Universidad de San Andrés

Escuela de Negocios

Maestría en Gestión de Servicios Tecnológicos y Telecomunicaciones

Metodologías ágiles y trabajo en red para asegurar el correcto desarrollo e implementación de software : estudio del sector de e-commerce en gran empresa de retail en Argentina

Autor: Lascano, Diego Mario

Legajo : 35423444

Director/Mentor de Tesis: Artopoulos, Alejandro

2018

Universidad de
SanAndrés

Maestría en gestión de servicios tecnológicos y de telecomunicaciones

Metodologías ágiles y trabajo en red para asegurar el correcto desarrollo e implementación de software. Estudio del sector de e-commerce en gran empresa de retail en Argentina

Alumno: Ing. Lascano Diego Mario

Tutor: Dr. Alejandro Artopoulos

2018

Buenos Aires

`

AGRADECIMIENTOS

Un millón de gracias y toda una vida de felicidad, a la única persona del mundo que siempre estará conmigo en las buenas y en las malas, mi Mamá.

RESUMEN

Debido a que los negocios y la tecnología cambian a una tasa sin precedentes, la celeridad en el desarrollo de software para poder atender los cambios del negocio se ha vuelto cada vez más crítica. Los enfoques de desarrollo de software ágil, han sido adoptados por un número creciente de organizaciones para mejorar su agilidad de desarrollo de software y de este modo poder dar respuesta rápidamente a los requerimientos del negocio. Sin embargo, esta vorágine de cambios a productos o servicios existentes y la introducción de nuevas innovaciones sin un análisis detallado de los riesgos y cómo éstos afectan al ambiente de control, aumenta la probabilidad de que se materialicen los mismos y puedan desestabilizar el negocio y por lo tanto a la organización. Este trabajo busca lograr un entendimiento sobre en qué momentos y bajo qué circunstancias ocurren errores en el diseño y desarrollo del software que conllevan a la existencia de errores en el producto final y cuáles deben ser el modelo cooperativo y los artefactos de coordinación para evitar tal fin.

PALABRAS CLAVE: Desarrollo de software, comunicación, errores en el software productivo, metodológica de desarrollo del software.

Contenido

Índice de ilustraciones

`

Índice de tablas

1. Introducción

El nacimiento y posterior expansión de las empresas tecnológicas está dado por el grado de innovación que cuentan sus productos o servicios. Por este motivo una innovación no es suficiente para mantener a una empresa tecnológica en la cima, sino que se requiere de gran cantidad de innovaciones en forma de nuevos desarrollos o modificaciones a un servicio existente.

Por otra parte, el surgimiento de nuevas tecnológicas y servicios enfocados a millones de usuarios generan continuamente nuevos riesgos que atentan a varios factores críticos del negocio entre ellos la confidencialidad de la información (debido a los datos sensibles que se cuentan de usuarios). Dichos riesgos si son transportados al producto final podría generar errores materiales, omisiones, o fraude pudiendo pasar desapercibidos por meses antes de que sean detectados por una auditoría y por lo tanto con el daño ya realizado.

La problemática está dada en que al momento de innovar, se tienen en cuenta innumerables factores que resultan indispensables para lograr el éxito, como por ejemplo interfaz de usuarios, mapas de calor para el seguimiento de las acciones del usuario, distribución del contenido, entre otras cosas, pero no se tiene en cuenta un análisis asociado a los nuevos riesgos que dicha innovación puede introducir independientemente de los marcos regulatorios que se encuentren aplicados. Adicionalmente, el riesgo inherente asociado a la industria obliga a las empresas tecnológicas a innovar en un menor tiempo, lo que se traduce usualmente en menor tiempo para las pruebas de la solución desarrollada.

Por este motivo, existen múltiples casos en los que la introducción de un nuevo servicio o modificación de uno existente produce pérdidas millonarias a empresas de tecnología ya sea en forma de robo de información, incorrecto registro de asientos contables o calculando incorrectamente impuestos, entre otros, pudiendo ocasionar la quiebra de la empresa o sanciones para aquellas que estén reguladas.

1.1 Objetivos y alcance

El presente trabajo de investigación tiene el siguiente alcance se encuentra delimitado por los siguientes objetivos:

- Analizar el proceso de evolución del desarrollo del software, considerando sus ventajas y desventajas.
- Identificar cuáles son las metodologías de desarrollo del software utilizadas en la actualidad y sus características principales.
- Lograr un entendimiento de la teoría del actor-red y su relación con el proceso de desarrollo del software.
- Verificar el grado de conocimiento y la manera en que se encuentra implementada la metodología de desarrollo del software por los equipos de desarrollo del software.
- Analizar bajo la esquematización con la teoría del actor-red, los elementos involucrados y los flujos de comunicación entre los distintos participantes del equipo de desarrollo del software.

- Reflexionar sobre el proceso de desarrollo, y la comunicación entre el equipo de desarrollo de la solución y otras áreas, a fin de identificar dónde pueden ocurrir falencias que introduzcan errores en el producto final.

1.2 Hipótesis

La introducción de errores en los productos de software en empresas de tecnología, son generados por una incorrecta implementación de la metodología ágil de desarrollo, debido a problemas en la comunicación entre los miembros del equipo de desarrollo y con otras áreas, o una combinación de ambas.

1.3 Preguntas de investigación

¿Son los errores en los productos finales del software, generados por la falta de conocimiento o incorrecta implementación de la metodología de desarrollo?

¿Existen falencias en la comunicación entre los miembros del equipo que desarrollan el software, y la comunicación con otras área del negocio que favorecen la ocurrencia de errores en los productos finales de software?.

2. Marco teórico

2.1 Metodología de desarrollo

2.1.1 Proceso de desarrollo del software

La definición del proceso de desarrollo del software se encuentra ampliamente definido académicamente debido a la importancia del mismo. Según (Pressman, 2010), un proceso de desarrollo del software "es una estructura para las actividades, acciones y tareas que se requieren a fin de construir un software de calidad". Otros autores definen al proceso de software como la manera en la cual se crea o mantiene el software incluyendo las fases preliminares de planeación, modelado, prueba y puesta en producción, las cuales se deben cumplir para asegurar la realización del software (Bender, 2012).

En la actualidad existen dos metodologías que permiten ejecutar un proceso de software: la metodología tradicional y la metodología ágil de desarrollo, ambas co-existen y son implementadas por las empresas en base a su experiencia y sus procesos de negocio.

2.1.2 Metodología tradicional de desarrollo del software

Las metodologías como R.U.P., Cascada, o V-Model son clasificadas dentro del universo de tradicionales (M.T.D.S.) y resultan ser las más conocidas por los ingenieros y personal técnico (Nikiforova, Nikulsins, & Sukovskis, 2009). Estas metodologías están basadas en una secuencia de pasos ordenados para poder desarrollar un software de calidad. Las metodologías para el desarrollo del software tradicionales requieren que se

defina y documente gran cantidad de los requerimientos de usuarios al comienzo del proyecto debido a que el éxito del mismo se encuentra estrechamente ligado a cantidad de cambios que se deban realizar durante el proyecto (IBM Corporation, 2007).

2.1.3 Metodología ágil de desarrollo del software

Para poder obtener un entendimiento completo sobre el concepto de metodología ágil de desarrollo (M.A.D.S.), es importante analizar individualmente las palabras que lo conforman: por un lado el término ágil y por otro la metodología de desarrollo del software.

Algunos autores definen el término ágil como la capacidad para reaccionar a los cambios en su entorno más rápido que a la tasa de estos cambios (Highsmith J., 2002), así como también representa la capacidad de crear rápida y flexiblemente y responder al cambio en la organización y dominios técnicos (Lee & Xia, 2010).

Por otro lado, la metodología de desarrollo del software brinda un marco de trabajo definido a través de un subconjunto de procesos predeterminados que, complementado con la documentación, guía el desarrollo del software asegurando su calidad al fin del proyecto (IBM Corporation, 2007).

