APLICATIVO DE TESTING BASADO EN MODELOS ÁGILES.

POR:
MARTA ESTER GÓMEZ ADASME [1]
JUAN DAVID VAHOS M [2]
RUBÉN DARÍO CASTRILLÓN GUTIÉRREZ [3]

| RESUMEN

Dentro de los modelos de desarrollo de software están las metodologías ágiles que revolucionan y establecen nuevas formas eficientes de realizar aplicativos. Los aprendices del Sena en el transcurso de la etapa lectiva necesitan realizar pruebas a las aplicaciones para clientes reales durante todo el ciclo de desarrollo, teniendo en cuenta las condiciones de instalación de software del mundo actual, como es la virtualización de servicios y las buenas prácticas ágiles de desarrollo. El objeto de este proyecto es poner en marcha un ambiente de pruebas con productos propios, los cuales se compararán con productos del mercado para realizar las pruebas de software de las aplicaciones de etapa lectiva de aprendices del área del desarrollo de software y los productos empresariales que genera la Fábrica de Software del Centro.

El proyecto consiste en tres productos así: un ambiente de testing para permitir las pruebas del proyecto formativo en las diferentes fases que debe cumplir como análisis, diseño, desarrollo e implementación; al utilizar la virtualización en un solo servidor, se pueden tener varios servicios a la vez y es más fácil recuperarse ante caídas o fallas. La segunda fase es la creación de un software que permite hacer un seguimiento del ciclo o workflow del proceso de pruebas y donde se registran los artefactos para realizar los casos de prueba. La tercera fase es la implementación de software que permita la automatización de las pruebas; clasificados en seguimiento del proceso, generación de scripts y manejo de la concurrencia o performance.

Palabras claves: Pruebas, virtualización, Metodologías ágiles, desarrollo, aplicativos.

[1] Filiación (Semillero de Investigación MERLIN - Instructor- SENA), mega2808@misena.edu.co
[2] Filiación (Semillero de Investigación MERLIN -Instructor- SENA), jdvahos@misena.edu.co
[3] Filiación (Semillero de Investigación MERLIN -Practicante- SENA), kalelr@gmail.com

I. INTRODUCCIÓN

Las aplicaciones de software actualmente resuelven grandes contabilidades de empresas en todo el mundo, inclusive pequeñas soluciones para una empresa familiar. El software y el hardware están presentes desde lo cotidiano, como en el transporte público simple (buses, taxis, entre otros), hasta sistemas más complejos como el funcionamiento de los trenes, que transportan miles de personas diariamente (este es el caso del Metro de Medellín). Estas tecnologías se utilizan para el control del acceso a los usuarios, los sistemas de aire acondicionado, entre otros; los cuales son controlados a través de aplicaciones, que garantizan su óptimo funcionamiento. Las consecuencias de la mala operación de los softwares pueden llegar a ser fatales. Tal es el caso de las aplicaciones en salud que pueden ir desde una cita mal creada hasta un aparato de electrocardiograma que funcione erróneamente, pueden generar un tratamiento equivocado y quizás la muerte.

Se puede complementar la anterior afirmación con la siguiente referencia "La creciente dependencia de tareas críticas respecto del software conlleva a que el valor del mismo ya no reside solamente en la aptitud que posee de mejorar o sostener la productividad y la eficiencia de las organizaciones, sino que además se requiere que posea la capacidad de continuar operando de manera confiable aun cuando deba enfrentar eventos que amenazan su utilización."Castellaro, M., Romaniz, S., Ramos, J., & Pessolani, P. (2009).

Si las aplicaciones de software están presentes en la cotidianidad del hombre y en las empresas, se debe garantizar su buen funcionamiento. Para ello, se deben controlar, validar y verificar constantemente, con el propósito de evitar posibles pérdidas económicas, reprocesos y tiempo.

"La construcción de software es el evento fundamental de la ingeniería de software. Los programadores trabajan construyendo e integrando programas a través de técnicas de codificación, validación y pruebas. Pero ese carácter esencial no minimiza fases tan cruciales como la planeación del proyecto, el análisis de requerimientos, el diseño y la gestión de la calidad." Parra Castrillón, E. (2011).

