Séptima Conferencia de Directores de Tecnología de Información, TICAL 2017 Gestión de las TICs para la Investigación y la Colaboración, San José, del III al V de julio de 2017

Gestión de Requerimientos a través del Modelo de Fábrica de Software aplicando Metodologías Ágiles

Lisset Alexandra Neyra, María Paula Espinoza, María Patricia Samaniego

Dirección de Operaciones,
Universidad Técnica Particular de Loja, Loja, Ecuador
laneyra@utpl.edu.ec, mpespinoza@utpl.edu.ec, mpsamaniego@utpl.edu.ec,

Resumen. La Universidad Técnica Particular de Loja (UTPL) es una institución de educación superior que oferta cada año carreras de pregrado y postgrado en modalidad presencial y a distancia, así como cursos cortos a nivel nacional e internacional (Nueva York, Roma y Madrid). Actualmente cuenta con aproximadamente 30000 estudiantes en sus dos modalidades. Su modelo de gestión, tiene como disciplina base la mejora continua y la gestión de la calidad, elementos que se reflejan en sus líneas estratégicas y que define los proyectos estratégicos de la institución, orientado al bienestar de los estudiantes y la sociedad en general. Tanto los proyectos como la operación propia de sus procesos misionales, generan un sinnúmero de requerimientos de mejora o implementación a nivel de procesos o sistemas, los mismos que deben ser priorizados y atendidos con una visión integral, para garantizar una correcta implementación y alineación con las necesidades de la organización.

Palabras Clave: Metodologías ágiles, gestión de requerimientos, desarrollo de software.

Eje Temático: Soluciones TIC para la Gestión.

1. Introducción:

La Universidad Técnica Particular de Loja (UTPL) es una institución de educación superior que brinda oferta académica de pregrado, postgrado y cursos cortos a través de las Modalidades de Estudios: Presencial y Abierta y a Distancia, en esta última cuenta con aproximadamente 30000 estudiantes, 82 centros de atención distribuidos en el Ecuador y 3 centros internacionales en Madrid, Roma y Nueva York [1], razón por la cual, la plataforma tecnológica se convierte en un apoyo primordial de su gestión. Además, la línea estratégica de Liderazgo y Excelencia, determina claramente la necesidad de institucionalizar los procesos y la estructura universitaria así como alcanzar una cultura de calidad [2] .

Su Modelo de Gestión Estratégica basado en fundamentos permanentes (modelo educativo, estatuto y macropolíticas) y conyunturales (LOES, plan nacional del buen vivir, matriz productiva, etc.) incorpora un conjunto de disciplinas como el modelo PDCA, el Sistema de Gestión de Calidad y la Gestión por Proyectos y junto con ello la necesidad de gestionar de forma estratégica los requerimientos de tecnología que garanticen que los desarrollos tecnológicos estén fuertemente alineados a la estrategia de la organización, pues como señala Fernández & Llorens, “la planificación tecnológica en la universidad debe formar parte de la planificación global, pues tienen un carácter estratégico y horizontal” [3].

Es así que la gestión de requerimientos constituyen una pieza fundamental de los proyectos de desarrollo de software y marcan el punto de partida para actividades de planificación a través de las estimaciones de alcance, tiempo y costo, así como la definición de recursos necesarios y la elaboración del cronograma, uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo.

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos [4].

Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad [4].

Actualmente las metodologías agiles están revolucionando la manera de producir el software [4], sin embargo un error que se ha cometido durante muchos años ha sido pensar que las metodologías ágiles, sin adaptación al caso concreto y real sobre el que operan, eran la mejor opción para todo tipo de proyectos [5], por lo que cada proyecto, empresa, producto, línea de negocio, etc. necesita de una adecuación específica de estas metodologías.

2. Problema identificado y solución planteada

La Unidad de Gestión de Tecnologías de Información (UGTI) tiene como misión entregar valor a la comunidad universitaria, a través de la operación tecnológica basada en estándares de calidad y de la implementación de proyectos de innovación, alineando sus esfuerzos hacia el cumplimiento de la misión y objetivos institucionales.

Si bien, el principal cliente externo de la unidad son los estudiantes, por otra parte cada una de las áreas de la universidad se constituyen en clientes internos de UGTI, a través de los requerimientos de tecnología o desarrollo de software, que les permite soportar su operatividad, cumpliendo la función de sus procesos principales. Estos requerimientos son gestionados a través de la política institucional[1], que norma el proceso de gestión de requerimientos integral, obteniendo como resultado aquellos

---

1 https://procuraduria.utpl.edu.ec/sitios/documentos/NormativasPublicas/ORGANIZACION%20INSTITUCIONAL/GESTIÓN%20ADMINISTRATIVA%20FINANCIERA/Política%20de%20gestión%20de%20requerimientos%20para%20mejora%20de%20software.pdf

proyectos que demandan de intervención tecnológica y que han sido priorizados a nivel institucional orientados a la atención de las líneas estratégicas de UTPL.

Estos requerimientos de software se incrementan en determinadas temporadas en función de la planificación interna, y deben ser atendidos de forma oportuna lo que demanda que el equipo de tecnología implemente estrategias efectivas para la gestión de los mismos.

Una de éstas estrategias es contar con un modelo de Fábrica de Software a través de proveedores (aliados estratégicos), que permitan dar una respuesta rápida y oportuna a los requerimientos, tanto para los clientes externos; es decir los estudiantes como los clientes internos, que puede ser cualquier dependencia interna.

Con este antecedente se realiza la implementación del modelo de Fábrica de Software en la universidad, a través de las siguientes etapas descritas de forma general:

1. Se inició con un proceso de búsqueda de proveedores con experiencia en la prestación del servicio de Fábrica de Software.
2. Definido el proveedor, se inició con un proceso de capacitación e involucramiento en las metodologías ágiles definidas, con los principales funcionales y el equipo técnico de la Unidad de Gestión de Tecnologías de la Información, con el fin de conocer el equipo de fábrica, y las características relevantes de cada metodología ágil, esto se lo llevó a través de talleres de trabajo.
3. Se realizó el proceso de transición al equipo de Fábrica de Software del código fuente de los sistemas definidos en el alcance del servicio y se realizó la transferencia de conocimiento del manejo de las aplicaciones y arquitectura definida de los sistemas.
4. Se entregó al equipo de la Fábrica de Software los primeros requerimientos de desarrollo de software, inicialmente el equipo de la UGTI es el frente con los clientes, y posterior se realizó el cambio, donde el equipo de la Fábrica de Software es el frente con los clientes internos de la universidad y el equipo de UGTI es apoyo en el proceso de implementación.
5. Se realizó revisiones periódicas sobre el modelo de trabajo y las metodologías ágiles de desarrollo aplicadas, a través de métricas preestablecidas, lo que permitió detectar puntos de mejora y corrección.