Para que una metodología sea considerada como ágil, debe cumplir con el manifiesto de desarrollo ágil, el cual fue desarrollado por los primeros practicantes en el 2001. El manifiesto revela qué elementos son considerados valiosos y determinan cuando una metodología es considerada como ágil.

El siguiente gráfico evidencia como los elementos de la metodología tradicional son reemplazados por nuevos elementos para la metodología ágil.

Se adjunta a continuación la Figura 1, que detalla el manifiesto ágil versus el enfoque tradicional.

Figura 1: Resumen del manifiesto de metodologías ágiles. Elaboración propia en base a Agile Manifiesto. Octubre 2011.Web. 23 de Febrero 2018.

No existe una única definición de M.A.D.S., sino que varios autores han abordado el tema realizando una priorización semántica para destacar los elementos que consideran indispensables para el éxito de la metodología.

Según (Henderson-Sellers & Serour, 2005) la M.A.D.S. implica tanto la capacidad de adaptarse a los diferentes cambios así como también modificar el desarrollo según sea necesario. Por otro lado, (Conboy, 2009) lo define como la preparación continua para crear rápida o intrínsecamente el cambio, adoptar el cambio de manera proactiva o reactivar y aprender del cambio mientras se contribuye a percibir el valor agregado para el cliente a través de sus componentes colectivos y relaciones de entorno. Una definición más precisa y técnica podría ser la que propone (Lee & Xia, 2010), que define a la M.A.D.S. como la capacidad que tiene un equipo de desarrollo de software para responder eficaz y eficientemente e incorporar los requerimientos del usuario durante el ciclo de vida del proyecto.

2.1.4 Comparación M.T.D.S. vs M.A.D.S.

Existen muchas características que difieren a ambas metodologías según Nerur, Mahapatra & Mangalaraj (Nerur, Mahapatra, & Mangalaraj, 2005), consideran que el M.T.D.S. sistematiza el desarrollo del software haciendo al sistema totalmente especificable, predecible y es construido a través de una planificación minuciosa y extensa incorporando grandes equipos de trabajo, mientras que el M.A.D.S. busca realizar un "software de adaptación" de alta calidad, desarrollado por pequeños equipos de trabajo (brindándoles flexibilidad y velocidad), utilizando los principios de mejora continua del diseño y pruebas, basadas en la retroalimentación rápida y el cambio.

Otros autores como T.Dyba y T.Digsoyr (Dyba & Dingsoyr, 2009) analizan las diferencias entre las metodologías en base al contexto. Siendo que las M.A.D.S. abordan el desafío de un mundo impredecible y la posibilidad de cambios en los requerimientos

del software durante el proyecto, haciendo hincapié en la competencia de las personas involucradas y sus relaciones para resolver los inconvenientes inesperados, mientras que la M.T.D.S. se basa en un mundo predecible y acotado, el cual es un requisito esencial inmodificable que es utilizado para poder definir los requerimientos del sistema antes de comenzar a realizarlo y con una baja capacidad de respuesta ante los cambios inesperados.

Se adjunta a continuación la Figura 2, que realiza un comparativo entre las fases de desarrollo del software.

Figura 2: Comparativo fases de desarrollo del software enfoque tradicional vs enfoque ágil. Elaboración propia

A continuación se puede observar un cuadro comparativo en el que se identifican aspectos que comparten ambas metodologías y como difieren estos:

Tabla 1: Cuadro comparativo sobre las metodologías de desarrollo del software

Aspectos	**M.T.D.S.**	**M.A.D.S**
Enfoque	Predictivo	Adaptativo

Medición del éxito	Acorde a lo planeado	Valor del negocio
Tamaño del proyecto	Grandes	Pequeños
Gerenciamiento	Autocrático	Descentralizado
Perspectiva del cambio	Cambio sostenible	Cambio adaptable
Cultura	Monitoreo y control	Liderazgo y colaboración
Documentación	Gran documentación	Poca documentación
Énfasis	Orientada a los procesos	Orientada a las personas
Ciclos	Limitados	Numerosos
Dominio	Previsible	Impredecible / exploratorio
Planificación por adelantado	Exhaustiva	Mínima
R.O.I.	Al final de proyecto	En el comienzo del proyecto
Tamaño del equipo	Grande	Pequeño / Creativo

Tabla 1: Cuadro comparativo sobre las metodologías de desarrollo del software.

Elaboración propia en base a Comparsion Summary of Plan-Driven and Agile Principles (Rubin, 2012).

2.1.5 Principales marcos de trabajo y prácticas basadas en M.A.D.S.

En la actualidad existen gran cantidad de marcos de trabajo y prácticas para la metodología ágil de desarrollo, las cuales difieren en el foco que se desea implementar a la hora de llevar a cabo un proyecto.

De los marcos de trabajo y prácticas para el desarrollo ágil utilizados en la industria se destacan Scrumy XP Hybrid, los cuales son utilizados por la mayoría de las empresas de tecnología alrededor del mundo.

En la figura 3, se observan las metodologías ágiles utilizadas en la actualidad a nivel mundial.

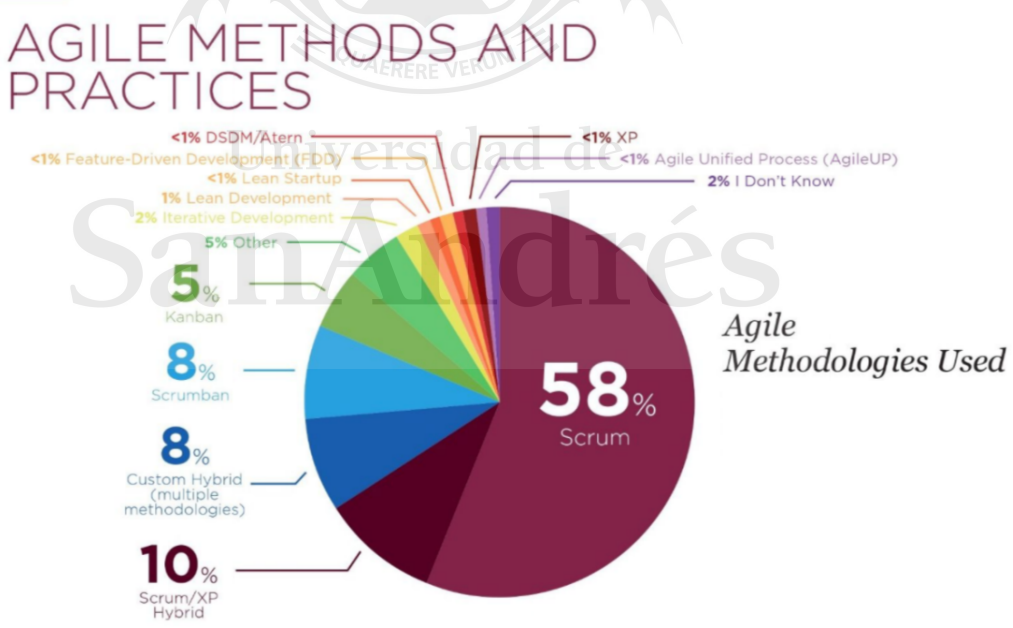

Figura 3: Porcentaje de adopción de la metodologías ágil y su variante a nivel mundial en empresas de tecnología. Fuente: VersionOne - 11° annual state of agile report 2016

Varios autores han abordado el tema de marcos de trabajo y prácticas basadas en la metodología ágil y en el libro de Kenneth S. Rubin, “Essential Scrum: A Practical Guide to the Most Popular Agile Process (Shawn Kahl's Library)”, se refiere al Scrum como un marco de trabajo para el desarrollo de productos y servicios innovadores cuyo objetivo principal es un enfoque ágil para la gestión de proyectos de software y aumentar la probabilidad de un desarrollo exitoso de software, mientras que las prácticas “XP Hybrid” se centra más en las actividades a nivel implementación del software. Ambos enfoques, sin embargo incorporan los principios centrales del software ágil de desarrollo los cuales se encuentran declarados en el manifiesto.