La importancia de las pruebas de software ha ido creciendo con el tiempo, ya que las aplicaciones son cada vez más complejas por la información que manejan, la utilidad que tienen y los dispositivos involucrados en las soluciones informáticas. Se llaman así soluciones porque pueden involucrar tanto software como hardware y ambos en relación. "Las pruebas de software son seguramente la actividad más común de control de calidad". Sanz, L. F. (2005).

Las pruebas son realizadas en los proyectos de desarrollo o mantenimiento de aplicaciones y sistemas. La evolución de las pruebas ha ido de la mano del aseguramiento de la calidad "El aseguramiento de calidad del software es el conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (software) satisfará los requisitos dados de calidad" Lovelle, J. (1999).). Las certificaciones internacionales que garantizan que las empresas pueden trabajar en todo el mundo porque han adquirido buenas prácticas en la implementación de sus proyectos de software para hacerlos exitosos como SEI con CMMI (Integración de modelos de madurez de capacidades o Capability Maturity Model Integration) o certificaciones como Scrum Master; la búsqueda de certificaciones internacionales para los analistas de pruebas como el ente certificador ISTQB (International Software Testing Qualifications Board) y toda la gama de certificaciones a nivel mundial para los tester.

Las metodologías de desarrollo de software también evolucionan para adaptarse a las necesidades

del medio y con ello surgen metodologías de desarrollo ágil que se plantean como alternativa a las metodologías tradicionales como RUP. Una metodología ágil como Scrum plantea un cliente activo dentro del proceso de desarrollo de su aplicación, con entregas totalmente funcionales en cada Sprint review y donde el cliente en conjunto con el equipo hacen la validación y verificación del desarrollo con cierto desprecio a la documentación de las fases y al UML (Lenguaje unificado de modelamiento: herramienta fundamental para las fases de análisis y diseño de proyecto en las metodologías tradicionales). Dentro de Scrum han surgido varias formas de ejecutar las pruebas ágiles como son TDD (Desarrollo guiado por pruebas de software) o su evolución ATDD (Acceptance test–driven development o desarrollo guiado por criterios de aceptación que son introducidos en cada historia de usuario).

En las pruebas tradicionales podemos mencionar dos grandes grupos las pruebas de caja blanca (o pruebas basadas en el código fuente caminos básicos, control de flujo, control de datos o pruebas de ramificación) y las de caja negra (o pruebas basadas en la funcionalidad del producto)y dentro de ellas podemos mencionar las pruebas de clases de equivalencia, tablas de decisión, transición de estados y pruebas de experiencia.

Los aprendices Sena de la tecnología de Análisis y desarrollo de software se forman con base a proyectos que involucran en su mayoría soluciones a la medida de clientes reales, que esperan una aplicación con altos niveles de calidad y que sea la evidencia de su formación, por lo tanto surge la necesidad de validar y verificar sus aplicaciones y que sean ellos mismos quienes lo hagan en un ambiente fiel a la realidad, registrando todo el proceso en software especializado para ello.

