo flexible y acorde que permita aprovechar al máximo los recursos.

Este trabajo intentó proporcionar ideas para que el área de desarrollo de software de GobFactory identifique las problemáticas y las necesidades actuales de negocio, optimice su proceso, valorice y potencie los recursos, seleccione e implemente buenas prácticas de SCRUM, aumente la calidad de su producto reforzando las áreas críticas del desarrollo, aplique métricas que aporten a la mejora continua del proceso, y optimice su comunicación interna y con los clientes.

En este marco se presentaron tres módulos con propuestas que han sido agrupadas en paquetes, los cuales fueron trasladados a planes de mejora que permiten visualizarlos de un modo más concreto, calendarizarlos y proyectar su implementación.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Anexo 1: Detalle de Procedimientos según el Tipo de Requerimiento Ingresado

Mejoras en el Equipo de Análisis:

Detalle de procedimientos de trabajo según el tipo de requerimiento ingresado

1. **En el caso de un requerimiento nuevo no debe saltearse las etapas de:**

   - Indagar sobre casos de similares características en la "Herramienta Solicitud de Requerimientos" y en el repositorio del producto para recuperar documentación o código existente que pueda ser reutilizado. Podrá recurrir a casos de uso, reglas de negocio, manuales de configuración, historias, defectos, manuales de usuarios y código. También debe apoyarse en los analistas de mayor experiencia que lo orientarán en las alternativas posibles.

   - Siempre que sea factible, confeccionar un prototipo de la solución. De lo contrario, debe existir ejemplos que detallen del comportamiento de la misma.

   - Cumplir con los pasos de carga de historia de usuario en la herramienta TFS de acuerdo al estándar sugerido anteriormente. Asimismo, deben quedar adjuntos a la historia todos los documentos de soporte recibidos del cliente y/o creados por el equipo.

   - Si se requieren parámetros y/o permisos especiales, describir los mismos en los manuales de configuración correspondiente.

   - Realizar tantas reuniones de análisis con el equipo y el dueño del producto como sean necesarias para depurar dudas y lograr uniformidad de criterios y conocimientos.

   - Validar con el DBA la estructura de la base de datos.

   - Repetir la instancia de validación con el cliente interno y el dueño del producto hasta que se confirme que la solución propuesta es la óptima para cumplir con los requerimientos del cliente.

   - Dejar registrado en el apartado Discusiones de la historia de usuario, cada una de las reuniones en donde hubo validaciones de aspectos técnicos o de negocio, y en las que hayan surgido modificaciones de requerimientos iniciales.

En el caso de un requerimiento de modificación deben cumplirse las siguientes etapas:

- Explorar la funcionalidad vigente.

- Recuperar documentación (casos de uso, reglas de negocio, manuales de configuración, historias, defectos, manuales de usuarios) que permita conocer detalladamente la funcionalidad actual.

- Identificar la existencia de parámetros o permisos que condicionen el comportamiento de la funcionalidad.

- Siempre que sea factible, confeccionar un prototipo de la solución. De lo contrario, debe existir ejemplos que detallen del comportamiento de la misma.

- Cumplir con los pasos de carga de historia de usuario o defecto de mejora en la herramienta TFS, según corresponda. La diferencia entre una historia de usuario y un defecto es el nivel de detalle de la documentación y las tareas involucradas. Además, la historia implica la carga de al menos un caso de prueba, mientras que el defecto no necesita un caso de prueba sino una evidencia que el defecto fue resuelto.

- Tanto la historia de usuario como los defectos de mejora deberían ser cargados considerando los estándares definidos anteriormente. Asimismo, deben quedar adjuntos todos los documentos de soporte recibidos del cliente y/o creados por el equipo.

- Realizar tantas reuniones de análisis con el equipo como sean necesarias para aclarar dudas y lograr uniformidad de criterios y conocimientos.

- Ante un cambio que implique retocar la base de datos, validar con el DBA.

- Repetir la instancia de validación con el cliente interno o dueño del producto hasta que se confirme que la solución es la óptima.