2.1.6 Marco de trabajo ágil Scrum

El primer artículo académico sobre el Scrum fue escrito en el “Harvard Business Review Article”, con el nombre de “The new new product development game” (Takeuchi & Nonaka, 1986). Este artículo académico detalla como Honda, Canon y Fuji-Xerox produjeron ganancias utilizando un enfoque escalable y basado en el trabajo de equipo para el desarrollo de productos de una sola vez. Este artículo marcó el origen del Scrum, ya que une y relaciona múltiples conceptos que dieron origen al Scrum.

El nombre de la metodología surge debido a que Takeuchi y Nonaka utilizaron dentro del artículo científico metáforas del rugby para detallar las operaciones dentro de las empresas. El scrum es la forma de reiniciar el juego después de una infracción accidental o cuando la bola ha salido del juego y es utilizado para describir el desarrollo del producto en una compañía.

En 1993, Jeff Sutherland y su equipo crearon la "Metodología de desarrollo Scrum" para su uso en el proceso de desarrollo del software. Jeff Sutherland y su equipo combinó conceptos del artículo "The new new product development game" con conceptos de desarrollo del software orientado a objetos, metodología de desarrollo tradicional, control de procesos empíricos y desarrollo iterativo e incremental, entre otros (Rubin, 2012).

2.1.7 Principales conceptos de Scrum

El scrum no es un proceso estandarizado donde se puede seguir metodológicamente una serie de pasos secuenciales con objetivos claramente definidos para producir a tiempo y según el presupuesto, el mejor producto que asegure calidad y satisfaga al cliente. Sino que el Scrum brinda un marco de trabajo iterativo e incremental para organizar y administrar las tareas, buscando eliminar posibles desviaciones en el producto final mediante la prueba de los usuarios permitiendo entregar productos de manera más eficiente, y con mayor fiabilidad, debido a que son probadas con el usuario y refinadas a través de iteraciones rápidas.

Este marco de trabajo para el desarrollo del software se encuentra centrado en las personas basado en la honestidad, respeto, confianza y la colaboración entre el equipo de trabajo (Rubin, 2012).

Las prácticas de Scrum están basadas en 4 pilares: Roles, Actividades, Artefactos, Reglas.

Roles

Los esfuerzos para el desarrollo de un software es realizado por uno o más equipos, cada uno compuesto por tres roles de Scrum: “Product Owner”, “Scrum Master”, y el equipo de desarrollo.

El “Product Owner” es el responsable de definir qué características y funciones se deben construir y en qué orden se deben realizar en relación al producto que se desea.

El “Scrum Master”, es el responsable de guiar al equipo de desarrollo y de realizar un seguimiento del proyecto con el fin de asegurar que cumpla con el marco de trabajo del Scrum.

El equipo de desarrollo es el responsable de determinar cómo entregar lo solicitado por el propietario del producto. Usualmente el equipo de desarrollo está formado por diferentes especialidades (programador, tester, arquitecto, entre otros).

En la Figura 4, se puede observar como es la interacción entre las actividades y los artefactos utilizados para el desarrollo del software utilizando el marco de trabajo de Scrum.

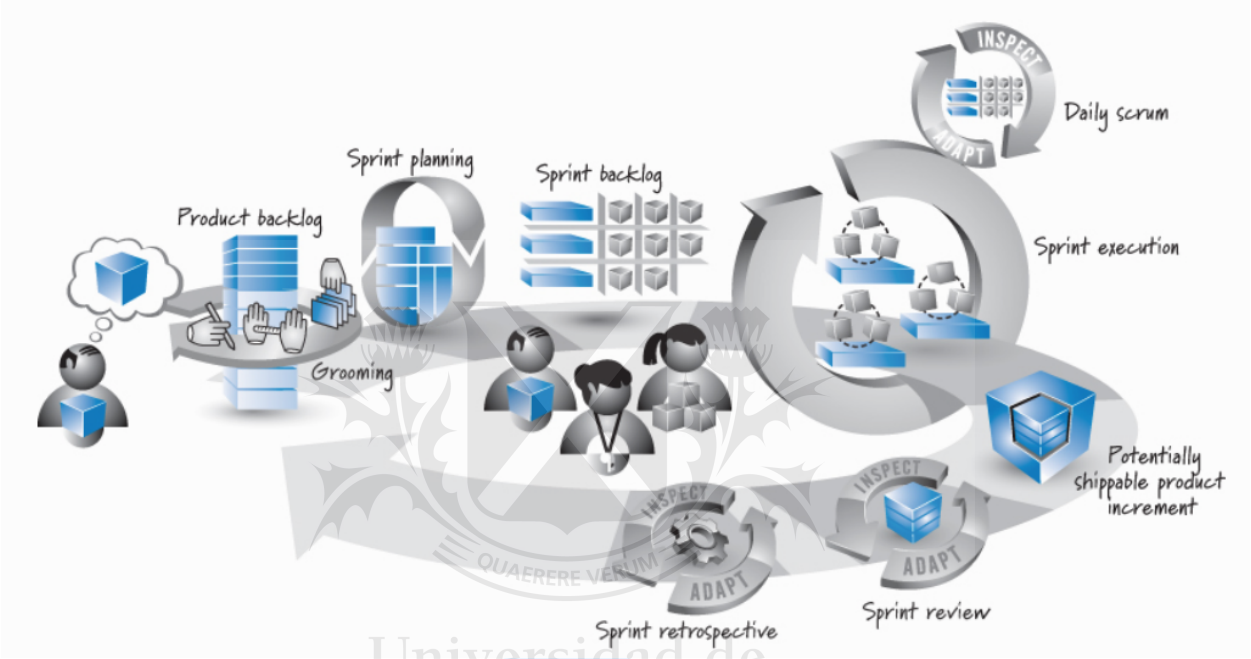

Figura 4: Actividades que ocurren durante la implementación del Scrum. Fuente: Scrum activities and Artifacts (Rubin, 2012).

Actividades

Refinar lista funcionalidades del producto ("Product backlog grooming"): El Product Owner, conoce el producto que desea crear. Para esto, divide el producto en características que se recopilan en una lista priorizada.

Sprint: Un “Sprint” incluye desde la planificación (“Sprint Planing”), abarca la ejecución (“Sprint execution”), continúa con la revisión (“Sprint Review”) y finaliza con la retrospección (“Sprint retrospective”). Estas iteraciones de igual duración son realizadas en ciclos de hasta un mes y cada vez que termina un ciclo, debe crear algo de valor tangible para el cliente o usuario.

Planificación del Sprint (“Sprint Planning”): El Backlog contiene tareas que usualmente son mayores a las que el equipo de desarrollo puede realizar en un Sprint, por lo que en la etapa de planificación el Owner del producto y el equipo de desarrollo determinan que conjunto de características se van a completar (comenzando por las características de mayor prioridad).

Asimismo, los miembros del equipo desarrollan en esta etapa una segunda lista de funcionalidades (“Sprint backlog”), la cual es utilizada para dividir la característica en tareas y asegurar que se está implementando dicha característica acorde a lo planificado.

Ejecución del Sprint (“Sprint execution”): En esta etapa, el equipo de desarrollo realiza las actividades necesarias para realizar las características comprometidas. Para ello el “Scrum Mater”, es el encargado de realizar un seguimiento sobre el equipo de desarrollo, para asegurar que todo el trabajo se haya realizado con la mayor calidad posible.

