Experiencia y resultados en la aplicación de gestión de requerimientos de Software en proyectos a gran escala

Natalia Andriano, Mauricio Silclir, Paula Izaurralde

{nandriano, 47920, 46054}@sistemas.frc.utn.edu.ar
Laboratorio de Investigación en Ingeniería y Calidad de Software
http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/
Departamento de Ing. en Sistemas de Información
Universidad Tecnológica Nacional
Maestro M. López esq. Cruz Roja Argentina
(X50165ZAA) Ciudad Universitaria, Córdoba, Argentina

Resumen. Desde hace ya algunos años los problemas en la especificación de los requerimientos ha sido un tema común en la industria del software. Para lograr minimizar el impacto de especificar incompleta o incorrectamente los requerimientos han surgido algunas técnicas y herramientas que permiten mitigar este problema. Las metodologías ágiles y los proyectos a gran escala no están ajenos a esta situación. Este trabajo presenta la definición e implementación de mejoras al proceso definido basado en prácticas de ingeniería de requerimientos con el fin de reducir el re-trabajo, proveer reportes objetivos respecto del estado del proyecto, incrementar la frecuencia de entregas del producto para obtener *feedback* (retroalimentación) y reducir así la cantidad de defectos reportados al final del proyecto, mejorando en consecuencia la satisfacción del cliente.

Palabras claves: requerimientos, historias de usuarios, especificación, trazabilidad, UI, agile a gran escala

1 Introducción

Desde hace ya algunos años los conceptos de requerimientos e ingeniería de requerimientos vienen trabajándose más en detalle para lograr satisfacer las necesidades de los clientes. Partiendo del concepto básico de requerimientos planteado por la IEEE [1] en donde los define como: (1) Una condición o capacidad requerida por un usuario para resolver un problema o alcanzar un objetivo; (2) Una condición o capacidad que debe ser poseída por un sistema o componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro tipo de documento formalmente impuesto; (3) Una representación documentada de una condición o capacidad según 1 y 2-, algunos autores mencionan que la consecuencia principal de los problemas en los requerimientos es el re-trabajo, rehacer algo que ya estaba hecho. El re-trabajo puede consumir entre el 30 – 50% del costo total de desarrollo del proyecto [2], y a su vez los errores en los requerimientos están asociados al 70-85% del costo total de re-trabajo [3] [4]. La Ingeniería de Requerimientos es un proceso cíclico que involucra: la elicitación, análisis,

especificación, verificación y validación de requerimientos [5]. Para ayudar a mitigar los errores en requerimientos estos procesos utilizan múltiples técnicas y perspectivas [6]. En particular y para este caso, el trabajo se enfoca en las técnicas de prototipado [7] [8], trazabilidad [9] [10] y casos de usos [6] [11].

Las metodologías ágiles no son la excepción respecto de la necesidad de gestionar requerimientos. Hoy en día es común el uso de metodologías agiles para el desarrollo de software [12] y en particular el uso del método Scrum [13], en cuyo marco surge también la necesidad de definición de requerimientos. En este caso el foco serán las historias de usuario [13]. Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario [13]. Están compuestas por una descripción escrita que será utilizada para planificar y posteriormente disgregar los detalles con el dueño del producto, las conversaciones propiamente dichas con el dueño del producto y las pruebas que han de determinar si las historias están finalizadas o no [13].

Como parte del ciclo de desarrollo de software, las pruebas del sistema son importantes a la hora de verificar la calidad de lo que se está entregando. Para verificar que dichos requerimientos son realmente implementados con la calidad esperada surge como herramienta la trazabilidad de requerimientos [14]. Trazabilidad se refiere a la habilidad de describir y seguir la vida de un requerimiento en ambas direcciones, es decir, desde su origen y especificación, hasta su implementación y uso, como así también durante su constante refinamiento [15]. Permite determinar que todos los requerimientos han sido completamente desarrollados, abarcando la relación con otros artefactos de trabajo. Es utilizada fundamentalmente cuando es preciso evaluar el impacto de los cambios en las actividades del proyecto o en los artefactos de trabajo [16].

El presente trabajo se desarrollará de la siguiente manera; sección 2 Contexto y objetivos generales; sección 3 Mejoras propuestas al proceso definido; sección 4 Resultados Obtenidos; sección 5 Conclusiones y Futuras actividades.