Esta implementación del modelo de Fábrica de Software para la atención de las necesidades de desarrollo de software de la Universidad Técnica Particular de Loja inició en enero del 2015 y se mantiene hasta la fecha. Durante todo este proceso se ha modificado la aplicación de las metodologías ágiles como SCRUM y KANBAN, adecuándolas a obtener los resultados esperados en la universidad.

3. Terminología:

Workaround: solución temporal que, si bien no soluciona el problema, sí que permite restaurar el Servicio cuanto antes [6].

SLA (Acuerdos de Nivel de Servicio): Es un acuerdo entre el proveedor de TI y un cliente. Un acuerdo de nivele de servicio describe los servicios de TI, documenta los objetivos de nivel de servicio, y especifica las responsabilidades del proveedor de

servicios de TI y el cliente. Un acuerdo único puede cubrir múltiples servicios de TI o varios clientes [7].

Metodologías Ágiles de Desarrollo: Las metodologías ágiles se centran en el factor humano y el producto software; es decir, ellas le dan mayor valor al individuo, a la colaboración del cliente y al desarrollo incremental del software con iteraciones muy cortas [8].

Product Backlog: La Lista de Producto es una lista ordenada de todo lo que podría ser necesario en el producto, y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación [9].

Historias de Usuario: técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas [8].

4. Modelo de Fábrica

4.1. Alcance de fábrica

En la Fig.1. se visualiza los componentes que son parte del alcance definido para la prestación del servicio a la Universidad Técnica Particular de Loja en el modelo de Fábrica de Software .

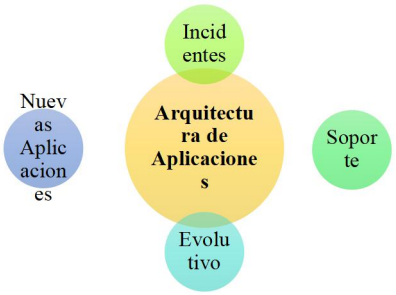


Fig. 1. Alcance de la Fábrica de Software.

Incidentes: se define como incidente a "interrupción no programada de un servicio que debe ser restablecido en el menor tiempo posible", por ello la Fábrica de Software busca dar una solución temprana a los incidentes reportados por los usuarios finales de las aplicaciones. Cuando surge un incidente que requiere modificación del sistema, este se escala al frente de soporte, sin embargo previamente se propone un workaround, con el fin de dar una solución temporal y evitar la continuidad del incidente reportado.

Soporte: El objetivo principal de este frente es agilizar los modelos de desarrollo, a través de cambios que no necesariamente son nuevas funcionalidades, sino que son

cambios menores pero permiten la continuidad del negocio. Existen los siguientes tipos de mantenimientos:

- **Mantenimiento Correctivo:** Se realizan todos los ajustes necesarios para la resolución de los incidentes que requieren la intervención, desde el punto de vista programático), de la aplicación para su correcto funcionamiento [10].
- **Mantenimiento Preventivo:** Son todos los cambios a las aplicaciones que apuntan a mejorar el sistema para evitar su degradación. Dentro de algunas de las tareas a realizar en este frente se encuentran tales como: Mejoras en el rendimiento de procesos, disminución de la deuda técnica de la aplicación, entre otros [10].
- **Mantenimiento Adaptativo:** Es la modificación de la aplicación realizada después de la entrega para permitir que la misma siga pudiéndose utilizar en un entorno diferente [10].
- **Mantenimiento de Información:** Para las actividades del día a día de los usuarios, se requiere constantemente extracción y/o ajustes sobre la información de los sistemas para dar respuesta a las solicitudes del negocio [10].

Evolutivo: El objetivo principal de este frente es de evolucionar y/o adicionar nuevas funcionalidades que son necesarias para que el negocio evolucione. Se clasifica en dos tipos de solicitudes.

- **Express:** Los requerimientos Evolutivos express son aquellos requerimientos que por la situación de urgencia (legal, Negocio, etc) requieren estar rápidamente en producción y que sus dimensiones son considerables pequeños [10].
- **Evolutivos:** Son los requerimientos normales que buscan evolucionar la aplicación agregando funcionalidades o componentes que permitan generar valor a los usuarios y al negocio [10].

Nuevas aplicaciones: Con este frente se pretende dar un valor agregado a la Universidad ya que el objetivo principal radica en el desarrollo de nuevas soluciones informáticas que están por fuera del dominio inicial planteado para la fábrica y permitan cumplir con los objetivos estratégicos de la universidad.

Arquitectura de Aplicaciones: Este es un componente transversal que articula que requerimientos estén alineados a las políticas y estándares establecidos por la universidad, teniendo en cuenta las nuevas tecnologías.

Lo que se busca con esta capacidad es que se mantenga una gobernabilidad sobre los diferentes desarrollos que se inicien en la fábrica garantizando la homogeneidad arquitectónica establecida o estableciendo los planes para que se lleve a cabo. [10].

4.2. Enfoque Metodológico fábrica

La metodología de inicio de una fábrica consiste en las etapas que se muestran en la Fig.2.

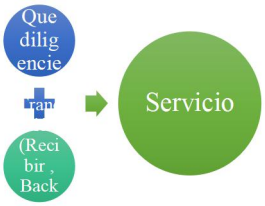


Fig. 2. Enfoque metodológico de fábrica.

4.2.1. Que Diligencie

Esta etapa gestiona los primeros acercamientos con la universidad, donde se determina el equipo de trabajo, la comunicación entre equipos y talleres de adopción de las metodologías y herramientas ágiles a aplicar. Esta etapa es importante ya que es el punto de partida de la implementación del modelo de Fábrica de Software.