Scrum diario (“Daily Scrum)”: Cada día durante la ejecución del Sprint, los miembros del equipo colaboran entre sí para administrar el flujo de trabajo. Para ello,

realizan una reunión de 15 minutos donde se realizan 3 preguntas específicas que permiten inspeccionar el estado del trabajo y adaptar el mismo dependiendo las necesidades y prioridades para asegurar el cumplimiento de los objetivos establecidos.

Revisión del Sprint ("Sprint review"): Antes de finalizar con el Sprint, se reúnen los auspiciantes del proyecto (personal operativo de las áreas del negocio), clientes, usuarios y otros equipos interesados junto al equipo del Scrum para que obtengan una visibilidad clara de lo que está ocurriendo y tener la oportunidad de ayudar a guiar el próximo desarrollo para garantizar que se cree la solución más apropiada para el negocio.

Retrospección del Sprint ("Sprint retrospectiva"): Luego de la revisión con los auspiciantes del proyecto, el equipo de Scrum realiza una retrospección sobre cómo se está trabajando en el proyecto. El resultado de estas conclusiones puede modificar la lista de productos ("Product Backlog") para asegurar que se cumpla con el objetivo.

Una vez finalizada el Sprint, ésta se repite nuevamente desde el comienzo con el equipo de desarrollo determinando qué conjunto de características de la lista de productos se deben contemplar para realizar en la etapa. Una vez que se haya completado un número de Sprint, el producto queda finalizado y puede ser utilizada la solución.

Artefactos

Lista de productos (“Backlog”): El “backlog” está conformada por una cantidad de tareas ordenadas secuencialmente por importancia. Esta lista es manipulada por el “Product Owner”, quien se comunica con los auspiciantes del producto para conocer las funcionalidades esperadas para el producto. La lista de características o “Backlog” se encuentra en constante evolución ya que se pueden añadir, eliminar y modificar nuevas características dependiendo de los cambios en el negocio.

Cada característica especificada en el “Backlog”, tiene un peso estimado relativo dependiendo el costo de desarrollar dicha característica, el marco de trabajo no especifica medida para tal fin.

Lista de Sprint (“Sprint Back log”): Para asegurar que el equipo de desarrollo ha asumido un compromiso razonable con la selección de las características (“Product backlog grooming”), los miembros del equipo crean una segunda lista de funcionalidades (“Sprint backlog”) sobre lo estipulado en la planificación, donde se describen las características a través de un conjunto de tareas detalladas, como el equipo debe diseñar, construir, integrar y probar el subconjunto de características de la lista de funcionalidades del producto seleccionadas (“Product backlog”).

Producto potencialmente entregable (“Product shippable product increment”): Cada fin de Sprint, se realiza un incremento en el producto parcial, lo que permite que la característica sea incluida en el producto parcial y pueda ser entregada al cliente para su utilización.

2.2 Teoría Actor-Red

2.2.1 Introducción a la teoría actor-red

El desarrollo inicial de la teoría de actor-red dentro del ambiente de la ciencia de la sociología fue desarrollado por Michael Callón (Callon, 1986) y Bruno Latour (Fujimura & Latour, 1989). Sin embargo, fue relacionado y aplicado a la tecnología por Latour en su artículo "Aramis, or, The Love of Technology" (Lyman, Latour, & Porter, 1997), y en "Social theory and the study of computerized work site" (Latour, 1996).

La teoría del actor-red aplicada a la tecnología, se ocupa de investigar lo social y lo tecnológico en conjunto a través del análisis de las redes que coexisten entre elementos humanos y no humanos como por ejemplo el software, hardware, infraestructura, y las comunicaciones. El enfoque de esta técnica se centra en las interacciones que se producen entre los actores que colaboran para alcanzar algún objetivo, y al hacerlo crean un actor-red (Law, 2009).

2.2.2 Beneficios de utilizar la teoría del actor-red

Según Zana Ahmedshareefm, Robert Hughes y Miltos Petridis, el resultado de un análisis aplicando la teoría del actor-red puede ser una descripción, un modelo o una explicación del área investigada, con el objetivo de "aprender de los actores" a través del rastreo de las asociaciones siguiendo las actividades de la gestión de proyectos (Ahmedshareef, Hughes, & Petridis, 2014).

Por otro lado, esta técnica permite aplicarse en contextos particulares para rastrear y explicar los procedimientos mediante los cuales se crean y mantienen redes

(entre actores). Adicionalmente, permite observar la estabilidad de los intereses que estos actores comparten y examinar para los casos donde las redes no se establecen, la razón del error.

2.2.3 Principales conceptos de la teoría actor-red

A continuación se detalla los principales conceptos que son utilizados por la teoría del actor-red:

Tabla 2: Principales conceptos de ANT y su relación con elementos de tecnología

Concepto	Descripción
Actor	Es un elemento dentro de una red que tiene la capacidad de influenciar sobre otros elementos de la red. Pueden ser humanos o no humanos como artefactos tecnológicos.
Intermediario	Es un elemento dentro de una red de actores que facilita la interacción entre los actores. Son las relaciones / asociaciones que los actores forjan para permitir la interacción. Traduce los intereses entre los actores para que puedan relacionarse.
Actor-Red	Es una red heterogénea que está formada por actores que interactúan a través de intermediarios para cumplir un objetivo.
Equipo	Representa a más de un elemento del tipo Actor dentro de una red. Pueden ser humanos o no humanos como artefactos tecnológicos.

Caja negra	Representa un elemento o conjunto cuyo funcionamiento no tiene importancia en la investigación por ser demasiado complejo

Tabla 2: Elaboración propia – en base a "Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity" (Law, 1992).

2.2.4 **Definición de los elementos para la representación de conceptos**

Con fines prácticos se utilizarán las siguientes representaciones para caracterizar los elementos que intervienen en el desarrollo del software bajo la teoría del Actor-Red.

En la figura 5, se puede observar cómo son representados los distintos elementos que componen la teoría del Actor-Red

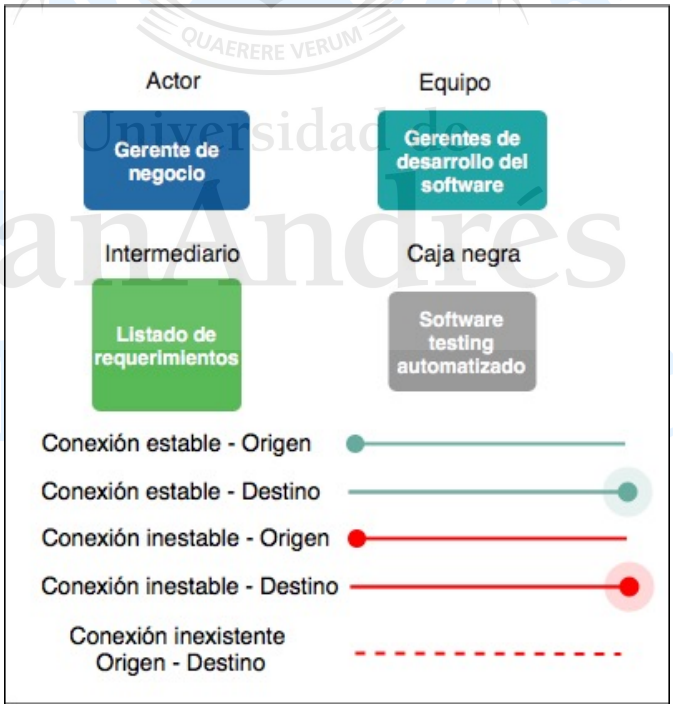

Figura 5: Representación de los elementos de la teoría del actor-red. Fuente: Elaboración propia en base a "Actor-Network theory in information systems research" (Alexander & Silvis, 2014).

2.2.5 Teoría de actor-red ejemplificada en el marco de trabajo Scrum

Se detalla a continuación un ejemplo sobre cómo se puede ejemplificar el uso de Scrum mediante la teoría del actor-red. Por cada actividad que sea generada por los distintos involucrados en el proceso, se identificará entre paréntesis a qué concepto de la teoría del actor-red hace mención.