2 Contexto

Este trabajo expone la experiencia en la definición e implementación de mejoras propuestas al proceso definido utilizando prácticas de ingeniería de requerimientos teniendo en cuenta conceptos de metodologías ágiles basada en definiciones de interfaces de usuario (*specs*) provistas por el cliente, para un proyecto de *infotainment*[1] que implica el desarrollo del sistema de entretenimiento e información para automóviles que se nutre de la interacción entre el hombre y la maquina (*Human Machine interface*) [17] [18].

[1] Infotainment es un tipo de medio que intenta combinar información educativa o útil y contenido entretenido. Infotainment está diseñado para ayudar a promover la adquisición de información específica, habilidades u oficios en un formato que atraiga a los usuarios. [31]

El objetivo de este proyecto a gran escala [19] [10] [20] es el desarrollo de un sistema de *Infotainment* de un automóvil, tomando como base la plataforma Android 5.1 [21], generando a partir de allí las distintas aplicaciones que estarán disponibles en la computadora central de dicho automóvil, tales como el teléfono, aplicaciones de mensajería, navegación por mapas, radios AM y FM, entre otros. Para el desarrollo del proyecto se cuenta con una estructura de 9 equipos distribuidos en distintas localizaciones alrededor del mundo, y 2 equipos dedicados a la validación y entrega final del producto (RMT). Cada equipo se gestiona utilizando la metodología ágil SCRUM [13]. Pero además, en la estructura existen roles definidos de más alto nivel que permiten la coordinación de estos equipos, tanto desde el punto de vista técnico como del de gestión.

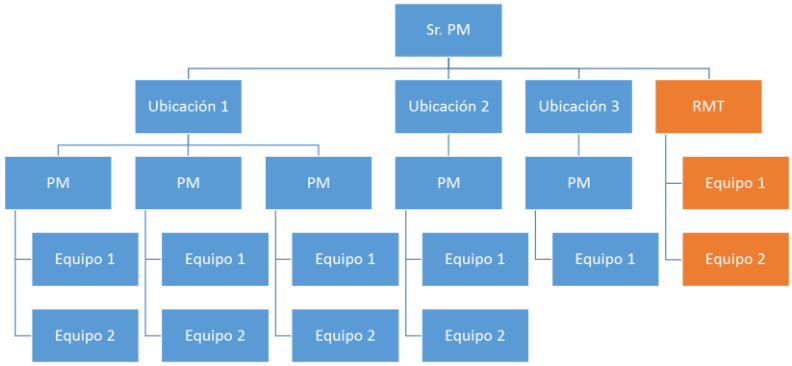

Fig. 1. Coordinación equipos distribuidos

El proyecto está dirigido por dos actores principales: la automotriz, encargada de sacar una nueva línea en sus vehículos, y una empresa integradora, contratada para llevar adelante la arquitectura, diseño e implementación del componente de software para el sistema de *Infotainment*.

La gestión es llevada adelante con el soporte de una serie de herramientas tales como Rational DOORS Next Generation [22] para la gestión de requerimientos; Repo [23], para manejo de los distintos repositorios que se utilizan para gestionar el código; JIRA [19] tanto para la gestión del proyecto a través de la definición de historias de usuarios para el seguimiento del progreso del desarrollo, gestión de defectos y definición de casos de prueba; Slack/Skype [25] [26] herramientas utilizadas como medios de comunicación, SVN [22] repositorio utilizado para el registro de métricas y entregables de proyecto; y Confluence [28] herramienta utilizada para guardar la documentación del proyecto tales como planes, guías, manuales de usuarios, notas de entregas, etc.

En proyectos anteriores, ambas compañías se enfrentaron con problemas derivados de una definición, documentación y seguimiento de los requerimientos de software poco eficientes [11] [29] [30]. Estos problemas pueden resumirse en dos grandes grupos: 1) falta de completitud en la implementación de las funcionalidades comprometidas; y 2) falta de visibilidad acerca del progreso real del proyecto. Estos problemas se describirán en detalle en la siguiente sección.

Con miras de dar una solución a estos dos problemas, surgió la iniciativa de mejorar el proceso de documentación de los requerimientos, desde la especificación de los mismos hasta el planteo de una trazabilidad con elementos de desarrollo y prueba para monitoreo de progresos y análisis de impacto. Los principales objetivos de esta iniciativa fueron los de proveer reportes objetivos respecto del estado del proyecto, incrementar la frecuencia de entregas del producto para obtener *feedback* (retroalimentación) y reducir así la cantidad de defectos reportados al final del proyecto, y como consecuencia final mejorar la satisfacción del cliente.