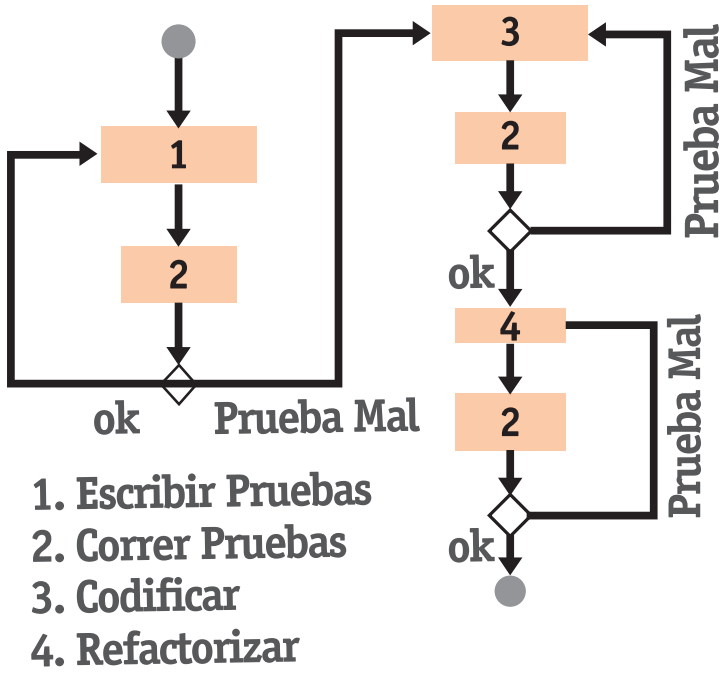

Figura 1. Pruebas TDD.
Fontela, M. C. (2011).

II. METODOLOGÍA

En el proceso de desarrollo con modelos ágiles el primer paso es el levantamiento de requisitos con el cliente, el cual se hace periódicamente y juega el rol de Product Owner (PO), quién es el que da el lineamiento sobre el "Qué" del producto. Adicional a esto, hace la priorización de dichos requisitos, lo cual, se convierte en el primer artefacto SCRUM como lo es el Product Backlog (Peralta, 2005). Con esta priorización se da inicio a la fase de planeación del primer Sprint. Los sprint corresponden a la funcionalidad priorizada por el cliente, la cual, se despliega en una cartelera llamada Srint Backlog que posee tres columnas (Por Hacer, en Proceso, Terminado) para ubicar las historias de usuario (similar a los requerimientos, pero en un lenguaje más amigable para el cliente). Una vez planteados estos Sprint iniciales, entra en acción el rol del TEAM (Equipo de Desarrollo), el cual es el encargado del desarrollo del software. Una vez realizados de dos a tres Sprint se realiza el entregable llamado RELEASE, que es el equivalente a una liberación de producto. En el caso del aplicativo Gestor de Casos de Prueba (NIVAL) se prioriza inicialmente la liberación de la funcionalidad de Gestión de Casos de prueba, la cual ha sido entregada inicialmente para que el cliente (PO) pueda hacer un feedback en la reunión de Sprint Review de forma que se puedan hacer ajustes en la carga de tareas del siguiente Sprint y de ser el caso se pueda hacer una repriorización de la pila de requerimientos (Product Backlog).

Paralelo a el desarrollo de Nival 2.0, se está realizando por medio de SENNOVA y su semillero MERLIN la gestión de adquisición de una herramienta CASE que permita al igual ejecutar pruebas similares a las que se pretenden realizar con el aplicativo hecho a la medida. En el mercado se han encontrado otras herramientas que poseen licencias GPL (Generic Public Licence) o de uso Público como Mantis Bug Tracker y Selenium HQ, las cuales pretenden crear un escenario de línea base que le permita al proyecto de Testing poder realizar comparativos de los diferentes escenarios de prueba que se obtendrán con la aplicación desarrollada por el Team (Equipo) del proyecto.

Figura 2. Logos del Software Mantis. Fuente: (https://www.mantisbt.org).

Una vez se tengan tanto la herramienta propia a la medida como la adquisición del software CASE y el permiso de uso de las aplicaciones con licencia GPL, se realizará la configuración de los equipos en el entorno de trabajo del laboratorio. El laboratorio consta de 20 Equipos de tipo Work Station con procesador Intel® Xeon® Quad Core Processor, 16GB Memory, 1TB Hard Drive, Tarjeta Video Nvidia 2GB. Monitor de 21". También se adquirió un servidor que permite la virtualización de los diferentes escenarios de pruebas en diferentes configuraciones de OS (Sistema Operativo) y Bases de datos. Este servidor contó con las siguientes características relevantes: procesador Intel® Xeon® E5-2620 v3 2,4 GHz, caché de 15 M, Memoria RDIMM de 16 GB, 2400 MT/s con clasificación doble ancho de datos con posibilidad de ampliar a 64 GB y 2 discos duros de conexión en caliente de 1TB 7200 RPM 6Gbps de 3.5" ampliable a 20 TB por medio de bahías de expansión. Estas especificaciones permitirán correr las diversas pruebas de diferentes productos de forma simultánea, para evitar tiempos de espera, o productos en cola con una configuración multi-hilo.