El ejemplo demuestra cómo se relacionan los distintos elementos que grafican la red de actor-red utilizada para describir cómo es la confección del "backlog":

Durante la etapa de la confección del "backlog" (Actor-Red), el gerente de desarrollo de software (Actor) le informa al líder de desarrollo (Actor) el listado de iniciativas (Intermediario – Conexión estable) que deberá desarrollar en el sprint.

El líder de desarrollo (Actor) analiza las iniciativas del mes a realizar y asigna responsables (Intermediario – Conexión estable) de su equipo al "backlog" de tareas y responsables (Actor).

No se observa que el líder de desarrollo (Actor) tenga en cuenta el informe de retrospectiva y las habilidades de desarrolladores (Intermediario – Conexión inestable) que se encuentran almacenados en el repositorio de documentación (Actor).

Una vez confeccionado el backlog de tareas y responsables (Actor), el gerente de negocios (Actor) procede a asignar a analistas de negocios a cada tarea ("Intermediario –

Conexión inestable) descripta en el backlog para que puedan describir la funcionalidad esperada por el negocio en las reuniones de "Definición funcional y Requerimientos técnicos" ("Actor-Red").

En la figura 6, se representan los actores que intervienen al momento de la confección del "backlog" cómo interactúan entre ellos y la estabilidad de sus conexiones.

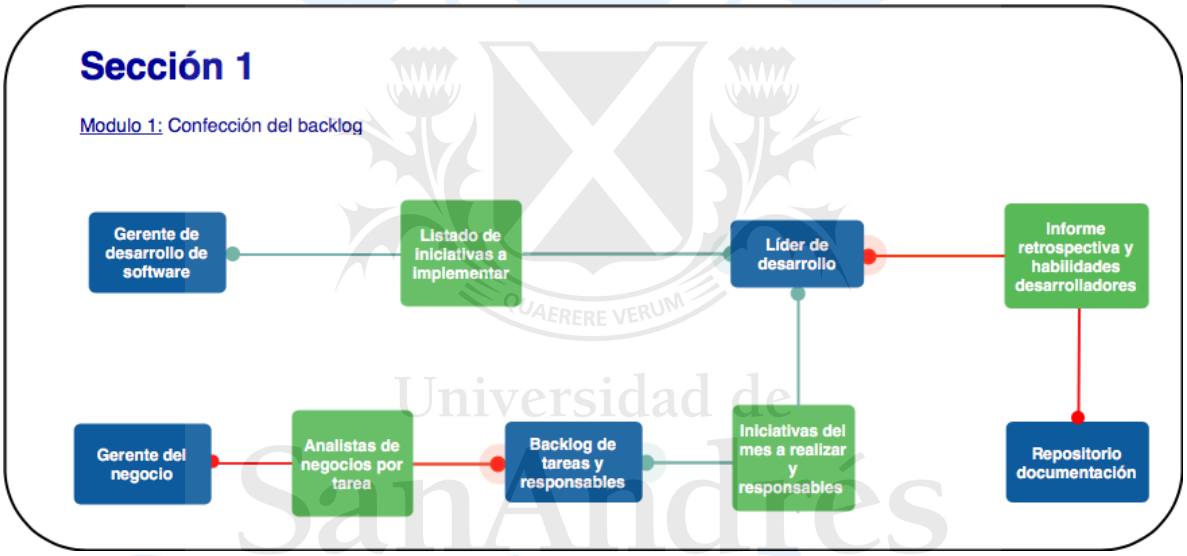

Figura 6: Diagrama ejemplificador de la teoría del actor-red y etapa de la confección del backlog según la metodología S.C.R.U.M. Fuente: Elaboración propia en base a conceptos de Scrum (Rubin, 2012).

3. Metodología de la investigación

3.1 Tipo de investigación

Este trabajo de investigación se realizó bajo un paradigma cualitativo y una investigación explicativa con fin de abordar todos los conceptos claves utilizados para el desarrollo de la tesis. Dicho abordaje fue realizado a partir de la investigación, análisis, contraste y evaluación de los resultados de que fueron obtenidos de distintas fuentes (bibliografía, entrevistas y registros no gráficos).

3.2 Procedimiento de la investigación

Para comprender el comportamiento de las empresas de tecnología a nivel comunicación, área operativa de sistemas, y su relación con la introducción de errores en el código del software asociados a procesos de innovación, se realizó un estudio de caso único sobre el proceso de desarrollo dentro del sector de "e-commerce" en una gran empresa de retail en la Argentina.

En este estudio del caso único se analizó cómo se implementa la metodología de desarrollo del software en un grupo de desarrolladores, y cómo son los elementos y las personas que están involucrados en las tareas y la comunicación.

Asimismo, fueron analizados todos los informes que se generados dentro de la compañía como análisis de causa y efecto sobre errores en el software que se reportaron el último año. Se seleccionó uno de esos informes con el fin de realizar entrevistas con los involucrados para conocer a detalle las razones por las cuales ocurrió el error en el software y, entender el contexto (metodología que utilizaba, confección del grupo de

desarrollo y comunicación con las áreas del negocio) del equipo de trabajó en ese momento.

Posterior a la realización de las entrevistas, revisión documental y del estudio de caso único, se utilizó la teoría del Scrum y el actor-red descriptas en el marco teórico del presente trabajo académico para definir los esquemas de actor-red basados en Scrum que fueron utilizados para el análisis y la comparación de los elementos que intervinieron en el desarrollo del software.

Con dichos esquemas, se realizó un contraste de información donde se revelaron las deficiencias en el proceso, en términos de incorrecta implementación del proceso de desarrollo del software o problemas de comunicación que ocasionaron la detección tardía de errores en el software.

3.3 Técnicas implementadas

Se realizó una revisión documental de los informes R.C.A. (Análisis de causa raíz) internos de Compañía de Tecnología S.R.L donde se seleccionó un evento el cual fue objeto de estudio a lo largo del trabajo académico.

Los informes R.C.A. son generados dentro de las organizaciones posterior a la remediación del error por el personal que lo detectó. Son utilizados para lograr un entendimiento histórico y preciso de las causas por las cuales ocurrió dicho error, cuál fue el impacto en los equipos y en caso de que existiese, el impacto monetario del mismo junto a documentación soporte que respalde lo detallado.

Luego de que se seleccionó un caso a analizar, se procedió a coordinar con los involucrados una serie de entrevistas para lograr un entendimiento completo de lo sucedido.

Las entrevistas fueron realizadas en un momento y lugar adecuados para los entrevistados con el fin de asegurar la comodidad de los mismos. Debido a que una de las secciones de la entrevista estuvo enfocada en detallar la comunicación entre el equipo de trabajo y otros sectores, las entrevistas fueron realizadas en salas de reuniones sin interacción con otros colegas. Cuando fue posible, se solicitó a los entrevistados a demostrar alguna de sus prácticas de trabajo durante la entrevista.

La entrevista fue realizada bajo la premisa del anonimato de los entrevistados, por lo que el nombre de las personas entrevistadas fue modificado.

Adicionalmente a la revisión documental y las entrevistas se realizó una observación pasiva junto al equipo de desarrollo "IT – Facturación & Cobranzas" con el fin de contrastar lo comentado por los entrevistados. La observación pasiva se realizó una vez por semana durante 2 meses (64 horas laborables), y constó de la revisión de la documentación generada, la utilización de herramientas de software para el desarrollo, la comunicación, el trabajo diario, las reuniones de equipo y las reuniones de otras áreas del equipo "IT - Facturación y Cobranzas".