4.2.2. Transición

En esta etapa el principal objetivo es recibir las aplicaciones y transferir los artefactos y conocimiento a la Fábrica de Software. Dentro de este proceso de transición se tienen dos etapas que se describen a continuación:

- **Recibir:** Esta etapa inicia con el empalme, el cual tiene como objetivo recibir las aplicaciones seleccionadas, recibir el código fuente, implementar sus ambientes de desarrollo en la fábrica -si es el caso- y realizar todas las actividades de transferencia y configuraciones requeridas con el objetivo de prestar los servicios expuestos en esta propuesta. A su vez se establecen las condiciones de infraestructura, ambientes, conectividad y todos los elementos de plataforma que sean necesarios para poder prestar el servicio, entre ellas tenemos: contexto del negocio, arquitectura, base de datos, integración, código fuente y montaje de ambientes [10].
- **Back Y Front:** Inicialmente la Fábrica de Software comienza con la atención de las aplicaciones en un esquema donde UTPL estará en el FRONT y la Fábrica de Software en el BACK. Para esta etapa la responsabilidad del servicio es de UTPL Posteriormente la Fábrica de Software continúa con la atención de las aplicaciones en un esquema donde la Fábrica de Software estará en el FRONT y UTPL en el BACK para soportar que todas las actividades se hagan de manera adecuada y se refuerce la transferencia de conocimiento. La responsabilidad del servicio es de UTPL. [10]

Se definen los entregables que la universidad debe entregar a la Fábrica de Software los cuales son: código fuente, componentes empleados por las aplicaciones, diseño de la solución, especificación de requisitos de software, guion de instalación en ambientes de desarrollo y pruebas (incluyendo un guion que permita probar la instalación), guías para el despliegue a otros ambientes de las aplicaciones,

documentación sobre errores conocidos para cada aplicación, y peticiones o Backlog (priorizadas).

Por otra parte los entregables de la Fábrica de Software para UTPL son: Acta de recepción y Ficha técnica de cada aplicación recibida.

Es importante mencionar que durante esta etapa se adecuaran los equipos de trabajo, procedimientos con la universidad, con el fin de ajustar los elementos necesarios para el cumplimiento de los Acuerdos de Nivel de Servicio (ANS) y no se aplicarán penalizaciones por incumplimiento de los Acuerdos de Nivel de Servicio (ANS).

4.2.3. Servicio

Esta etapa engloba la ejecución del alcance definido de la Fábrica de Software como son: incidentes, evolutivo, nuevas aplicaciones y soporte. También se propone inicialmente que metodologías ágiles se aplicaran como se muestra en la Fig.3.

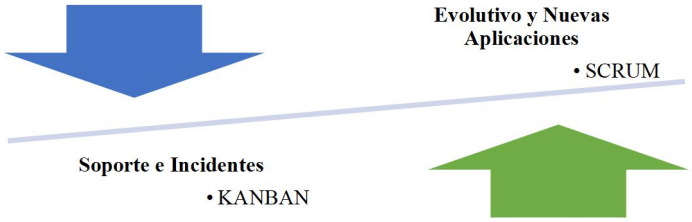


Fig. 3. Aplicación de metodologías propuesta por la Fábrica de Software

SCRUM: es un marco de trabajo o Framework ampliamente aceptada para el desarrollo ágil de productos de software fundamentada en la teoría del control empírico de procesos, con enfoque iterativo e incremental orientado a minimizar los riesgos que se presenten [10].

Acorde con la teoría del control empírico de procesos, SCRUM se sostiene por tres pilares fundamentales:

- **Transparencia**: asegura que cada aspecto del proceso que puede afectar el resultado es conocido por todos los involucrados en el logro de ese resultado. Esto implica que no existen suposiciones (“creo que está hecho”) sin hechos (“eso ya está hecho”) [10].
- **Inspección**: Se debe mantener un control a partir de la inspección frecuente de los diversos aspectos de un proceso de forma tal que se puedan detectar a tiempo variaciones que afecten el resultado o que sea inaceptables. Esto facilita una respuesta rápida ante el cambio minimizando el riesgo que éste cambio origine [10].
- **Adaptación**: A partir de la inspección, el producto se debe adaptar rápidamente a los cambios detectados minimizando el riesgo y la desviación de esfuerzo [10].

En base a estos pilares, SCRUM implementa la metodología de ciclos iterativos cortos con la entrega de productos funcionales al finalizar cada iteración.

KANBAN: para la atención de incidentes se propone utilizar la herramienta KANBAN lo que permite solucionar de manera oportuna los incidentes que se presenten. Existen tres reglas principales de KANBAN las cuales se detallan a continuación:

- **Visualizar el trabajo en KANBAN y las fases del ciclo de producción, o flujo de trabajo:** Al igual que SCRUM, KANBAN se basan en el desarrollo incremental, dividiendo el trabajo en partes. Una de las principales aportaciones es que utiliza técnicas visuales para ver la situación de cada tarea. El trabajo se divide en partes, normalmente cada una de esas partes se escribe en una nota adhesiva y se pega en un tablero (el tablero KANBAN). Las notas adhesivas suelen tener información variada, si bien, aparte de la descripción, debieran tener la estimación de la duración de la tarea. El tablero tiene tantas columnas como estados por los que puede pasar la tarea (ejemplo, recibido, en atención, terminado, etc.) [10].
    - **Determinar el límite de "trabajo en curso":** Una de las principales ideas del KANBAN es que el trabajo en curso (Work In Progress o WIP) debería estar limitado, es decir, que el número máximo de tareas que se pueden realizar en cada fase debe ser algo conocido. En KANBAN se debe definir cuantas tareas como máximo puede realizarse en cada fase del ciclo de trabajo (ejemplo, como máximo 5 incidentes "en atención", como máximo 1 en pruebas, etc.), a ese número de tareas se le llama límite del "work in progress". A esto se adiciona otra idea tan razonable como que para empezar con una nueva tarea alguna otra tarea previa debe haber finalizado. La idea es centrarse en cerrar tareas y no en comenzar tareas. Por ello limitar el WIP impide empezar cosas hasta que se han cerrado aquellas en las que se está ya trabajando. Los límites dependen del tamaño y capacidad del equipo de solución de incidentes [10].
    - **Medir el tiempo en completar una tarea:** El tiempo que se tarda en terminar cada tarea se debe medir, a ese tiempo se le llama "lead time". El "lead time" cuenta desde que se hace una petición hasta que se hace la entrega. Aunque la métrica más conocida del KANBAN es el "lead time", normalmente se suele utilizar también otra métrica importante: el "cycle time". El "cycle time" mide desde que el trabajo sobre una tarea comienza hasta que termina. Si con el "lead time" se mide lo que ven los clientes, lo que esperan, y con el "cycle time" se mide más el rendimiento del proceso [10].