Para el laboratorio de pruebas de software, se adquirieron equipos con unas especificaciones técnicas que facilitaran la instalación de las aplica-

ciones de los aprendices y la realización de las pruebas, recordemos lo que nos dicen "Las interfaces gráficas (GUIs) representan un elemento fundamental y crítico de las aplicaciones de hoy en día, llegando a acaparar incluso hasta el 60% del código. Por lo tanto, probar la funcionalidad de las GUIs se presenta como una tarea imprescindible para asegurar la robustez, usabilidad y calidad del sistema" Luis, P., Navarro, M., Pérez, G. M., & Ruiz, D. S. (2009).

Se completaron los equipos con la adquisición de un servidor para poder atender a los diferentes ambientes de software, como: Windows con su servidor de http- servidor que permite publicar páginas web - internet information server - IIS, el motor de bases de datos SQL Server y el lenguaje de programación Visual Studio con C#; o Windows con el servidor de Http, con Apache Tomcat , el motor de bases de datos Mysql o Mariadb y el lenguaje de programación Java para web o PHP ; o Linux en alguna de sus distribuciones, ya sean Centos o Red Hat con el servidor de http con Apache o con Internet Information Server IIS para Linux , el motor de bases de datos Mysql, MariaDb o Postgres y con los lenguajes de programación PHP, Java web, Java escritorio y C#. Todo esto en ambientes virtuales dentro del mismo servidor.

III. RESULTADOS Y DISCUSIÓN

La información es actualmente un bien que debe ser cuidado y preservado, es valioso para las empresas y las aplicaciones que la manejan deben cumplir con unas propiedades que permitan a sus propietarios estar satisfechos con las mismas. Además, deben de cumplir con ciertas métricas que le permiten ser evaluado como software con o sin calidad, tales como: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y calidad en uso, como se menciona la norma ISO 9126 (UNE-ISO/IEC 9126-1:2004). Como lo mencionan los autores en la siguiente cita: "La mejora continua en los procesos y productos de software es lo que se conoce como calidad total, que en gran parte es conseguida a través de la aplicación y evaluación de las métricas, y la realimentación del proceso productivo con estos resultados." Piattini, M. G & García, F. O. (2003).

Ante esto surge y se establece como responsable de la validación y verificación de las diferentes aplicaciones de software las pruebas o testing de las mismas siendo una herramienta que debe ser efectiva, eficiente y el evaluador que los procesos implementados en una aplicación de software hacen lo que deberían hacer.

Permitir que los aprendices de la tecnología en análisis y desarrollo de sistemas de información den la importancia y reconozcan las responsabilidades, que como tester o probadores se tiene dentro del proceso de desarrollo de software, es uno de los objetivos del proyecto de testing en ambientes ágiles. Muchos desconocen o evitan asumir la tarea de testing dentro de los entornos ágiles, pero es ahí donde las pruebas de software toman una relevancia más alta, ya que en las historias de usuario -requisitos funcionales pero contados en palabras y necesidades del cliente - para efectuar una verificación efectiva de las mismas, se redactan los criterios de aceptación. Los criterios de aceptación, son para el tester o analista de pruebas, los casos de prueba a ser demostrados en la ejecución de la historia de usuario. Esto da pie, para hacer el paralelo entre lo que se considera la metodología tradicional para realizar pruebas de software y lo que se consideran las pruebas ágiles o testing ágil; con el inicio de la metodología tradicional o metodología RUP (El proceso racional unificado o en sus siglas en inglés Rational unified Process), que establece que todo proyecto de software debe trasegar las fases de

identificación de requisitos, análisis, diseño, desarrollo e implantación. Dentro de esta última, se sitúan las pruebas de software. El modelo establece que cada fase debe entregar unas evidencias o artefactos a la siguiente, y los responsables de cada fase asumen roles de acuerdo a las tareas asignadas dentro de la misma. El ciclo de vida más utilizado es el de cascada, que establece que una fase no puede empezar hasta que la otra no este correctamente terminada. Cabe resaltar que es justo en este punto donde Calidad de Software debe verificar que las evidencias se han cumplido y pueden ser el insumo para la siguiente fase.