2.1 Proceso anterior

La empresa automotriz define y comunica sus requerimientos de software a través de la distribución de especificaciones (*specs*) de UI de las distintas pantallas del sistema. Cada documento provisto por la automotriz se corresponde con una aplicación del sistema (radios, mensajería, teléfono, navegación, etc). Estos *specs* definen tanto la organización (*layout)* de la pantalla (qué componentes de UI están presentes, ubicación y tamaño) como también los distintos flujos posibles de acuerdo a la interacción del usuario con cada uno de los componentes de la pantalla.

En el proyecto anterior, se tomaban estos *specs* y se derivaban elementos de trabajo para que los desarrolladores implementaran cada una de las pantallas. Los elementos de trabajo se documentaban como tareas en JIRA [24], y los distintos equipos iban tomando los distintos elementos de trabajo (*ítems*) e implementando las pantallas a partir de ellos.

Para el desarrollo de las pruebas, se tomaban como entrada los *specs* de interfaz de usuario, y se generaban con ellos conjuntos de casos de prueba por aplicación, que se documentaban también en JIRA [24]. Estas pruebas tenían una correlación directa con

las aplicaciones, pero no con los elementos de trabajo que se utilizaban para implementar las distintas pantallas.

Dado que no había una trazabilidad entre los *specs* (requerimientos), elementos de trabajo y casos de prueba, la ejecución de las pruebas se realizaba cada cierto tiempo (prefijado en el plan del proyecto, típicamente cada dos o tres meses) e involucraba la ejecución de todos los casos de pruebas desarrollados hasta el momento, en una regresión completa de pruebas de software. A partir de la ejecución de las pruebas se generaba un reporte de verificación que incluía los resultados de las pruebas y la lista de defectos identificados durante la ejecución de las mismas.

Esta estrategia trajo aparejadas algunas desventajas e inconvenientes:

1- Completitud: el desarrollo del código se basaba en la descripción de las pantallas solamente sin tener en cuenta los distintos casos de usos para las mismas. Esto produjo que ciertos flujos (o casos) no fueran tenidos en cuenta a la hora del desarrollo. Cuando un flujo no era tenido en cuenta el equipo de pruebas creaba defectos para cada uno de estos flujos faltantes. Esto hizo que la cantidad de defectos creciera de manera descontrolada.

2- Visibilidad: Debido a la cantidad de defectos generados, se hizo muy difícil conocer el estado real del proyecto a un momento dado: la implementación de las funcionalidades se había declarado completa, pero sin embargo la cantidad de flujos no implementados (reportados en este punto del proyecto como defectos) demostraba que el avance real era ciertamente menor.

3 Mejoras propuestas al proceso definido

En base a los problemas descriptos anteriormente, se planteó una alternativa que diera solución a estos inconvenientes para el proyecto actual.

Tomando como punto de partida los documentos de especificación de interfaz de usuario enviados por el cliente como entrada principal al proceso de requerimientos y teniendo en cuenta los conceptos básicos relacionados a SCRUM [31], historias de usuario [13], trazabilidad [32] [10] [33], y pruebas [34], se define a continuación el proceso propuesto:

Cada documento de especificación de interfaz de usuario es descompuesto en una serie de requerimientos de software o Flujos. Como herramienta de soporte se decidió utilizar DOORS Next Genereation [22], después de una evaluación de diversas herramientas se determinó que DOORS es que es una herramienta de gestión y desarrollo de requerimientos. Tiene conceptos de línea base, confección de diagramas, prototipos, etc. Asimismo, permite la adaptación a las necesidades propias del proyecto y generar una trazabilidad (unidireccional-DOORS a Jira) al a través de la creación de hipervínculos a sitios externos; para mantener las interfaces de usuario, los mapas [2]de cada aplicación y la descripción de cada uno de los flujos como requerimientos formales.

[2] Los mapas también conocidos como *applications maps*, son un esquema donde se muestran todas las pantallas de una aplicación, con sus respectivas líneas de flujo, representando las transiciones de pantalla de acuerdo a distintos eventos, como acciones del usuario final, tiempo transcurrido (en algunos casos), etc.

Cada pantalla y cada mapa del documento de especificación de interfaz de usuario es creado como un artefacto en la herramienta, tarea que lleva adelante un equipo de Analistas de Negocios, creado con este propósito. También dicho equipo es responsable de la descripción de los flujos para describir requerimientos de software funcionales y no funcionales derivados de este documento.