Se puede resumir que KANBAN permite visualizar el flujo de trabajo, limitar el WIP (Work in Progress, trabajo en curso) y medir el lead time y cycle time.

4.3. Modelo General de Atención Del Servicio

Soporte e Incidentes: En la Fig.4. se muestra el proceso para la atención de requerimientos de soporte e incidentes.

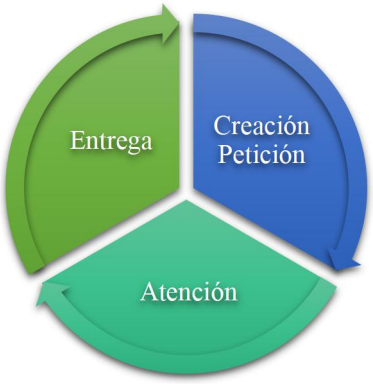


Fig. 4. Esquema de soporte e incidentes.

Evolutivo y Nuevas Aplicaciones: La Fig.5. resume a nivel macro, las actividades y procesos involucrados en el ciclo de trabajo de cada una de las peticiones de desarrollo de software, mantenimiento, evolución y entre otras todas aquellas actividades, que para su ejecución se convertirán en una orden de servicios, la cual definirá alcance, entregables y se estructurarán como proyectos [10].

Fig. 5. Ciclo de trabajo de peticiones de desarrollo de software.

4.4. Acuerdos de Niveles de servicio

En la Tabla 1. se presenta la clasificación de nivel de severidad de los incidentes de soporte de segundo nivel con los que se propone el servicio.

Tabla 1. Descripción de Tiempos de resolución de incidentes [10].

Nivel de Severidad	Definición	Tiempo de Solución
A –	Problemas críticos que ponen en riesgo el negocio y	6 horas

Incidente Critico	no se identifica solución al problema. Incluye interrupción del sistema o pérdida de información. Problemas que afecten a los usuarios directamente y se debe dar atención inmediata.	
B – Incidente Urgente	No críticos pero importantes, son problemas de producción que no interrumpen la programación operativa y no están asociados con un alto riesgo.	12 Horas
C – Incidente Importante	Problemas que no afectan la operación normal del sistema pero que deben ser solucionados. Incluyen defectos menores o malos funcionamientos que afectan de manera menor la comodidad de los usuarios con el sistema.	24 Horas

5. Esquema de Ejecución

En la Fig. 6. se muestra el esquema de trabajo que actualmente se ejecuta con el equipo de fábrica de software, donde de forma general existen 6 etapas por las que todo requerimiento o proyecto de desarrollo de software atraviesa.

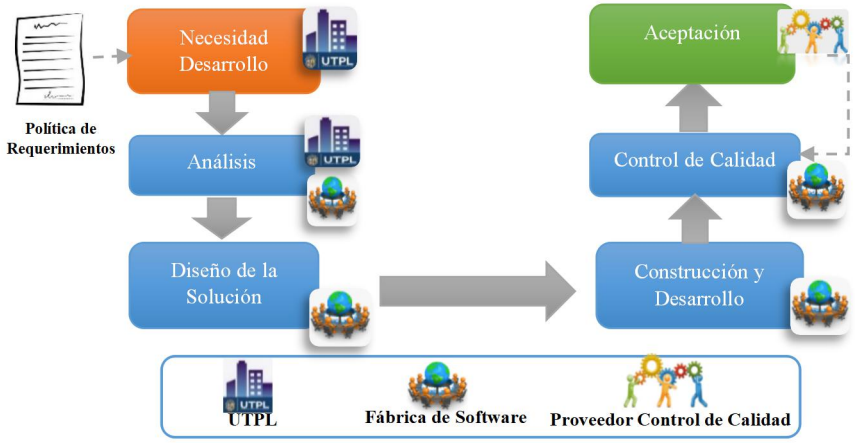


Fig. 6. Esquema de ejecución con el equipo de fábrica de software.

En este esquema existen 3 actores principales: UTPL, Fábrica de Software y proveedor de control de calidad, a continuación se describe cada uno de ellos:

- **UTPL:** se refiere al equipo de la Unidad de Gestión de Tecnologías de la Información y a los funcionales internos solicitantes.
- **Fábrica de Software:** Se refiere al equipo de la fábrica de software conformado por: el gerente del proyecto, el SCRUM Master, arquitecto, ingenieros de desarrollo y líder de QA.
- **Proveedor de Control de Calidad:** Se refiere al equipo de control de calidad gestionado por la Unidad de Gestión de Tecnologías de la

Información que ejecuta el proceso de aceptación de los requerimientos entregados por la fábrica de software en ambientes que son réplicas de producción denominado preproducción, y está conformado por : Líder de QA y testers.

5.1. Etapas

En la presente sección se detallan cada una de las etapas descritas en la Fig. 6. que es el esquema de trabajo actual con el equipo de fábrica de software.

a) Necesidad de Desarrollo: Los requerimientos o proyectos son gestionados a través de la política institucional de requerimientos, donde existe un proceso de solicitud por el funcional y esta solicitud es aprobada bajo ciertos criterios y priorizada por el comité estratégico, sin embrago también existen requerimientos emergentes o de mantenimiento, el cual es priorizado por el comité técnico; en cualquiera de estos casos se notifica a la gerencia de la Unidad de Gestión de Tecnologías de Información y los requerimientos o proyectos son registrados en el listado de requerimientos general de UGTI.

Existe una persona del equipo de Soluciones de Negocios que es una sección de UGTI, que gestiona al equipo de la fábrica de software notificando los requerimientos priorizados y en conjunto con el SCRUM Master se coordina y se asigna un analista de la fábrica de software.

b) Análisis: Priorizado el requerimiento o proyecto, se asigna una persona del equipo de fábrica de software así como un analista técnico de la Unidad de Gestión de Tecnologías de Información el cual será de apoyo técnico para la fábrica de software.

De acuerdo a la política institucional se ha definido un concepto de triada donde todos los requerimientos o proyectos que son priorizados por comité estratégico deben pasar por un equipo denominado triada, el cual es conformado por el funcional o funcionales, un analista de procesos y un analista técnico, donde se realizan reuniones para modelar y realizar la especificación de negocio de la necesidad; esto genera un documento de especificación de requerimientos de negocio. En algunas ocasiones el equipo de fábrica ha sido vinculado desde esta etapa, de esta forma es involucrado directamente con el funcional para conocer los requerimientos de negocio.