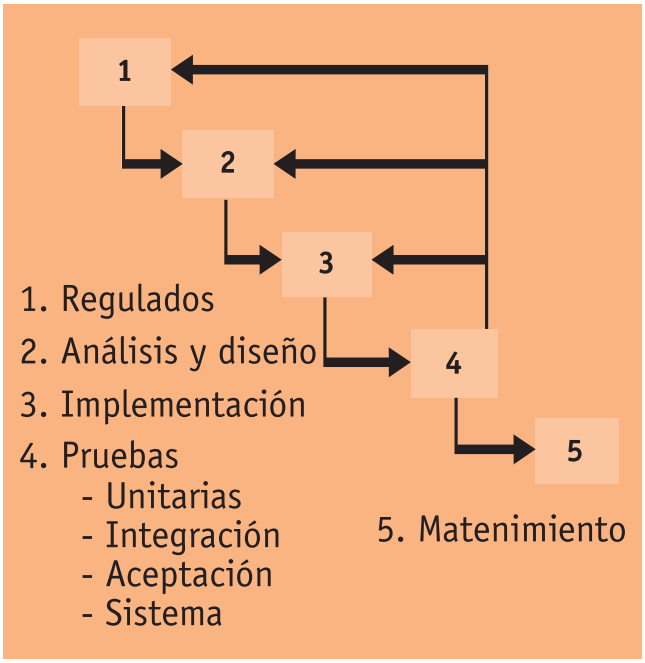

Figura 3: Figura del modelo RUP cascada.

Un ciclo de vida para los proyectos de software, que promueve las buenas prácticas de pruebas tradicionales, es el ciclo en V, donde las pruebas se realizan en cada fase, para no permitir que un error trascienda de una fase a otra; pero siguiendo los lineamientos de la metodología tradicional, donde las pruebas están en la fase de implantación. Al final del proceso de desarrollo, los errores han pasado de fase en fase y es responsabilidad del equipo de pruebas la calidad del producto. Entonces, al aplicar las buenas prácticas de testing en la metodología tradicional; se encuentran pruebas unitarias o de componente, de integración, de sistema y de aceptación, como se aprecia en la Figura 4:

Figura 4. Ciclo de vida en V.

Las metodologías ágiles presentan las pruebas TDD- Desarrollo orientado a las pruebas o ATDD- Desarrollo orientado a los criterios de aceptación. Por parte del usuario han motivado la incorporación de las metodologías ágiles dentro de las empresas Se puede comprobar que cerca del 70% de las empresas ya están incorporando algunas prácticas ágiles en su proceso de desarrollo. Según estos autores, esto ha traído como consecuencia una mejora de la calidad de los productos entregados en casi un 70% de los proyectos, como puede verse en la Figura. 5. Yagüe, A., & Garbajosa, J. (2009).

El proyecto presentado a Sennova, "Aplicativo de testing para ambientes ágiles", incluye tres productos:

-Un laboratorio de pruebas de software.