3.4 Fuentes de datos utilizadas

Las fuentes de datos a utilizadas surgieron de los resultados de entrevistas realizadas a personal jerárquico de la organización, los cuales se fueron contrastadas con

los registros no gráficos generados en la observación pasiva realizada. Asimismo se analizó la documentación interna de la compañía con fin de lograr un mayor entendimiento de los sucesos y su correlación con lo relevado en las anteriores etapas.

4. Trabajo de campo y análisis de los resultados

4.1 Justificación de la compañía IT seleccionada

Compañía de Tecnología S.R.L es una compañía retail en Argentina que cotiza en la bolsa. La empresa cuenta con un sector de e-commerce el cual da soporte al marketplace (canal de distribución a través de internet).

La empresa Compañía de Tecnología S.R.L cuenta con operaciones en varios países de Latinoamerica. A través de su plataforma de comercio en línea y servicios relacionados, ofrece a sus clientes herramientas de comercio y pago en línea que contribuyen al desarrollo de la comunidad de comercio electrónico.

La empresa cuenta con varios millones de usuarios registrados en su plataforma y genera una ganancia neta mayor a U$S 100M anualmente.

Para ello, Compañía de Tecnología S.R.L cuenta con más de 5.000 empleados distribuidos en más de 2 países, de los cuales más del 10% de ellos realizan actividades relacionadas con el sector de la tecnología.

Su central operativa está ubicada en Argentina, donde se encuentra aproximadamente el 80% de sus empleados distribuidos en varios edificios y en varias provincias.

Compañía de Tecnología S.R.L es considerada una empresa innovadora en servicios de tecnología ya que genera nuevos servicios los cuales surgen de un proceso de innovación que se gesta en la compañía la cual es acompañada de una solución tecnológica que soporte a la misma.

Por lo descripto anteriormente, es posible afirmar que Compañía de Tecnología S.R.L cuenta con un gran sector de e-commerce, que innova en sus productos los cuales son respaldados con productos tecnológicos, por lo que la convierte en la mejor compañía para el caso de estudio de esta tesis.

4.2 Instrumentos utilizados

A continuación se detallan los instrumentos de recolección de datos que fueron utilizados para un enfoque cualitativo con el fin de obtener la información y los datos relacionados con el tema de estudio:

4.2.1 Instrumento I: Observación pasiva

Justificación del instrumento seleccionado

Se seleccionó dicho instrumento para la recolección de datos ya que permite dar una imagen instantánea de lo que realmente sucede dentro del equipo de trabajo. Dado que es pasiva, se disminuye el riesgo de que los sujetos modifiquen sus hábitos de conducta por la presencia del investigador. Este método es el indicado para corroborar y complementar la información recopilada mediante entrevistas.

Selección del grupo a observar

Dentro de Compañía de Tecnología S.R.L los grupos de desarrollo del software se dividen en tantos grupos como líneas de negocios o productos existan. Siendo que para cada línea de negocios o producto pueden existir sub-grupos que realicen módulos de software que conforman al software final que respalda el negocio o producto.

Esta fragmentación del software final en porciones de software realizadas por distintos sub-grupos les permite agilizar los procesos de desarrollo, minimizar la probabilidad de ingresar código erróneo y, de este modo entregar un software final en menor tiempo y más eficiente. De los posibles grupos de desarrolladores de software disponibles para realizar la observación pasiva, se seleccionó el grupo de "IT - Facturación & Cobranzas – Cargos".

Este sub-grupo pertenece al grupo de "IT-Facturación & Cobranzas" y es el encargado de desarrollar todo el software necesario para que la compañía pueda facturar y cobrar correctamente.

La razón por la cual se seleccionó dicho grupo es debido a que el sub-grupo de cargos fue creado hace 2 años, lo que lo convierte en un equipo que implementa de manera madura la metodología de desarrollo de Compañía de Tecnología S.R.L. Adicionalmente este equipo mantiene múltiples relaciones con otros equipos de desarrollo, ya que los cargos que generan, los cuales son solicitados por el departamento de facturación y cobranzas, son notificados por otro equipo de negocios (ventas) y consumidos por otros procesos de negocio (cobranzas).

Debido a la madurez del equipo de trabajo en relacion a la metodología de desarrollo que implementa acorde a los estándares de Compañía de Tecnología S.R.L y la comunicación que tiene el equipo con otras áreas de desarrollo y de negocios, hace que el equipo de cargos sea uno de los equipos dentro de la compañía de mayor interés para nuestro objeto de estudio.

4.2.2 Instrumento II: Revisión documental

Justificación del instrumento utilizado

Se seleccionó dicho instrumento ya que nos permitió entender rápidamente que informes R.C.A. fueron generados durante el año en Compañía de Tecnología S.R.L con el fin de poder realizar una clasificación sobre el motivo de los mismos y su impacto, para seleccionar el que resultó de más interés para nuestro objeto de estudio.

Los informes R.C.A (Análisis Causa Razón) constituyen una de las principales y más efectivas herramientas utilizados en el proceso de análisis y diagnóstico del origen de los eventos de falla en el software. Esta herramienta permite la evaluación de los hechos reales que generaron la pérdida de la función (indisponibilidad del software); y el impacto que provocaron estos eventos en términos monetarios o cualquier otra medida crítica para el negocio.

Selección del informe R.C.A. para investigación forense

El informe R.C.A. seleccionado fue generado por el departamento de "Desarrollo de Software – IT SAP" a mediados del mes de Julio 2017, donde se detalla la existencia de un error en una porción de código de software que contiene el comportamiento del esquema contable de un producto de la empresa.

Las razones por las cuales este caso resulta de interés es debido a que existió una pérdida monetaria para la compañía debido al error en el software, donde varios sub-grupos de desarrollo de software y negocios estuvieron involucrados en el proceso de

desarrollo del software, y, que el error ocurrió bajo un proceso de innovación de un producto de Compañía de Tecnología S.R.L realizado en el 2017.

Análisis de la documentación

Se realizó una revisión de toda la documentación realizada y utilizada durante el proceso de desarrollo del software para "Desarrollo del software IT-SAP", de ahora en adelante "IT-SAP", la cuales incluye archivos compartidos, conversaciones entre el personal involucrado y planillas de seguimientos propias y comunes utilizadas.

Adicionalmente, se analizó la documentación utilizada por el equipo de "IT - Facturación y Cobranzas - Cargos", de ahora en adelante "Cargos" durante la observación pasiva realizada.

4.2.3 Instrumento III: Entrevista equipo de IT-SAP

Justificación del instrumento utilizado

El método de recolección de datos de datos mediante entrevista es una técnica eficaz que permite obtener datos puntuales y relevantes ya que la información que se obtiene es superior a una respuesta escrita u observación. Por su condición oral y directa, se permite captar gestos y tonos de voz, entre otros que enriquecen la información verbal que el entrevistado comunica.

Se seleccionó este instrumento para poder debido a que permitió conocer con los involucrados las razones por las cuales ocurrieron los hechos más allá de lo detallado en el informe R.C.A. Asimismo, la entrevista permitió recorrer otros aspectos por fuera del

incidente que ocurrió que sirvió para entender el contexto del equipo de trabajo y su forma de trabajar.

Selección de personal a entrevistar

Se mantuvo una entrevista preliminar con uno de los participantes involucrados al momento de la detección del error, detallado en el informe R.C.A., con el fin de conocer los responsables que estuvieron involucrados en el proceso de desarrollo, y detección y corrección del error.

Dada la lista de participantes involucrados, se procedió a generar entrevistas con cada uno de ellos en salas de reuniones sin interacción con otros colegas con el fin de evitar el condicionamiento de las respuestas frente a supervisores o colegas.

Adicionalmente a la lista de participantes involucrados en el informe R.C.A., se procedió a realizar una entrevista con un experto en materia de Innovación basada en metodologías ágiles, seguridad y control interno.