En el caso de un requerimiento de error:

- Replicar el defecto detallado en la solicitud de requerimiento en el servidor utilizado para release, ya que representa el entorno real del cliente.

- Para identificar las causas del defecto, si es posible, ejecutar un software (Ej. SQL Server Profile) que permita analizar y reproducir resultados de la funcionalidad a los fines de determinar el momento exacto en que se produce la falla.

- Dentro de las posibilidades del analista, leer el código, a los fines de conocer el comportamiento de la funcionalidad para complementar con la herramienta de análisis (trace).

- Recuperar documentación (casos de uso, reglas de negocio, manuales de configuración, historias, defectos, manuales de usuarios) que permita conocer detalladamente la funcionalidad actual.

- Identificar la existencia de parámetros y permisos que condicionen el comportamiento de la funcionalidad.

- Cumplir con los pasos de carga defectos en la herramienta TFS considerando los estándares indicados anteriormente.

Mejoras en Testing

Detalle de pautas para testing, que tiendan a eliminar los defectos en el cliente final

Dependiendo del tipo de requerimiento (nuevo, modificación o error) el analista de prueba debería recorrer una serie de pasos diferentes.

1. **En el caso de un requerimiento nuevo no deberían saltearse las etapas de:**

- Indagar, conjuntamente con el analista funcional, sobre casos de similares características en la “Herramienta Solicitud de Requerimientos” y en el repositorio del producto para recuperar documentación y casos de pruebas anteriores que puedan ser reutilizados.

- En base al doc. de soporte propuesto y/o con el aporte de los tester de mayor experiencia, encarar la definición del alcance y técnicas de pruebas a utilizar.

- Cumplir con los pasos de carga de los escenarios de prueba en la herramienta TFS. Estos deben estar establecidos considerando, al menos, el comportamiento de cada parámetro o permiso impactado y cada criterio de aceptación incluido en la historia de usuario.

- Debe respetar que una historia contenga al menos un caso de prueba.

- La ejecución del caso de prueba debe realizarse, al menos, en la base de datos del cliente que realizó el pedido y otra base de datos, a los fines de contemplar aspectos de negocio diferentes ya que los clientes pueden o no, tener diferentes configuraciones, funcionalidades desarrolladas en distintos lenguajes de programación, etc. Recordemos que como política de la empresa, una solución para un cliente puede luego ser distribuida a todos los demás.

2. En el caso de un requerimiento de modificación deberían cumplirse las siguientes etapas:

- Explorar la funcionalidad vigente para interiorizarse sobre el comportamiento de la misma.
- En base al documento de soporte propuesto y/o con el aporte de los tester de mayor experiencia, encarar la definición del alcance y técnicas de pruebas a utilizar.
- Cumplir con los pasos de carga de los escenarios de prueba en la herramienta TFS. Estos deben estar establecidos considerando, al menos, el comportamiento de cada parámetro o permiso impactado y cada criterio de aceptación incluido en la historia de usuario o defecto de mejora.
- Respetar que una historia de usuario contenga al menos un caso de prueba. Si se trata de un defecto de mejora, el tester no tiene obligación de crear un caso de prueba, se consideraría suficiente con la especificación del “Resultado de verificación” en la sección “Detalle de corrección y verificación”, y con adjuntar documentación o capturas de imágenes como evidencia.
- La ejecución del caso de prueba debe realizarse, al menos, en la base de datos del cliente que realizó el pedido y otra base de datos, a los fines de contemplar aspectos de negocio diferentes ya que los clientes pueden o no, tener diferentes configuraciones, funcionalidades desarrolladas en distintos lenguajes de programación, etc. Recordemos que como política de la empresa, una solución para un cliente puede luego ser distribuida a todos los demás.