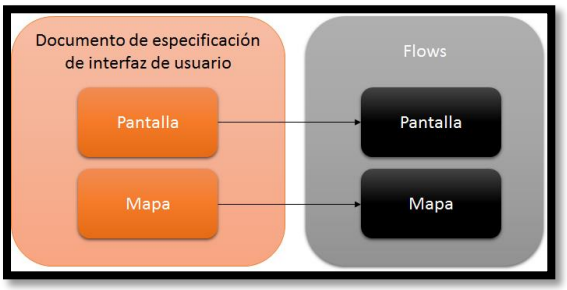

Fig. 2. Derivación de requerimientos en base al documento de especificación de Interfaz de usuario

Luego de la creación del documento de especificación de interfaz de usuario se inicia el proceso de revisión. El propósito de este proceso es verificar la correctitud y completitud de los requerimientos de software derivados del documento de especificación de interfaz. Los arquitectos del producto y los líderes técnicos participan en el proceso de revisión de los requerimientos generados, principalmente para asegurar que se han cubierto todos los flujos del sistema. Una vez que todos los comentarios de la revisión han sido corregidos y aprobados el equipo procede a la creación de una línea base. Tanto el proceso de revisión como el de línea base se realizan en DOORS Next Generation [22].

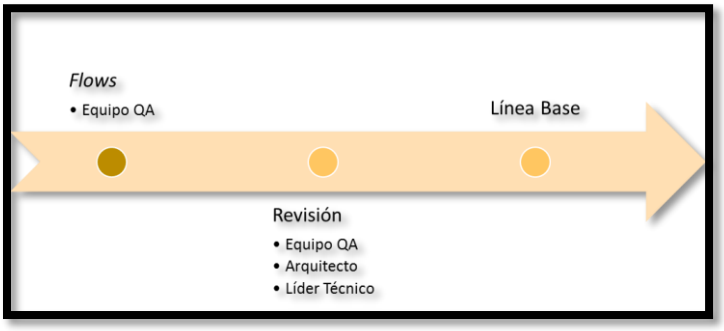

Fig. 3. Proceso de creación de requerimientos

3.1.1 Trazabilidad

Se definieron dos tipos de trazabilidades tomando como base los conceptos explicados en [33]:

1) Trazabilidad horizontal: en la herramienta se utilizó el tipo de enlace "***Embedded In/Embeds***" (Incluído en / Incluye) para registrar la trazabilidad de los mapas a los flujos funcionales , y se utilizó el tipo de enlace "***Screen/Screen for***" (Pantalla / Pantalla para) para registrar la trazabilidad de los flujos a las pantallas

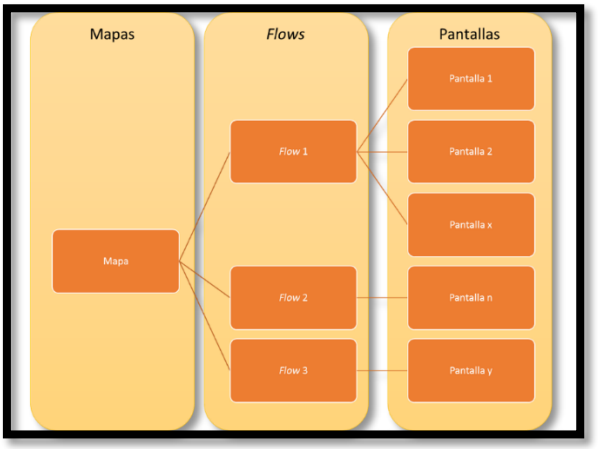

Fig. 4. Trazabilidad Horizontal

2) Trazabilidad vertical: La trazabilidad vertical permitirá asegurar que todos los requerimientos han sido implementados, y además han sido verificados y validados mediante las distintas estrategias de prueba. Para verificar que todos los requerimientos han sido implementados, se mantendrá la trazabilidad de requerimientos (flujos) a historias de usuario y tareas en JIRA [24], lo que también permitirá al equipo cuantificar el progreso del desarrollo. Para verificar que todos los requerimientos han sido probados, se utilizará el tipo de enlace ***"Tested by test case"*** (Verificado por caso de prueba), para asociar los requerimientos (flujos) a los casos de prueba.