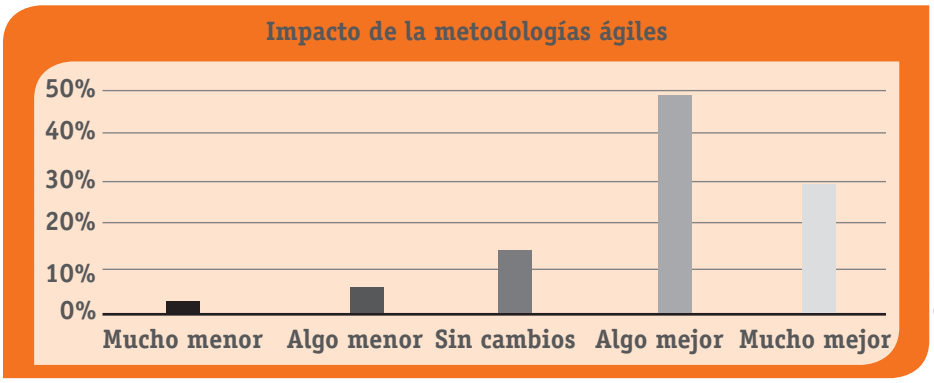

Figura 5. Impacto de las metodologías ágiles en la Calidad de Software: Yagüe, A., & Garbajosa, J. (2009).

-Un software especializado para ese laboratorio.
-Un software para el manejo de las pruebas.

La compra de un software especializado y la utilización de software para pruebas de software, como Testlink software, de uso gratuito que permite la creación y desarrollo del plan de pruebas con su cronogramas, sus casos de pruebas ya sea unitarias, de integración y de regresión; también Mantis que permite la comunicación entre el analista de pruebas y el desarrollador de la aplicación; otros software a ser trabajados y probados son X-qual (Manejo de planes de pruebas), Jmeter (Pruebas de performance o cuantas peticiones al mismo tiempo puede atender nuestro software web), Jira (Software que permite el manejo de proyectos en Scrum) , Selenium (Permite la automatización de pruebas ya sea por grabación de acciones dentro de una página web o dos con la ayuda de Junit, que permite una automatización con más variaciones y verificación de las especificaciones de las aplicaciones) , Watir (Software que se trabaja con el lenguaje de programación Ruby y que permite la automatización de pruebas, a través de grabar acciones y estas generan el código que puede ser modificado, de acuerdo a los criterios de aceptación de la historia de usuario de la aplicación) y otros software como Cucumber, AppIUM-Android, TestLodge, Sonarqube y muchos más que permiten la automatización de las pruebas de software.

Se desarrolló un software llamado Nival para el manejo de las pruebas, creado por aprendices SENA, un producto de la Fábrica de software para el semillero Merlín y un insumo, no sólo para el laboratorio de pruebas de software, sino también para el quehacer de los instructores SENA, que dan formación en pruebas de software, en sus respectivos centros de formación a nivel nacional. El proyecto Nival 2.0 es un gestor de casos de pruebas – GCP, que fue utilizado como proyecto de formación para la especialización tecnológica de pruebas de software. El proyecto de software Nival 2.0, es desarrollado bajo la arquitectura de software SOA (Aplicación Orientada a Servicios).

Figura 6. Inicio de sesión. Aplicativo Nival 2.0 Fuente: propia.

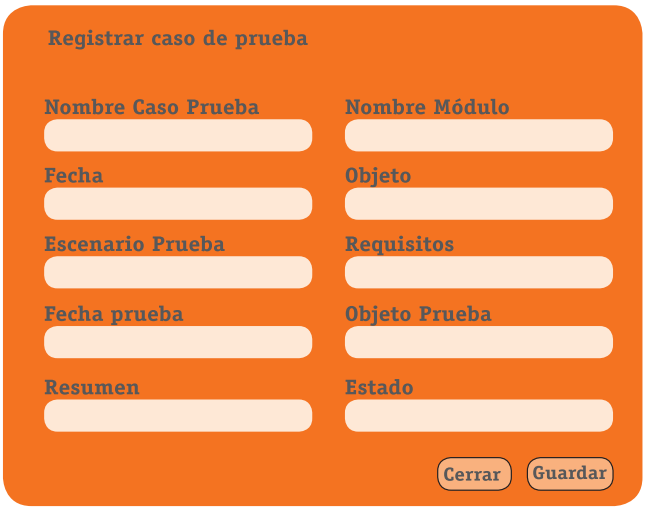

Figura 7. Módulo de Registro Plan de Pruebas, aplicativo NIVAL 2.0 Fuente: Propia.

La aplicación se desarrolló en el lenguaje de programación de C# con MVC 5, motor de bases de datos SQL Server.