3. En el caso de un requerimiento de error se sugiere contemplar las siguientes etapas:

- Explorar la funcionalidad vigente para interiorizarse sobre el comportamiento de la misma y para verificar con precisión el error más allá de lo especificado en el defecto.
- Cumplir con los pasos de carga de los escenarios de prueba en la herramienta TFS si el defecto implica más de un escenario de prueba. Caso contrario, sería suficiente con la especificación del “Resultado de verificación” en la sección “Detalle de corrección y verificación”, y con adjuntar documentación o capturas de imágenes como evidencia.
- La ejecución del caso de prueba debe realizarse, al menos, en la base de datos del cliente que realizó el pedido y otra base de datos, a los fines de contemplar aspectos de negocio diferentes ya que los clientes pueden o no, tener diferentes configuraciones, funcionalidades desarrolladas en distintos lenguajes de programación, etc. Recordemos que como política de la empresa, una solución para un cliente puede luego ser distribuida a todos los demás.

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Anexo 2: Encuesta de Opinión y Entrevistas

Entrevista Guía al Product Owner

1. ¿Qué productos y servicios ofrece la empresa?¿ Cuántos años llevan en el negocio?
2. ¿Quiénes son sus clientes? ¿Aproximadamente que cantidad de clientes tienen?
3. ¿Cuáles son sus objetivos a largo plazo?
4. ¿Tienen competidores en el mercado?
5. ¿Cuáles son los requisitos sobresalientes con los que debe cumplir el producto? (Funcionales y No funcionales)
6. Quiénes son los usuarios?¿ Cuáles son las habilidades, conocimientos y ambiente o entorno tecnológico requeridos para utilizar el producto?
7. ¿Cuáles son las características más sobresalientes del producto que ofrece GobFactory?
8. Cuál es la metodología de desarrollo de software que se utiliza? Por qué?
9. ¿Considera que la metodología de trabajo actual del equipo de desarrollo está permitiendo lograr los objetivos de negocio de la empresa y del producto?
10. ¿Cuál es el grado de cohesión y acoplamiento entre los módulos del sistema?
11. ¿Cómo se gestiona la configuración del Software?
12. ¿Se realiza mantenimiento del software? En caso afirmativo: cómo es el procedimiento y con qué periodicidad.
13. Jerarquice 3 (tres) aspectos que usted considera más destacados
14. Indique los aspectos que, a su criterio, debieran ser optimizados

Entrevista Guía al Scrum Master

1. Cuál es la metodología de desarrollo de software que se utiliza? Por qué?
2. Cómo definirías tus responsabilidades principales como Scrum Master?
3. Difiere mucho de lo que a nivel teórico se espera de un Scrum Master? ¿Por qué?
4. Cuáles son los principales problemas o dificultades a los que te enfrentas en GobFactory?
5. Qué herramientas utilizas en tu tarea diaria?
6. Qué procesos de estimación se utilizan?
7. Cómo se mide la ejecución del proyecto
8. Qué herramientas de gestión de proyectos se utilizan? (Pert, Gantt, Histogramas, etc.)
9. Qué tipo de documentación se elabora? Quiénes son los encargados?
10. El proceso existente es acorde a las necesidades actuales del área de desarrollo? En caso negativo, que mejoras considera que podrían implementarse?
11. Quién define el proceso de trabajo?
12. Quién define las prácticas de la metodología de trabajo que se implementan o se eliminan por ser inapropiadas para las necesidades y ritmo de trabajo?
13. Considera que el equipo necesita reforzar el conocimiento sobre la metodología actual? Generalmente, cómo se implementan las capacitaciones.
14. Cuenta con colaboración externa sobre el proceso y la metodología utilizada?
15. Recibió suficiente capacitación sobre la metodología de trabajo antes de la implementación de sus prácticas?
16. Qué métricas se toman?
    - Métrica de requisitos: Requisitos de duración, integridad...
    - Métrica de producto: Líneas de código, métrica orientada al objeto, métrica de evaluación y diseño
    - Métrica de proceso: Evaluación y seguimiento del presupuesto, temporalización, recursos humanos.
17. Cómo se llevan a cabo la validación y la verificación

18. Qué nivel de madurez crees que GobFactory ha alcanzado respecto a la adopción de Agile?
19. Jerarquice 3 (tres) aspectos que usted considera más destacados
20. Indique los aspectos que, a su criterio, debieran ser optimizados

Encuestas de opinión aplicadas a diferentes roles de la empresa

Marcar con [X] la opción que corresponda.