En total se realizaron 7 entrevistas durante el mes de Diciembre, con una duración aproximada de 40 minutos cada una. Durante cada entrevista, se le consultó al entrevistado si estaba de acuerdo con que se grabe la misma, siendo que todos accedieron. Asimismo cabe destacar que las entrevistas fueron realizadas en salas de reuniones dentro de la compañía.

Tabla 3: Lista de participantes de la entrevista y su puesto organizacional.

Personal Compañía de Tecnología S.R.L			
Nombre	**Especialidad**	**Puesto**	**Relación**
Guido	Ingeniero en sistemas	Gerente Desarrollo de Software	Experto
Pablo	Lic. En administración de empresas	Gerente de Facturación & Cobranzas.	Informe R.C.A.
Diego	Estudiante de Ingeniería en informática	Líder desarrollador IT SAP	Informe R.C.A.
Victoria	Lic. Recursos Humanos	Líder de Recursos Humanos IT	Experto
Esteban	Ingeniero en Sistemas	Desarrollador backend IT SAP	Informe R.C.A.
Agustina	Licenciada en Administración	Analista Sr. Control Contable de Negocios	Informe R.C.A.
Gonzalo	Estudiante de Ingeniería en Sistemas	BackEnd - Desarrollador IT SAP	Informe R.C.A.

Tabla 3: Listado de participantes de entrevistas. Fuente: Elaboración propia.

4.3 Análisis de resultados de los instrumentos utilizados

A continuación se detallan los resultados obtenidos de la aplicación de los instrumentos previamente definidos:

Instrumento I: Observación pasiva al equipo de Cargos, la cual tuvo una duración de 64 horas distribuidas entre 2 meses.

Instrumento II: Revisión documental, en la que se revisaron 122 documentos generados por los equipos de Cargos y IT-SAP en conjunto a las áreas de negocios que participaron de los desarrollos, y 8 documentos generados por el equipo de Recursos Humanos de la compañía.

Instrumento III: Entrevista al equipo de IT-SAP y experto, las cuales tuvieron una duración de 6 horas distribuidos en 7 entrevistas.

4.3.1 Instrumento I: Observación pasiva del Equipo de Desarrollo de Software Facturación & Cobranzas – Cargos

Análisis de contexto equipos desarrollo en Compañía de Tecnología S.R.L

Los equipos de desarrollo de software en Compañía de Tecnología S.R.L se encuentran distribuidos en más de 3 edificios ubicados en dos países. En Argentina se encuentran tres ubicados en la provincia de Buenos Aires, uno en San Luis, uno en Córdoba y uno en otro país de Latinoamérica.

La fragmentación del sector de desarrollo en Compañía de Tecnología S.R.L está dada por la gran cantidad de desarrolladores que cuenta la compañía debido al gran crecimiento de la empresa en este último tiempo.

Para poder asegurar una correcta cohesión de los equipos, los mismos son clasificados según los procesos de la compañía en que se especializan y a su vez reubicados en los distintos edificios.

Cada equipo de software se encuentra internamente dividido en grupos y cada uno se encuentra dividido en subgrupos, que se especializan en desarrollar una característica del producto final.

La infraestructura de cada edificio está conformada por mesas de trabajo compartidas donde los desarrolladores se ubican sin un lugar en particular con sus notebooks individuales.

Si bien todos los equipos de desarrollo del software (conformados por grupos y subgrupos) trabajan para Compañía de Tecnología S.R.L, dada la observación pasiva realizada, cada subgrupo se comporta como una empresa de software independiente ya que cada uno:

- Mantiene sus beneficios laborales por sobre las normas de la compañía (tienen mayor flexibilidad horaria, disponen de posibilidad directa para la contratación de personal, se les permite trabajar dos o más días en la semana desde la casa, entre otros beneficios).
- En términos de reuniones algunos equipos de desarrollo las realizan de manera informal, diariamente, otros llevan a cabo reuniones semanales y algunos no implementan reuniones periódicas, manteniendo el seguimiento de sus proyectos mediante el uso de documentos compartidos.
- Cada subgrupo implementa una variante de metodología de desarrollo ágil que ellos consideren la más acertada. Algunos equipos aseguran implementar la metodología del Scrum y otros emplean una metodología híbrida utilizando el Scrum como marco principal y realizandole modificaciones al mismo acorde a su experiencia.
- Cuentan con conocimiento sobre una parte de la arquitectura del complejo sistema de Compañía de Tecnología S.R.L, por lo que cada subgrupo solo entiende una porción del sistema completo y, en ninguno de ellos, los desarrolladores entienden la arquitectura de la empresa.

- Gestionan su propia manera de documentar la planificación y las tareas que realizan. La manera en que la planificación y el seguimiento de las tareas es realizado depende de lo que defina el líder del subgrupo.
- Utilizan distintas herramientas para programar y realizar seguimiento a sus proyectos.
- Cuentan con su propio esquema de remuneración, por lo que los sueldos pueden diferir ampliamente entre individuos de grupos de una misma categoría.

Existe una gran incorporación de desarrolladores a los equipos de software debido a la constante demanda de personal de tecnología que requieren como consecuencia de las tasas de crecimiento de la compañía.

Se solicitó al departamento de Recursos Humanos documentación sobre los nuevos ingresos por parte del personal de desarrollo durante el transcurso de la observación pasiva. La documentación expone que ingresaron 27 personas al equipo de desarrollo en 2 meses, lo que significa que existe una constante incorporación de desarrolladores.

La gran mayoría de los desarrolladores que ingresan, comienzan a programar porciones de software sin entender la arquitectura en la que su subgrupo se especializa.

La rotación del personal de tecnología entre equipos de desarrollo, según lo que se observó, es poca o nula. La razón aparente surge de dos principales motivos:

- Cuando un desarrollador no está conforme con las tareas que desempeña dentro del equipo, optan por irse de la empresa antes de pedir el cambio hacia otro equipo. Esto es debido a la gran cantidad de oferta laboral que reciben diariamente.
- Debido a que cada subgrupo tiene sus propios beneficios, la posible rotación de equipo puede ocasionar la pérdida de beneficios y comodidad que actualmente existe.

La falta de rotación de personal de desarrollo, impide que los desarrolladores aprendices acumulen experiencia en otros equipos para lograr un entendimiento completo de la arquitectura de la empresa.

Observación pasiva equipo de cargos

El equipo de desarrollo Cargos, está conformado por un líder (Leandro), y de él dependen 8 ingenieros de desarrollo.

Dentro del equipo de cargos, en términos de rotaciones, no se han realizado entre un equipo y el de cargos, siendo que ninguno de los miembros actuales del subgrupo ha rotado y que desde el momento en que se comenzó la observación hasta la finalización (2 meses de duración aproximada), ingresaron dos nuevos miembros; uno se fue de la compañía.

Durante la observación pasiva se analizó de qué manera los recursos distribuían su trabajo diariamente. En base a lo observado se procedió a realizar la siguiente tabla,

resumen con las tareas habituales del sector y el tiempo promedio diario que demanda cada tarea para el líder de IT y su equipo de desarrollo:

En la tabla 4, se puede observar como es la distribución del trabajo diario para el equipo de desarrollo del software - cargos en base a la observación pasiva realizada.

Tarea / Participante	Según observación pasiva – Líder IT	Según observación pasiva – Equipo desarrollo
Confección del backlog y definición funcional	30%	20%
Soporte al equipo	20%	-
Desarrollo de software	-	25%
Soporte y mantenimiento	10%	25%
Pruebas unitarias y revisión de código	5%	10%
Pruebas integradoras	-	15%
Puesta en producción y documentación	Baja	Baja
Retrospectiva	-	-
Otras tareas	35%	-

Tabla 4: Distribución del trabajo operativo para el equipo de desarrollo del software – cargos. Fuente: Elaboración propia en base a la observación pasiva.