Ambos tipos de trazabilidades se mantienen en Rational DOORS Next Generation [22]

3.1.2 Gestión de cambios de los requerimientos

Cuando una nueva versión del documento de especificación de interfaz de usuario es enviada por el cliente, los analistas de negocio analizan el impacto de los cambios y se procede a la actualización de los requerimientos en DOORS [22]. Para ello, se sube el nuevo mapa de cada aplicación. Cada mapa tiene una marca que etiqueta los flujos que se han agregado, modificado o eliminados, las que sirven de base para que el equipo de testing refleje dichos cambios en los requerimientos correspondientes. Cada requerimiento actualizado se marca con una nueva versión, y una vez actualizados todos los requerimientos, se genera una nueva línea base de los requerimientos en la herramienta. Luego, en base a los requerimientos actualizados, se crean nuevas historias de usuario

en la pila del producto dentro de JIRA [24] que luego son estimadas, planificadas e implementadas en el código. A su vez, en base a los cambios de requerimientos se actualizan los casos de prueba y se analizan los cambios (si corresponden) en las suites de pruebas de sanidad y regresión.

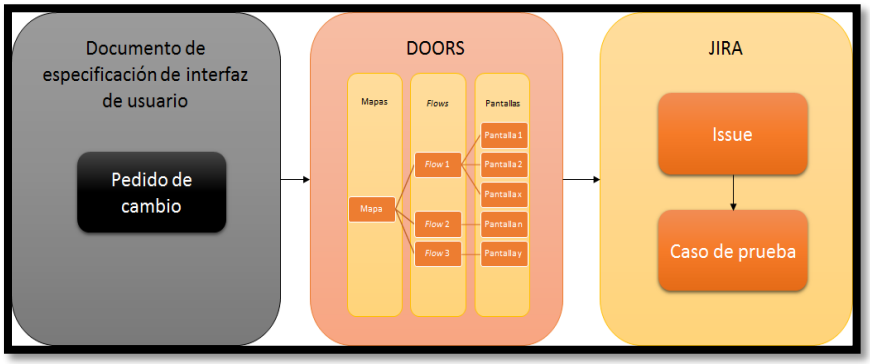

Fig. 5. Gestión de cambios de los requerimientos

4 Resultados obtenidos

La implementación y definición de estas mejoras surge, como se mencionó anteriormente, de la necesidad de mejorar la gestión del alcance del proyecto en base a una descripción más clara de los requerimientos y su correspondiente trazabilidad a elementos de trabajo y casos de prueba, con la consiguiente disminución en la cantidad de defectos reportados y la capacidad de planificar ciclos de pruebas en base al análisis de impacto de los cambios realizados. A continuación se detallan los resultados obtenidos teniendo en cuenta los dos grandes problemas identificados anteriormente. También se menciona cómo la implementación de estas mejoras influyó en la percepción del cliente respecto de la calidad del producto entregado.

4.1 Completitud - Trazabilidad de requerimientos a elementos de trabajo

El proceso de documentar los flujos de cada aplicación en forma de requerimientos determinó la necesidad de generar una pila de trabajo priorizada con cada uno de los elementos de trabajo necesarios para asegurar que cada uno de dichos requerimientos fuera implementado. Esta pila de trabajo se mantiene en JIRA [24] en formato de historias de usuario [31], manteniendo una convención de nombres que permite luego agrupar elementos (tanto historias de usuario como defectos) por aplicación. Así, se pudo definir una trazabilidad de requerimientos a elementos de trabajo, que permite monitorear de manera efectiva el progreso del proyecto respecto a requerimientos implementados, tal como se puede apreciar en el siguiente gráfico.

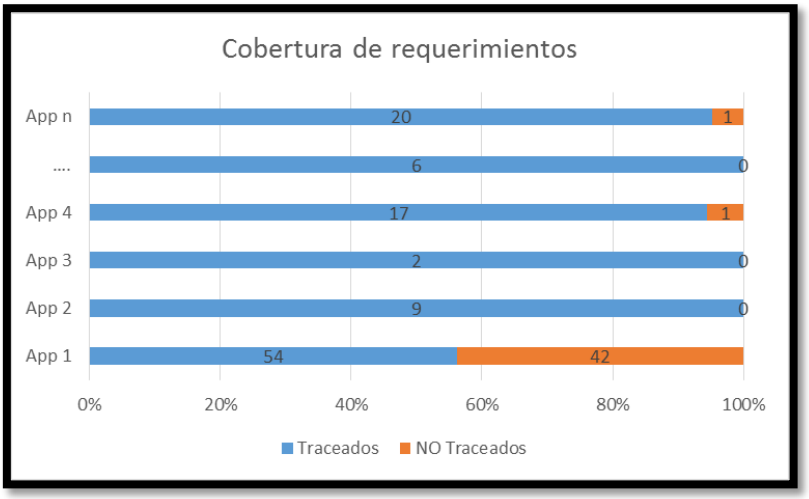

Fig. 6. Resultados obtenidos / Cobertura de requerimientos - Completitud

Esta métrica se reporta semanalmente, con el objetivo de monitorear de manera correcta el desarrollo de la funcionalidad.