Para los requerimientos emergentes o de mantenimiento el análisis se realiza con el equipo técnico de la Unidad de Gestión de Tecnologías de Información y el equipo de fábrica de software, con ello se pretende conocer la necesidad y verificar la factibilidad del requerimiento.

c) Diseño de la Solución: En esta etapa se realizan reuniones con el equipo de Arquitectura de la Unidad de Gestión de Tecnologías de la Información, el cual aprueba la solución y el diseño definido por el equipo de fábrica de software que en conjunto con el arquitecto de fábrica buscan cumplir con los estándares y lineamientos arquitectónicos definidos por la universidad, con el fin de brindar soluciones idóneas para el funcional y que técnicamente sea implementado de forma correcta.

Con la aprobación del equipo de Arquitectura de UGTI y el analista técnico designado de UGTI, se procede a la elaboración de historias de usuarios, donde se plasma las necesidades y los criterios de aceptación, de una forma sencilla y fácil de entender, y finalmente se emite una estimación del esfuerzo por cada historia de usuario.

d) Construcción y Desarrollo: En esta etapa el gestor del equipo de la fábrica de software con el SCRUM Master definen cuantos desarrolladores van a ser asignados en base a la capacidad y fechas comprometidas de entrega. Es importante mencionar que en conjunto se define que metodología ágil de desarrollo se va a aplicar, esto dependerá de algunos criterios y en base a la experiencia con el modelo de fábrica de software se tiene una tendencia, la cual nos dice que para requerimientos emergentes o de mantenimiento se suele definir KANBAN, mientras que para proyectos cuya duración es considerable se suele definir SCRUM.

Cada una de las metodologías ágiles de desarrollo antes mencionadas han sido adecuadas a la realidad de la universidad, lo que nos ha dado muy buenos resultados.

e) Control de Calidad: Una vez desarrolladas cada historia de usuario el líder de QA de la fábrica de software, realiza el proceso de pruebas funcionales en un ambiente de calidad que UGTI ha destinado para su uso.

f) Aceptación: Cuando se certifica las historias de usuario son notificadas al gestor de la fábrica de Software y estas son desplegadas en los ambientes de preproducción para que el proveedor de control de calidad realice el proceso de aceptación de los requerimientos, este proceso consiste en verificar que lo solicitado ha sido implementado y que en un ambiente replica de producción no presente ningún bug.

5.2. Equipo de trabajo

En esta sección se describen todos los equipos de trabajo que están involucrados para la implementación de los requerimientos de software y cómo interactúan entre sí.

5.2.1. Estructura de la Unidad de Gestión de Tecnologías de la Información.

En la Fig. 7, se detalla la estructura de la Unidad de Gestión de Tecnología de la Información.

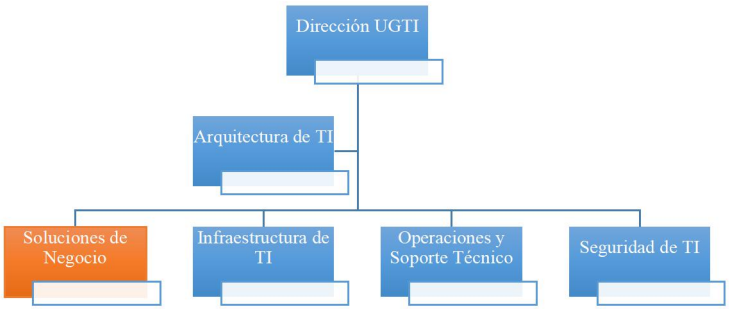


Fig. 7. Estructura UGTI.

A continuación se detallan cada una de las funciones de forma general de cada sección de la Unidad de Gestión de Tecnologías de la Información.

- **Soluciones de Negocios:** Este equipo se encarga de entender los requerimientos de nuestra universidad con la finalidad de identificar y planificar la implementación de soluciones tecnológicas que incrementen los niveles de productividad y gestión. Innovar a través del uso de nuevas tecnologías., operar y soportar la infraestructura de operación. Es importante mencionar que este equipo es que interactúa directamente con el equipo de fábrica de software.
- **Infraestructura de TI:** Gestionan la infraestructura de hardware, servidores y redes necesarias para soportar los servicios de tecnología que soportan los procesos de la universidad.
- **Operaciones y Soporte Técnico:** Su función es de soportar la Infraestructura Tecnológica y la operación de las aplicaciones dentro de los Niveles de Servicio definidos con las diferentes áreas de negocio.
- **Seguridad de TI:** Se encargan de Identificar, evaluar y monitorear los riesgos de tecnología y mantener ciclos de mejora continua que eleven los niveles de servicios de tecnología.

5.2.2. Equipo de Soluciones de Negocios

El equipo de la sección Soluciones de Negocios está conformado los siguientes perfiles que se muestran en la Fig.8.

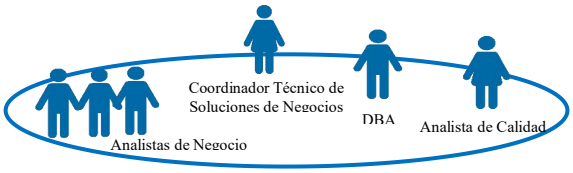
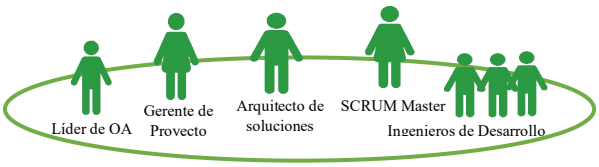


Fig. 8. Perfiles del Equipo de Soluciones de Negocios

- **Coordinador Técnico de Soluciones de Negocios:** Coordinar la identificación, diseño y planificación de la implementación de las soluciones tecnológicas en función de los requerimientos de la universidad para soporte a los procesos académicos y de gestión, base a estándares y metodologías.
- **Analista de Negocio:** Identificar junto con el cliente interno de UTPL los requerimientos de negocio, proponer alternativas de solución y traducirlas a requerimientos técnicos, identificando la vinculación con otras áreas de la Universidad a fin de coordinar con los equipos correspondientes su implementación.
- **Administrador de Base de Datos (DBA):** Administrar y garantizar el correcto funcionamiento de las bases de datos a través de la implementación de controles que permitan asegurar su buen funcionamiento y disponibilidad.
- **Analista de Calidad:** Gestionar y coordinar la ejecución y cumplimiento de estándares, políticas y lineamientos de control y aseguramiento de calidad al

software desarrollado o comprado para minimizar la presencia de errores que afecten a la operación de los sistemas.