Importante: Los datos suministrados son anónimos y confidenciales.

Sexo		Edad		Antigüedad en la Compañía		Departamento
Hombre		Menor de 25 años		2 años o menos		
Mujer		Entre 25-35 años		De 3 a 5 años		
		Entre 36-45		De 6 a 10 años		
		Entre 46-55		De 11 a 15 años		
		56 ó más		Más de 15 años		

COLABORACIÓN	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Cuando ingresé en la Compañía me sentí bienvenido						
Considero que existe un buen ambiente de trabajo						
Cuento con la colaboración de mis compañeros de departamento						
Cuento con la colaboración de las personas de otros departamentos						

COMUNICACIÓN	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Tengo disponible información sobre la organización y la evolución de WebFactory a nivel mundial						
Los comunicados internos me proporcionan información útil						
La comunicación sobre los resultados y marcha de la Compañía es clara y transparente						
Cuando ingresé en la Compañía recibí suficiente información sobre la misma						
Tengo disponible información sobre el producto y los servicios que ofrece WebFactory						
Conozco el Código Ético y de Conducta de WebFactory						
Al unirme a la Compañía, recibí suficiente información sobre el área donde trabajo y la función que realizo						
La comunicación interna en WebFactory es una actividad permanente y planificada						

CARRERA PROFESIONAL	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Las promociones internas se realizan de manera justa						
Creo que tengo la oportunidad de desarrollarme profesionalmente en WebFactory						
Pienso que si desempeño bien mi trabajo, tengo posibilidad de hacer carrera en WebFactory						
Tengo autonomía para llevar a cabo mi trabajo						
Tengo disponible información sobre los puestos vacantes en la Compañía						
Considero adecuados los criterios de evaluación de mi desempeño en WebFactory						
Al asumir una nueva posición en WebFactory , mi responsable me informa de manera clara sobre las funciones y responsabilidades del puesto de trabajo						

LIDERAZGO	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Mi responsable me proporciona periódicamente información sobre mi desempeño						
Mi responsable escucha mis opiniones y me hace partícipe de las decisiones						
Mi responsable hace un seguimiento de mi Plan de Desarrollo Individual						
Mi responsable se preocupa por conocer mis necesidades e intereses						
Puedo tomar decisiones propias sin necesidad de consultar con mi jefe						
Mi responsable es claro y específico cuando define mis objetivos de trabajo o los del departamento						
Mi responsable se preocupa por mantener un buen clima en el equipo						
Mi responsable es un referente en la Compañía						
Mi responsable respeta las diferencias de cultura, sexo, religión…						
Mi responsable me trata justamente y evita cualquier tipo de favoritismos						
Mantengo una buena relación con mi responsable						
Mi responsable se preocupa por transmitir los valores, misión y objetivos de WebFactory						
Mi responsable me felicita cuando realizo bien mi trabajo						

ORIENTACIÓN AL CLIENTE	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Estoy dispuesto a realizar un esfuerzo extra para satisfacer a mi cliente interno o externo						
Los procesos y procedimientos de trabajo en WebFactory me orientan hacia el cliente interno/externo						

SATISFACCIÓN EN EL PUESTO DE TRABAJO	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Tengo la oportunidad de proponer nuevos proyectos o nuevas formas de realizar el trabajo						
Mi capacidad profesional está de acuerdo a las tareas y responsabilidades asignadas						
Mi trabajo es reconocido y valorado						
WebFactory me da/ofrece la oportunidad de trabajar en proyectos/actividades que suponen nuevos retos						
Tengo claro cuáles son mis tareas y responsabilidades						
Tengo la información que necesito para realizar mi trabajo con excelencia						
Mi trabajo me ofrece retos y la oportunidad de seguir mejorando						
Conozco como mi trabajo contribuye a conseguir los resultados de mi departamento						