Así, se logró dar visibilidad del progreso del proyecto a un nivel más granular de producto, pasando de un nivel "de aplicación" a un nivel de "flujos de una aplicación". De esta manera, cada entrega parcial puede incluir flujos de distintas aplicaciones, y esto se puede comunicar claramente a todos los *stakeholders* del proyecto.

4.2 Visibilidad - Disminución en la cantidad de defectos

El control de cambios implementados a través de los requerimientos dio una mejor visibilidad en el progreso de la implementación de los requerimientos, lo cual tuvo gran impacto en la cantidad de defectos generados.

Al haber una visibilidad clara de casos de usos o flujos están involucrados en que requerimientos estarían implementados en cada entrega parcial del producto y qué requerimientos no, se disminuyó la cantidad de defectos reportados a causa de un mal entendimiento del alcance entregado: es decir, asumir que ciertas aplicaciones estarían totalmente implementadas cuando en realidad sólo algunos flujos de dichas aplicaciones formaran parte de la entrega parcial.

Por otra parte, dado el conocimiento del alcance a un nivel más detallado permitió generar un plan de entregas desde el principio, mejorando no sólo una distribución eficiente del trabajo en los equipos de desarrollo, sino que además permitió que el equipo de pruebas y calidad de software pudiera involucrarse en las etapas tempranas del desarrollo, retroalimentando al equipo de desarrollo con comentarios sobre las distintas implementaciones antes incluso de que estos comentarios se convirtieran en defectos del software.

Estos dos puntos, la visibilidad detallada del alcance de cada entrega, y el involucramiento del equipo de pruebas en etapas tempranas del desarrollo, permitieron una disminución considerable en los defectos del producto reportados. El siguiente gráfico muestra la cantidad de defectos que se esperaban encontrar, estimados al comenzar el proyecto, comparado con el número de defectos que en realidad se fueron reportando a lo largo del tiempo.

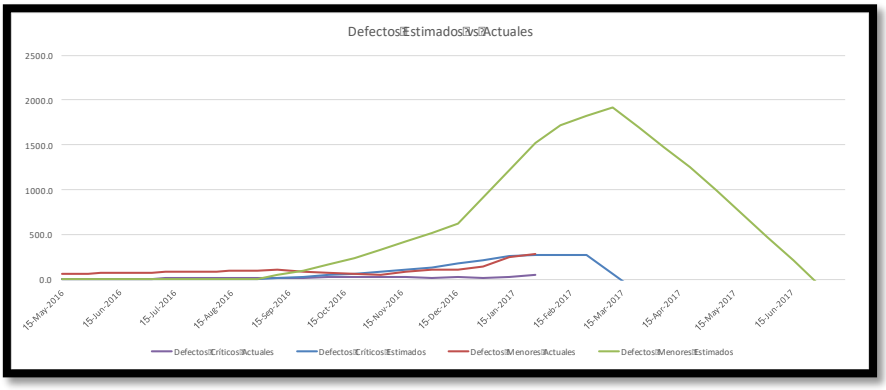

Fig. 7. Resultados obtenidos / Defectos Estimados vs. Actuales - Visibilidad

Las líneas azul y verde representan el número estimado de defectos hasta el final del proyecto, mientras que las líneas violeta y roja muestran el número de defectos realmente reportados hasta el momento.

Las líneas violeta y azul corresponden a defectos de severidades altas (Críticas y Mayores), mientras que las líneas roja y verde corresponden a defectos de severidades más bajas (Menores y Cosméticos).

Para ambos tipos de severidades se puede apreciar claramente como la tendencia en los defectos encontrados mejoró en alrededor de 10 veces el número originalmente estimado.

4.3 Percepción del cliente

Los cambios implementados en el proceso de gestión de requerimientos no solamente mejoraron la gestión del alcance y control de los defectos del producto, sino que también mejoraron la percepción del cliente respecto del trabajo realizado por el equipo.