5.2.3. Equipo de Fábrica de Software



En la Fig. 9. se muestran los perfiles que conforman el equipo de Fábrica de Software.

Fig. 9. Perfiles del equipo de Fábrica de Software

Para la atención de los servicios por parte de la Fábrica de Software se plantean los siguientes perfiles:

- **Gerente de proyecto**: Responsable por la Gestión y el control del proyecto en conjunto con UTPL.
- **Arquitecto de soluciones:** El arquitecto es un colaborador del gerente del proyecto, pero enfocado en velar por la calidad técnica de los requerimientos que se implementen dentro de los servicios a prestar. Su responsabilidad es la de guiar y apoyar técnicamente al grupo de ingenieros asignados. También, tiene una gran responsabilidad en velar que los requerimientos cumplan con los delineamientos de arquitectura que sean definidos.
- **Ingeniero de Desarrollo:** Encargado de entender en profundidad los requerimientos, crear los artefactos de diseño e implementación de las funcionalidades siguiendo los lineamientos de arquitectura definidos por el Arquitecto.
- **SCRUM Master:** es una persona al servicio del equipo, vela principalmente por la aplicación de la metodología y ayuda al Product Owner en la gestión del Product Backlog.
- **Líder QA:** Realiza los escenarios de pruebas y los ejecuta, con el fin de detectar bugs, para su corrección con el equipo de desarrollo.

5.2.4. Esquema de Comunicación entre Equipos

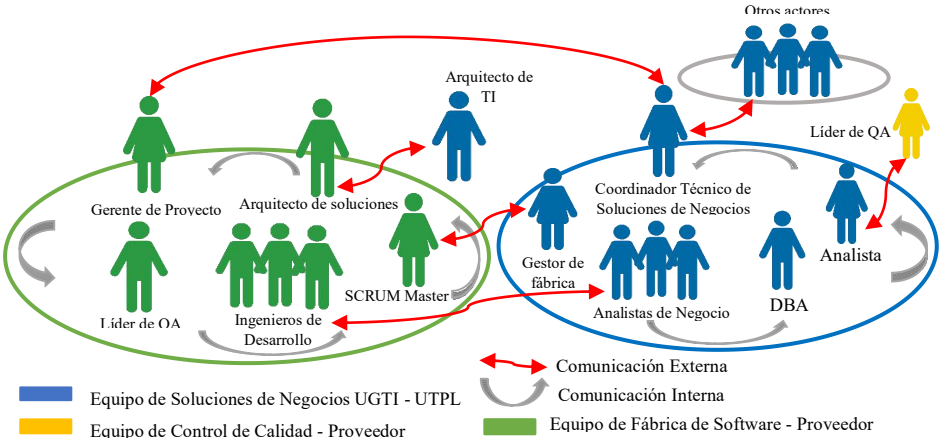


***Fig.* 10.** Esquema de comunicación interna y externa. **Fuente**: Elaboración propia

Como se puede observar en la Fig.10. el equipo de Soluciones de Negocios como el equipo de Fábrica de Software se comunican, tanto internamente como externamente. Cada perfil en cada equipo cumple su función específica, de tal forma que la integración y comunicación entre equipos sea la correcta.

Para medir el correcto funcionamiento del servicio, se comunican directamente el gerente de proyecto y el coordinador técnico de soluciones de negocios, con el fin de encontrar puntos de mejora en el modelo de trabajo de la Fábrica de Software, así como dar seguimiento al cumplimiento de los acuerdos de nivel de servicio. Existen personas clave que coordinan el trabajo entre los equipos y se comunican bilateralmente, entre ellos tenemos el SCRUM Master que se comunica directamente con el gestor de Fábrica; ellos coordinan, planifican y dan seguimiento al trabajo que los ingenieros de desarrollo y los analistas de negocio van a ejecutar, también gestionan la resolución de impedimentos o reuniones de trabajo, desde la etapa de análisis hasta la etapa de aceptación (ver Fig. 6).

La comunicación entre el Arquitecto de TI y el Arquitecto de Soluciones es esencial, ya que en la etapa de Diseño de la Solución (ver Fig.6), el Arquitecto de Soluciones genera propuestas cumpliendo los lineamientos y estándares de arquitectura de UGTI, y estos son aprobados por el Arquitecto de TI.

Los ingenieros de desarrollo y los analistas de negocio, son asignados a requerimientos específicos, y se encargan de trabajar en conjunto de forma muy activa en la etapa de análisis (ver Fig.6).

El analista de Calidad gestiona directamente con el líder de QA, las pruebas de aceptación del producto entregado por la Fábrica de Software, ellos coordinan, planifican y dan seguimiento a la planificación del alcance de las pruebas definidas.

5.3. Estrategia en la selección de metodologías ágiles

A continuación se detalla que criterios definen la selección y aplicación de metodología o herramientas ágiles para la implementación de los requerimientos que gestiona la Unidad de Gestión de Tecnologías de la Información a través del equipo de Soluciones de Negocios.

KANBAN: Se aplica la herramienta KANBAN cuando se identifican requerimientos de corta duración o bugs, en la Fig. 11. se visualiza las actividades principales y los entregables de cada una de ellas.

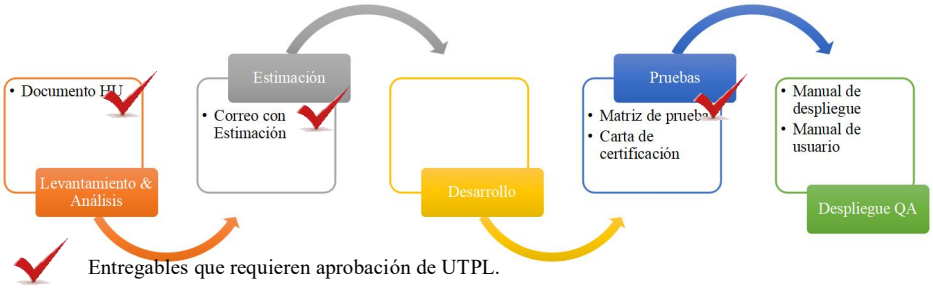


Fig. 11. Actividades y sus entregables al aplicar KANBAN

SCRUM: Para requerimientos con alcance no definido o cambiantes en el tiempo, en la Fig. 12. se muestran las actividades principales y los entregables de cada una de ellas.