CUESTIONES GENERALES	Nunca	Casi nunca	A veces	Casi siempre	Siempre	N/A
Considero adecuados e importantes los proyectos de Responsabilidad Social implementados en mi país						
Considero que planes de acción derivados de la anterior encuesta de satisfacción se han cumplido en su mayoría						
Desde mi entrada en la Compañía, pienso que WebFactory se ha ido transformando en un lugar mejor para trabajar						
Me siento partícipe del proyecto de WebFactory						
Actualmente estoy satisfecho con mi trabajo en WebFactory						
Estoy satisfecho con los beneficios sociales que me ofrece WebFactory						
Las personas con las que me relaciono en WebFactory actúan con respeto y de manera ética						
Considero que los Valores de WebFactory reflejan el estilo de trabajo que existe en la Compañía						
Me siento orgulloso de trabajar para WebFactory						
Estoy satisfecho con las actividades deportivas que apoya WebFactory						
Recibo información sobre los elementos que componen mi salario (salario base más beneficios)						
Pienso que WebFactory es un buen lugar para trabajar y me gustaría continuar trabajando aquí						
WebFactory innova y mejora continuamente para ser el líder de su sector						
Recomiendo WebFactory como un lugar donde trabajar						

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Glosario

Glosario

Backlog del Producto: es la lista maestra de toda la funcionalidad deseada del producto. Esta lista expresa los requerimientos del Dueño del Producto en su lenguaje y se encuentra ordenada de acuerdo a la prioridad de cada ítem.

Backlog del Sprint: Define el trabajo de un sprint, representado por un conjunto de tareas que deben completarse para cumplir los objetivos del sprint, y por un conjunto de elementos seleccionados del backlog del producto.

Defecto: es el desperfecto en un componente o sistema que puede ocasionar que dicho componente o sistema falle en su ejecución. En GobFactory, el término "defecto" también representa un elemento de trabajo de la herramienta Team Fundation Server (TFS).

Elicitación de Requerimientos: es la etapa de recolección de datos e información en el proceso de desarrollo de requerimientos. En este momento se define la visión del producto, el alcance y sus limitaciones.

Especificación de Requerimientos: es el proceso de grabado o el registro de los requerimientos en una o más formas, incluyendo el lenguaje natural y formal, representaciones simbólicas o gráficas [Tuffley:2005].

Estimaciones: Cálculo del esfuerzo que se prevé necesario para desarrollar una funcionalidad. Las estimaciones se pueden calcular en unidades relativas (puntos de función) o en unidades absolutas (tiempo teórico).

Proceso: secuencia de pasos encadenados con el fin de construir un producto u ofrecer un servicio.

Proceso de Desarrollo de Software: es la aplicación de un método para el desarrollo de software por parte de las personas con el entrenamiento adecuado.

Product Backlog Ítem (Elemento del Backlog del Producto): En Scrum, el PBI es una unidad de trabajo lo suficientemente pequeña para que el equipo pueda completarla en un sprint. Los elementos del backlog se descomponen en una o más tareas.

Product Owner (Dueño del Producto): es la persona encargada de definir las funcionalidades, priorizar el Backlog del Producto y de aceptar o rechazar los resultados de un Sprint.

Requerimiento: es una capacidad, atributo o restricción de diseño del software que provee valor o es necesario para algún interesado (stakeholder). Estos requerimientos pueden ser de sistema, de software o de hardware.

SCRUM: es un enfoque ágil para la gestión de un proyecto. Más que una metodología o proceso, es un Marco de Trabajo.

Scrum Master (Facilitador): es la persona encargada de asegurar que el equipo sea completamente funcional y productivo, de promover la cooperación entre todos los roles y de garantizar que el proceso sea cumplimentado.