Esta percepción se ve reflejada en una encuesta periódica que se envía a los clientes de los distintos proyectos, en la que se requiere la evaluación del equipo respecto de su funcionamiento y performance, enfocándose en cuatro aspectos principales: cumplimiento en tiempo y forma con los entregables, tiempo de respuesta para los pedidos de cambio, calidad en los productos / servicios provistos y documentación provista; esto permite identificar y atacar de manera temprana las desviaciones que se pudieran presentar. Los valores para clasificar los criterios son: 1) Muy insatisfecho, 2) Insatisfecho,

3) Normal, 4) Cumple con las expectativas, 5) Excede las expectativas. Los resultados obtenidos se observan en el siguiente gráfico:

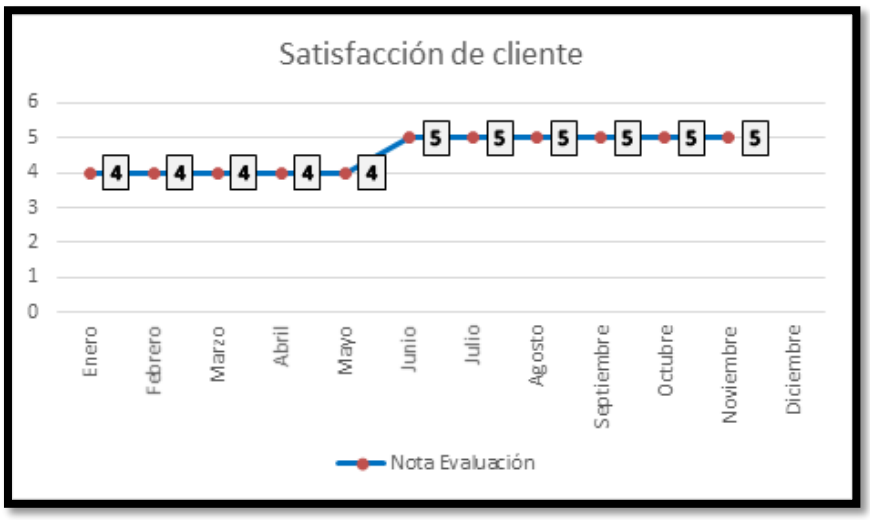

Fig. 8. Resultados obtenidos / Satisfacción del cliente

Como puede observarse en el gráfico, luego de algunos meses de aplicación de las mejoras identificadas, puede observarse una mejora (y mantenimiento) de la satisfacción del cliente respecto del desempeño del proyecto.

5 Conclusiones y futuras actividades

La implementación de la gestión de requerimientos permitió implementar mejoras en el proceso de desarrollo de software para un sistema de *infotainment* de un automóvil principalmente en dos aspectos:

1) Un mejor entendimiento del alcance del proyecto tanto de la implementación de las distintas funciones como de la validación del nivel de calidad (mediante pruebas) de dichas funciones. Esto se logró a través de la instauración de trazabilidad horizontal (*specs* de interfaces de usuario a descripciones de requerimientos funcionales) y vertical (a través de la relación de requerimientos a elementos de trabajo, y a pruebas formales, para asegurar que todos los requerimientos han sido debidamente implementados y validados).
2) Mejora en la visibilidad en el progreso del proyecto, dada la posibilidad de generar reportes de avance respecto de la implementación de los requerimientos y su validación, pero principalmente en base a la disminución fundamental en la cantidad de defectos. Esto último se logró a partir de la posibilidad de ejecutar pruebas formales enfocadas en los requerimientos que iban siendo implementa-

dos durante etapas tempranas del proyecto, y a partir de la disminución de defectos reportados debido a funcionalidades no implementadas (pero que en el pasado era reportados como "completas").

Como consecuencia final también se logró la mejora en la percepción del cliente respecto de la calidad del producto desarrollado.

Además de las ventajas de la gestión de requerimientos y el mantenimiento de una trazabilidad descriptas en el presente trabajo, es de esperar que se pueda hacer uso de la información provista por la trazabilidad para mejorar los procesos de planificación de las pruebas de sistema. Hoy en día, la validación del producto en cualquiera de sus fases implica la ejecución de todos los escenarios de prueba definidos. Con una trazabilidad de requerimientos a elementos de trabajo y a casos de prueba, se espera a futuro poder planificar la ejecución de las pruebas de sistema a partir de un análisis de impacto de los cambios realizados, utilizando la trazabilidad de requerimientos como base de información.

6 Referencias

[1] S. C. C. o. t. C. S. o. t. IEEE, «IEEE,» 1990.

[2] B. B.W. y P. P.N., «Understanding and controlling software costs,» 2002.

[3] D. Leffingwell, «Calculating the Return on Investment from More Effective Requirements Management,» 1997.