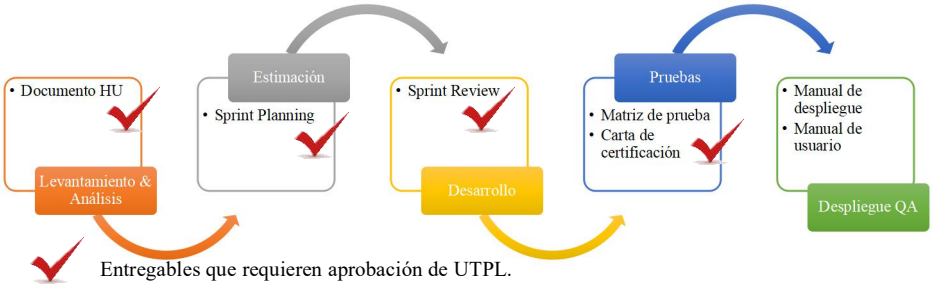


Fig. 12. Actividades y sus entregables al aplicar SCRUM.

6. Resultados

Durante el año 2016 se han obtenido notables resultados aplicando el modelo de Fábrica de Software con el apoyo del proveedor de ese servicio y el equipo de la Unidad de Gestión de Tecnologías de la Información de la universidad, evidenciándose así a través de los siguientes datos: se ejecutaron 6 proyectos institucionales, se implementaron 108 requerimientos globales y 11 sistemas fueron vinculados con estos desarrollos. A continuación en la tabla 2 se detalla cada uno de ellos:

Tabla 2. Resumen de resultados con el modelo de Fábrica de Software en el año 2016

Proyectos Institucionales	Postgrados (Admisión), Proyecto Pedagógico Institucional (Restructuración de mallas), Ampliación de Servicios Estudiantiles (Reconocimiento de estudios fase3, Certificado Índice Mérito Graduado y Certificado Promedio de Notas), Seguimiento Integral al Estudiante (Control de porcentaje), Integración Datos y Sistemas y Reingeniería CEDIB.
Sistemas Vinculados	Sistema de Gestión Académico, Sistema Financiero Académico, BAAN, Registro de Asistencias (QR), Apolo (Planes Docentes), Evaluaciones Tradicionales, Evaluación al Docente, CEDIB, EVA, SIEC y Educación Continua.
Requerimientos de TI para mejoras de mantenimiento.	Pasarela Documental, Reglas de Negocio (ODM – ILOG), Reingeniería Registro Inscripción, Campañas de Captación (Registro y Financiero), Mejora de trazabilidad de errores , Generaciones masivas , Conciliación Automáticas, Integración Bancos (Bolivariano), Crédito Educativo Banco Loja, Componentes Integrados y Nuevo modelo de aranceles

6.1. Requerimientos atendidos

Con el equipo de trabajo de la Fábrica de Software se ha obtenido una media de 23 requerimientos de desarrollo de software que se ejecutan por mes, esto quiere decir que la Fábrica de Software atiende este número de requerimientos ya sea que inicien en la etapa de análisis, desarrollo o pruebas, como se puede visualizar en la Fig.13.

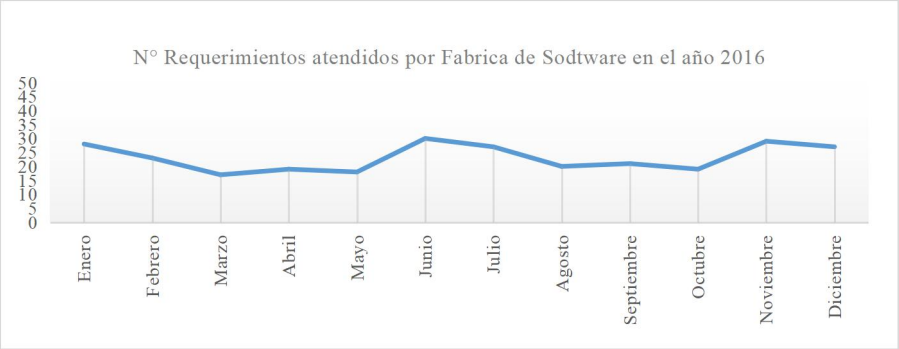


Fig. 13. Número de requerimientos atendidos por la fábrica de software

En la experiencia vivida se ha identificado que la capacidad de la fábrica y su uso efectivo dependen del conocimiento de negocio transmitido al equipo de desarrollo de fábrica y la selección de la metodología ágil de desarrollo.

6.2. Número de Incidentes

En la Fig.14. se muestra el número de incidentes que han sido reportados y corregidos desde el 2015 hasta el 2017, donde se evidencia el decrecimiento de la cantidad de correcciones de incidentes, que son el resultado de la implementación del modelo de Fábrica de Software, lo que involucra la intervención de una visión arquitectónica integral y cumplimiento de estándares de calidad.

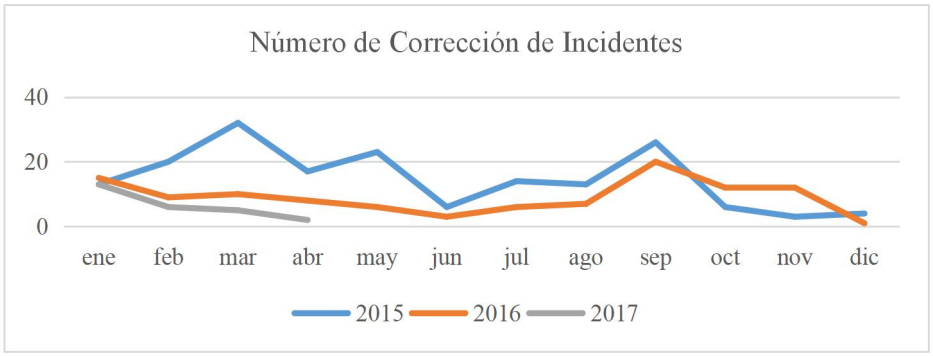


Fig. 14. Número de incidentes corregidos

6.3. Indicadores de calidad

Existen algunos indicadores de calidad que son medidos sobre los desarrollos realizados por fábrica en el Sistema de Gestión Académica, los cuales se detallan a continuación:

Líneas de código: Número de líneas de código físicas que contienen por lo menos una línea de código.