El proyecto de "El aplicativo de Testing basado en modelos ágiles" da como resultado un laboratorio de pruebas, un aplicativo que permite hacer el seguimiento tanto a las pruebas tradicionales como a las pruebas ágiles y la compra de un software para la mejora del proceso de pruebas de software. Pero lo más importante, darles a los aprendices un espacio de trabajo con tecnología de punta que les permita probar sus aplicaciones y registrar el proceso para mejorar el producto a los clientes

En la figura 8 se muestra el reporte de un Plan de pruebas realzado a un proyecto interno de la Fábrica de Software llamado GIDPI, en este resalta el detalle del informe por Roles, y atributos relevantes para el Jefe de Pruebas como lo son: Nombre del caso de prueba, Escenario de prueba, Resumen, Prueba realizada, Estado, Nombre del artefacto, Descripción, Resultado esperado y Resultado obtenido.

Figura 9. Estadísticas del Plan de Pruebas, aplicativo NIVAL 2.0 Fuente: Propia.

En la Figura 9 se puede observar la estadística del Plan de Prueba realizado para la aplicación GIDPI (http://www.gidpi.com) por parte de la aplicación NIVAL 2.0. En este se registran la cantidad de casos y la cantidad de evidencias recolectadas las cuales están asociadas a los criterios de aceptación. Otros aplicativos que han sido probados con la herramienta NIVAL 2.0, y que han sido desarrollados en la Fábrica de Software son: SAM (Soluciones ágiles en Mantenimiento), TruckCloud 2.0, y Calendar 365.

Para la Fábrica de Software del Centro de Servicios y Gestión Empresarial, el Laboratorio de Testing servirá como un ente externo a la Fábrica (de manera que no sea juez y parte) que permita que sus productos de software sean probados en un entorno especializado bajo unos protocolos establecidos y docu-

Reporte plan de prueba			
Plan	Fecha inicio	Fecha fin	Objetivo
Espacio para llenar	Espacio para llenar	Espacio para llenar	Espacio para llenar
Caso	Resumen	Prueba	Estados
Espacio para llenar	Espacio para llenar	Espacio para llenar	Espacio para llenar
Nombre de prueba		Escenario de prueba	
Espacio para llenar	Espacio para llenar	Espacio para llenar	Espacio para llenar
Resulatdos Esperados		Resultados obtenidos	
Espacio para llenar	Espacio para llenar	Espacio para llenar	Espacio para llenar

Figura 8. Reporte Plan de Pruebas, aplicativo NIVAL 2.0 Fuente: Propia.

mentados de forma que faciliten el cumplimiento de los estándares de calidad que pide la industria del software, tales como Funcionalidad, Mantenibilidad, Confiabilidad, Usabilidad, Portabilidad y Eficiencia. La herramienta NIVAL 2.0 es un producto de esta Fábrica y aporta significativamente a la mejora en la calidad de los productos de software allí elaborados.

IV. CONCLUSIONES

El laboratorio testing se hace necesario para realizar pruebas a las aplicaciones de los aprendices en etapa lectiva y a los proyectos realizados en la Fábrica de Sofware del Centro. La aplicación del software Nival 2.0 ayudará al laboratorio ya que en este software se podrán registrar las pruebas y tabular de forma clara sus resultados. Las pruebas de software para ambientes ágiles, pueden mejorar la calidad del software, ya que permiten la interacción del cliente con el desarrollo del software en los Sprint Review (reuniones convenidas entre 8 y 15 días donde el tester o probador con el cliente verifica el correcto funcionamiento de la historia de usuario desarrollado en este periodo).

Antes de poner en funcionamiento el laboratorio de pruebas, es necesario realizar una línea base con software de pruebas comercial (herramienta CASE) y software con licencias GPL (Generic Public Licence), las cuales deben ser aprobadas ante la oficina de Sistemas de la entidad para su uso en dicho ambiente, de forma que sirvan de comparativo y verificación de los resultados obtenidos con la herramienta hecha a la medida.