[4] J. O. Grady, «SYSTEMS ENGINEERING REALISATION,» 1999.

[5] J. W. Hans van Vliet, «Software Engineering: Principles and Practice. Second Edition.».

[6] K. Pohl, «Requirements Engineering,» 2011.

[7] IEEE, «Guide to the Software Engineering Body of Knowledge,» SWEBOK, 2004.

[8] M. Madigan, «StorageTek Manager, PAL Engineering. Requirements Elicitation.,» [En línea]. Available: http://ece-www.colorado.edu/~swengctf/standalone/presentations/293,20,Requirements Elicitation Guidelines1.

[9] B. y. J. Ramesh, Toward Reference Models for Requirements Traceability., 2001.

[10] S. W. (. Ambler, «Agile Modeling Best Practices,» 2013.

[11] K. Wiegers, More About Software Requirements, Microsoft Press, , 2006.

[12] J. L. P. &. P. M. C. (. Canós, «Métodologías Ágiles en el Desarrollo de Software.».

[13] M. (. Cohn, «Mountain Goat Software. (Mountain Goat Software),» 2012. [En línea]. Available: http://www.mountaingoatsoftware.com/topics/user-stories.

[14] ISO/IEC/IEEE, «IEEE Recommended Practices for Software Requirements Specifications,» 2011.

[15] O. G. y. A. Finkelstein, «An Analysis of the Requirements Trazability Problem,» 1994.

[16] S. E. Institute, «CMMI for Development, version 1.3,» 2010.

[17] W. Hardin, «Human Machine Interface (HMI) Software Information,» 21 04 2016. [En línea]. Available: http://www.globalspec.com/learnmore/industrial_engineering_software/industrial_controls_software/human_machine_interface_software_hmi.

[18] V. S. H. S. John J. Pannone, «Human Machine Interface Systems,» 20 9 2010. [En línea]. Available: https://www.pddnet.com/article/2010/09/human-machine-interface-systems.

[19] B. Fitzgerald, Scaling Agile Methods to Regulated Environments: An Industry, 2013.

[20] C. y. B. V. Larman, Practices for Scaling Lean & Agile Development, Boston, 2010.

[21] L. Android 5.0. [En línea]. Available: https://www.android.com/versions/lollipop-5-0/.

[22] IBM, «IBM Rational DOORS Next Generation,» [En línea]. Available: http://www-03.ibm.com/software/products/en/ratidoorng.

[23] «Repo command reference,» [En línea]. Available: https://source.android.com/source/using-repo.

[24] JIRA. [En línea]. Available: https://www.atlassian.com/software/jira?_mid=3acccfc0e3fed62afa79833dc1a0263b&aceid=&adposition=1t1&adgroup=9124289302&campaign=189412582&creative=179607921977&device=c&keyword=jira&matchtype=e&network=g&placement=&gclid=Cj0KEQiAuonGBRCaotXoycysvIMBEiQAcx.

[25] Slack. [En línea]. Available: https://slack.com/.

[26] Skype. [En línea]. Available: https://www.skype.com/en/.

[27] T. SVN. [En línea]. Available: https://tortoisesvn.net/.

[28] Attlasian, «Confluence,» [En línea]. Available: https://www.atlassian.com/software/confluence.

[29] I. S. a. P. Sawyer, Requirements Engineering – A good practice guide, 1997.

[30] C. R. Klaus Pohl, Requirements Engineering fundamentals, O'Reilly Media, 2011.

[31] S. Alliance, «Scrum Alliance,» 2013. [En línea]. Available: http://www.scrumalliance.org/.

[32] J. (. Ibañez, «Gestión de requerimientos IV: trazabilidad.,» 2012.

[33] CMMI, «CMMI® for Development, Version 1.3,» http://www.sei.cmu.edu/reports/10tr033.pdf , 2010.

[34] E. V. y. G. D. R. Black, «ISTQB Certification - The Foundations of Software Testing,» United Kingdom, Cengage Learning, 2009., 2009.

[35] Techopedia, «Infotainment,» [En línea]. Available: https://www.techopedia.com/definition/5520/infotainment.

[36] C. y. V. B. Larman, Scaling Lean & Agile Development., Boston :: Addison-Wesley,, 2009.

[37] C. y. V. B. Larman, Large-Scale Scrum. Practices for Scaling Lean & Agile, 2010.

[38] Gitiles, «Gerrit Code Review,» [En línea]. Available: https://www.gerritcodereview.com/.