API púbica documentada (%): Porcentaje de numero de clases públicas + el número de funciones públicas + número de propiedades públicas, el API documentada ayuda a entender la funcionalidad de una clase o función.

Líneas duplicadas (%): Porcentaje de líneas duplicadas que existe en el código, el código duplicado requiere más esfuerzo y tiempo para solucionar bugs ya que se encuentran en muchos lugares.

Deuda técnica ratio: Indica el porcentaje de incumplimiento de las métricas anteriores que se deben corregir para que un proyecto cuente con una calidad aceptable.

Para que un proyecto tenga una buena calidad de código debe cumplir los siguientes porcentajes según el estándar definido:

API publica documentada >= 80%
Líneas duplicadas <= 10%
Deuda técnica <= 20%

En la tabla 3 se describen los tres indicadores de calidad y su porcentaje respectivo durante mayo del 2015 hasta julio del 2016, donde el indicador de API publica documentada no cumple con ser mayor o igual al 80%. El indicador de líneas duplicadas inicia con un porcentaje del 12,10% y disminuye en julio del 2016 a 10,00%. Finalmente la deuda técnica disminuye de 23,80% a 21,40%.

Tabla 3. Indicadores de Calidad durante el año 2015 y 2016

Indicador	Año 2015				Año 2016			
	Mayo	Junio	Julio	Octubre	Enero	Febrero	Junio	Julio
API publica documentada	64,50%	64,50%	64,70%	66,50%	66,80%	67,70%	64,80%	64,90%
Líneas duplicadas	12,10%	12,00%	11,80%	10,00%	9,90%	9,60%	10,80%	10,80%
Deuda técnica	23,80%	23,70%	23,50%	21,40%	21,30%	23,80%	21,50%	21,40%

En la tabla 4 se detalla el número total de líneas de código del Sistema de Gestión Académica durante mayo del 2015 hasta julio del 2016, donde se visualiza que inicialmente existen 358.099 líneas de código y a julio incrementa a 573.790 líneas de código.

Tabla 4. Número de Líneas de código

Indicador	Año 2015				Año 2016			
	Mayo	Junio	Julio	Octubre	Enero	Febrero	Junio	Julio
Líneas de Código	358.099	360.947	365.703	423.413	433.154	582.560	557.343	573.790

Es importante mencionar que a pesar del crecimiento considerado de líneas de código en el Sistema de Gestión Académica, disminuye el porcentaje de líneas duplicadas y la deuda técnica, esto permite evidenciar el cumplimiento de estándares de desarrollo en las implementaciones que la fábrica de software ha realizado.

7. Conclusiones:

- Las metodologías ágiles de desarrollo son un marco de referencia y deben ser adecuadas a la realidad de cada institución, en el caso particular de la Universidad Técnica Particular de Loja se decidió utilizar los componentes de diferentes metodologías que generan mayor valor a la institución y permiten entregar un producto de forma ágil sin descuidar la calidad ni el cumplimiento de los estándares definidos. Esto permitió atender los requerimientos de forma más eficiente y con los criterios de calidad de los clientes internos.
- El número de incidentes reportados durante el año 2016 representa una disminución del 38% con respecto al año 2015, cuyos resultados nos permiten evidenciar la entrega de un producto con mayor calidad.
- La curva de aprendizaje y transición del conocimiento técnico de los sistemas asignados a fábrica así como de los principales procesos de negocio, ha ido decreciendo, gracias al apoyo del equipo técnico de UGTI, y la definición de estrategias como que el equipo de la fábrica de software durante el proceso de análisis trabaje en sitio, y participe directamente con los funcionales en el modelado de los requerimientos de negocio.
- Los indicadores de calidad evidencian que a pesar del incremento de líneas de código el porcentaje de líneas duplicadas disminuye considerablemente hasta el 10,80% por lo que decrece el esfuerzo y tiempo para solucionar bugs. Otro indicador de calidad del software es la deuda técnica, se obtuvo también una disminución de 2 puntos porcentuales aproximadamente, aunque no está dentro del rango aceptado se observa que la tendencia es optimista, esto permite reducir los sobrecostes operativos y de mantenimiento.
- La gestión de requerimientos a través de metodologías ágiles, permite extraer de los desarrollos de software el máximo valor para la universidad, facilitando la implementación eficaz de las estrategias de negocio.

8. Próximos Pasos:

A continuación se enlistan los próximos pasos a realizar con el modelo de fábrica de software.

- Incrementar el alcance en los sistemas que son atendido con el equipo de la fábrica de software, lo que permitirá tener mayor cobertura de los sistemas satélites que también requieren adecuaciones o modificaciones.
- Incorporar procesos de mejora continua en base a los indicadores obtenidos que permitan generar resultados con optimización de los recursos empleados.

Referencias

[1] Universidad Técnica Particular de Loja, «UTPL,» 2015. [En línea]. Available: http://www.utpl.edu.ec/. [Último acceso: 8 Junio 2016].

[2] Universidad Técnica Particular de Loja, «Plan estratégico,» 2011. [En línea]. Available: http://www.utpl.edu.ec/sites/default/files/documentos/PLAN-2011-2020-A.pdf.

[3] A. Fernández Martínez y F. Llorens Largo, «Gobierno de las TI para universidades,» de Conferencia de Rectores de las Universidades Españolas (CRUE), Madrid, 2011.

[4] P. Letelier Torres y E. A. Sánchez López, «Metodologías Ágiles en el Desarrollo de Software,» de Taller realizado en el marco de las VIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2003, Alicante, 2003.

[5] J. Garzás, «Javier Garzas,» 23 Marzo 2013. [En línea]. Available: http://www.javiergarzas.com/metodologias-agiles. [Último acceso: 21 Marzo 2017].

[6] ServiceTonic, «ServiceTonic,» [En línea]. Available: https://www.servicetonic.es/itil/itil-v3-gestion-de-problemas/. [Último acceso: 15 Marzo 2017].

[7] itService, Gestión de Servicios de Tecnología de la Información Fundamentos ITIL, Cali, 2014.

[8] A. Orjuela Duarte y M. Rojas C, «Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo.,» Avances en Sistemas e Informática, vol. 5, pp. 159-171, 2008.

[9] K. Schwaber y J. Sutherland, «Scrum,» Julio 2013. [En línea]. Available: http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf. [Último acceso: 15 Marzo 2017].

[10] Intergrupo, Propuesta Modelo de Fábrica, Quito, 2016.