Sprint: Una iteración de trabajo durante la cual se implementa un incremento de la funcionalidad del producto. Usualmente, una iteración dura 30 días.

Stakeholder: Son partes con un interés en el producto bajo desarrollo y/o en el proceso de Scrum.

Story Point (Punto de Historia): un SP representa una jornada de trabajo de un miembro del equipo con dedicación exclusiva.

Test Case (Caso de Prueba): un conjunto de condiciones o variables bajo las cuáles un analista determinará si una aplicación, un sistema software, o una característica de éstos es parcial o completamente satisfactoria.

Tiempo real o tiempo de trabajo: Tiempo efectivo para realizar un trabajo. Se suele medir en horas o días.

Tiempo teórico o tiempo de tarea: Tiempo que sería necesario para realizar un trabajo en "condiciones ideales": si no se produjera ninguna interrupción, llamadas telefónicas, descansos, reuniones, etc.

User Story (Historias de Usuarios): es una descripción simple y corta de una funcionalidad del sistema vista desde la perspectiva de la persona que desea dicha nueva funcionalidad, generalmente un usuario o cliente del sistema. Las historias favorecen el enfoque de discutir las nuevas funcionalidades más que escribir acerca de ellas.

Velocidad absoluta: Cantidad de producto construido en un sprint. Se expresa en la misma unidad en la que se realizan las estimaciones (puntos de función, horas o días reales o teóricos).

Velocidad relativa: Cantidad de producto construido en una unidad de tiempo de trabajo.
P. ej.: puntos de función / semana de trabajo real; o horas teóricas / día de trabajo real…

Trabajo Final de Grado
Caso de Reingeniería de un Proceso de Desarrollo de Software

Bibliografía

Bibliografía

- El Proceso Unificado de Desarrollo de Software (2000) - Ivar Jacobson, Grady Booch, James Rumbaugh.
- Calidad en el Desarrollo y Mantenimiento de Software (2003) - Mario Piattini Velthuis, Félix García Rubio
- Fábricas de Software: Experiencias, Tecnologías y Organizaciones (2010) - Mario Piattini Velthuis, Javier Garzás.
- Calidad en Sistemas Informáticos (2006) - Mario Piattini Velthuis, Félix García Rubio, Ismael Caballero.
- Medición y Estimación del Software (2008) - Mario Piattini Velthuis, Félix Garcia Rubio, Ismael Caballero, Javier Garzás. Marcela Genero.
- Manifesto for Agile Software Development (2001) - Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, entre otros.
- Procesos Ágiles para el Desarrollo de Aplicaciones Web - Extraído del artículo publicado por Paloma Cáceres, Esperanza Marcos.
- Gestión de Proyectos de Desarrollo de Software - Marcela Varas
- Metodología de la Investigación. Sampieri, R. H. McGraw-Hill Companies, 2006.
- Ingeniería del Software - Un Enfoque Práctico - Roger Pressman McGraw-Hill, 2005
- Kanban y Scrum - Obteniendo lo Mejor de Ambos - Henrik Kniberg & Mattias Skarin
- Scrum y XP Desde las Trincheras - Henrik Kniberg
- Proyectos Ágiles con Scrum - Alaimo, Diego Martín
- Flexibilidad con Scrum - Palacio, Juan
- Revista de Psicopatología y Psicología Clínica 2003 – Vol. 8 Nro. 2 – PP.153 - 172

Sitios Webs Consultados:

http://www.agilemanifesto.org/iso/es/
http://www.javiergarzas.com/2013/11/entrevista-henrik-kniberg.html
http://www.javiergarzas.com/2014/01/entrevista-jeff-sutherland.html
http://alistair.cockburn.us/
http://www.i2btech.com/
https://josepablosarco.wordpress.com
http://www.pmoinformatica.com
http://www.iso.org/iso/home.html
http://www.kleer.la/
http://www.isecauditors.com
https://proyectosagiles.org
https://manuelguerrerocano.wordpress.com
https://www.ieee.org
https://lineabase2.wordpress.com/category/metodologias/