La industria actualmente requiere pruebas automatizadas que permitan ejecutar en menores tiempos grandes procesos sin sacrificar el valioso tiempo de analistas, probadores y desarrolladores. El laboratorio de pruebas apoyará el versionamiento de la herramienta Nival a un nivel 3.0 donde este tipo de pruebas puedan ser ejecutadas bajo estándares nacionales (Icontec) e internacionales (ISO 25000).

V. AGRADECIMIENTOS

A los aprendices de Tecnólogo en Análisis y Desarrollo de Sistemas de Información: Jonathan Gil, Esteban Vergara y Daniel Agudelo, que han motivado el esfuerzo de presentar este proyecto y que, gracias a su determinación, pudo ser tomado en cuenta para convertirlo en una realidad. A todos los miembros del semillero MERLIN y el equipo SENNOVA del Centro, que han tenido fe en nuestro trabajo y han participado activamente en el cumplimiento de las actividades del proyecto.

REFERENCIAS

S., Jairo, J., García, T., Pérez, Sánchez, E. (2013). Mejora de historias de usuario y casos de prueba de metodologías ágiles con base en TDD. Journal Technology, 12, 82–104. Recuperado de: http://issi.dsic.upv.es/archives/f-10 69167248521/actas.pdf%5Cnhttps://www.dnp.gov.co/Portals/0/archiv os/documentos/Subdireccion/Conpes/3582.pdf

Engineering, C. (2007). Research Findings Impact of the Graduates from the Computer Engineering Program and of Their Perform.

Engström, E., & Runeson, P. (2011). Software product line testing - A systematic mapping study. Information and Software Technology, 53(1), 2–13. https://doi.org/10.1016/j.infsof.201 0.05.011

Fernandez, A., Insfran, E., & Abrahão, S. (2011). Usability evaluation methods for the web: A systematic mapping study. Information and Software Technology, 53(8), 789–817. https://doi.org/10.1016/j.infsof.201 1.02.007

Fontela, M. C. (2011). Estado del arte y tendencias en Test-Driven Development. Tesis de posgrado. Universidad Nacional de La Plata, 1, Argentina. Recuperado de: http://web.fi.uba.ar/~cfontela/Fontela_EstadoDelArteTDD_UNLP_E IS.pdf

Lovelle, J. (1999). Calidad del Software.
Conferencia, 21 de Octubre de 1999 Grupo GIDIS Universidad Nacional de La Pampa, 1–12. Recuperado de: http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r35043.PDF

Luis, P., Navarro, M., Pérez, G. M., & Ruiz, D. S. (2009). Open HMI Tester: un Framework Open-source para Herramientas de Pruebas de Software 2 Arquitectura Open HMI Tester. Event London, 3(4), 61–66.

Peralta, A. (2005). Universidad ORT Uruguay. Recuperado de: http://www.ort.edu.uy/facs/pdf/doc umentodetrabajo18.pdf

Schenone, M. H. (2004). Diseño de una Metodología Ágil de Desarrollo de Software. Tesis de pregrado. Facultad de Ingeniería: Universidad de Buenos Aires, Argentina. 1–199. Recuperado de: http://books.openlibra.com/pdf/dis eno-metodologia-agil-de-desarrollo-software.pdf

Vaca, P., Maldonado, C., Inchaurrondo, C., Peretti, J., Romero, M., & Bueno, M. (2015). Test-Driven Development - Una aproximación para entender su utilidad en el proceso de desarrollo de Software. Universidad Tecnológica Nacional, Facultad Regional Córdoba, 1–10. Recuperado de: http://conaiisi.frc.utn.edu.ar/PDFsParaPublicar/1/schedConfs/7/158-524-1-DR.pdf

UNE-ISO/IEC 9126-1:2004

Ingeniería del software. Calidad del producto software. Parte 1: Modelo de calidad 2004

(http://www.aenor.es/aenor/normas/normas/fichanorma.asp?tipo=N&codigo=N0032555#.Wf5_ZI_Wwy4)

Yagüe, A. (2009). Las pruebas en metodologías ágiles y convencionales: papeles diferentes, Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, 3(4), 67–73