Se observa en el cuadro de ponderación de tareas que el líder del equipo de cargos y el equipo de desarrolladores no cuentan con un proceso formal para la documentación sobre los desarrollos, por lo que es considerado bajo. Esto es debido a que la única documentación que realiza el líder es un block de notas privado en donde mantiene el seguimiento de los desarrollos, mientras que el equipo de desarrollo almacena archivos privados para registrar las minutas de las reuniones de relevamiento.

Asimismo observamos que el líder de tecnología utiliza un 35% de su tiempo en otras tareas que no hacen al desarrollo de los proyectos. Esto se observó en repetidas

ocasiones en los que el líder no asistía a las oficinas o por fuera de su horario laboral de manera excesiva.

Por último, se obesvó la existencia de una carga significativa del 25% por parte del equipo de desarrollo para las tareas de soporte y mantenimiento.

Durante las reuniones de seguimiento que el equipo realizaba semanalmente, aproximadamente un 30% de las mismas estaban focalizada en notificar, analizar y resolver los problemas que ocurrían diariamente por inconsistencias en los cargos generados por distintos procesos.

Estas inconsistencias se daban debido a que existía código de software incorrectamente desarrollado que no articulaba correctamente con otros procesos, lo que generaba que en algunos casos no se recibía el input esperado, mientras que en otros se generaba un output distinto o directamente no se generaba.

Un ejemplo que ocurrió relacionado a estas inconsistencias fue que varios usuarios del site vendieron artículos desde el sitio y la empresa no pudo cobrarle la comisión por venta debido a que no se había generado el cargo por una incorrecta articulación con el proceso origen. Dado que no se generó el cargo, el proceso de facturación no reconoció esa venta y por lo tanto la compañía no podía reclamar el dinero que correspondía.

Dado que estas inconsistencias ocurren con frecuencia en bajas cantidades (usualmente eran entre 100 y 1000 casos por semana), el sector de Cargos desarrolló un software para detectar las mismas y notificar mediante alarmas (las cuales eran definidas

por ellos previamente acorde a una criticidad) con el fin de que los desarrolladores genere las modificaciones necesarias para arreglarlas.

Durante los 2 meses en los que se realizó la observación pasiva, se obtuvo acceso a la lista de mails del equipo de cargos donde el software de monitoreo enviaba las alarmas de manera automática. En dicho período se recibieron aproximadamente 900 mails relacionados a las mismas por inconsistencias en datos productivos.

Como conclusión sobre la ponderación de las tareas diarias se observó que no existe un proceso formal de documentación para los relevamientos analizados, así como tampoco se documenta la planificación y las tareas a realizar. Asimismo, se vislumbro una gran cantidad de inconsistencias entre procesos que ocurren en el ambiente productivo, lo que implica que existen problemas en las integraciones de software entre equipos de desarrollo.

Con el fin de estructurar las observaciones detectadas durante observación realizada, se utilizara las fases de la metodología de desarrollo Scrum analizadas en el marco teórico del presente trabajo con el objetivo de facilitar el análisis y comprensión de los hechos.

En la figura 7, se observan las secciones que componen el marco de trabajo de la metodología de Scrum y su descomposición en módulos.

Figura 7: Etapas de la metodología S.C.R.U.M. y su división en tareas. Fuente: Elaboración propia en base a (Rubín, 2012).

Definición trimestral de iniciativas y asignación

No fue posible asistir a una reunión de definición trimestral de iniciativas debido a que al momento en que se comenzó la observación pasiva ya se habían establecido las iniciativas a realizar en el último trimestre. Sin embargo, durante las reuniones que se realizaron en el equipo de cargos se hizo mención al tema ya que un nuevo desarrollador había ingresado en ese período y no conocía cómo era el funcionamiento.

Las reuniones de iniciativas son realizadas trimestralmente en la que participan directores y alta gerencia de la empresa. En esta reunión se definen las tareas que van a realizar durante los próximos 3 meses y quienes son los gerentes encargados de llevar adelante dichas tareas.

Estas iniciativas incluyen nuevos productos como parte de una innovación, modificaciones a productos existentes para mejorar el rendimiento de los mismos o sea necesario la modificación de un producto cuestiones de contexto como temas impositivos, legales o de la competencia.

Esta reunión no influye al equipo de cargos ya que no participan de la misma y, dado que no se asistió no se realiza ninguna conclusión sobre la misma.

Las conexiones entre los equipos de trabajo son estables ya que logran comunicarse efectivamente. Asimismo los intermediarios que se configuran en el proceso, facilitan la interacción entre ellos por lo que la red no presenta conflictos.

Figura 8, se observan los elementos que intervienen para la definición trimestral de iniciativas y asignación de las mismas según observación pasiva.

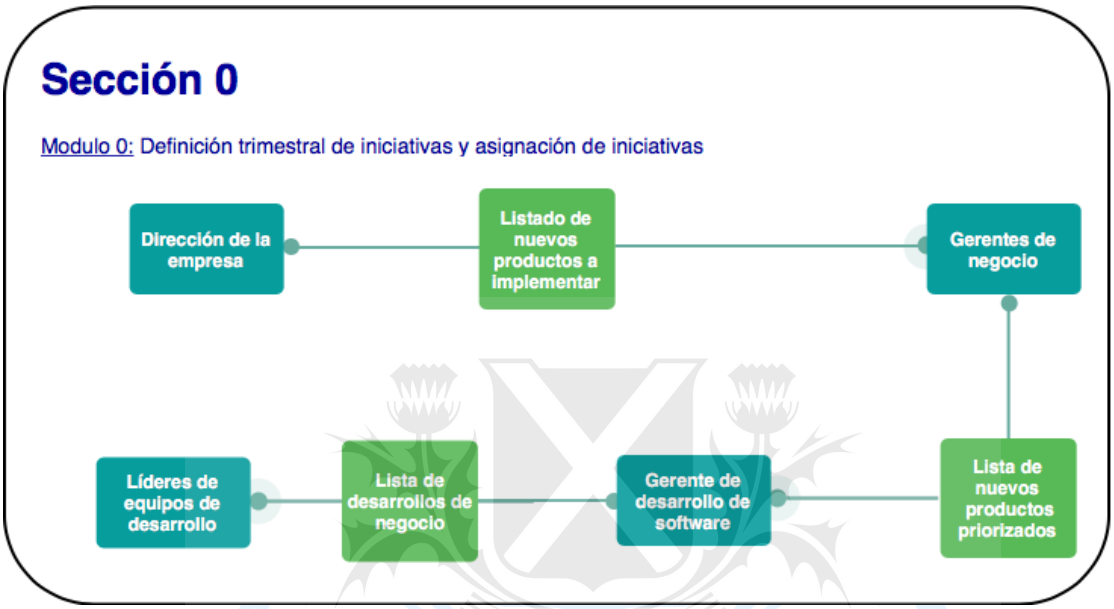

Figura 8: Diagrama de actor-red para la definición trimestral de iniciativas y asignación de iniciativas según observación pasiva. Fuente: Elaboración propia en base a la observación pasiva.

Confección del backlog

Las reuniones para la confección del backlog son realizadas mensualmente entre el equipo de cargos y facturación y cobranzas. En estas reuniones se toman las iniciativas que fueron definidas en los encuentros trimestrales, se priorizan y dividen en tareas dependiendo la carga operativa con la que cuentan ambos equipos. Cada tarea es asignada a un líder de desarrollo y un representante del negocio para que pueda describir los requerimientos técnicos que son necesarios para el desarrollo de la funcionalidad.

Se asistió a una reunión de confección del backlog en donde el gerente de facturación y cobranzas se acercó hacia las oficinas donde se encuentra el equipo de desarrollo. Durante la reunión se definieron las tareas y se observó que el gerente de facturación y cobranzas buscaba la manera de poder asignar los recursos que él creía más idóneos a las tareas.

Se vislumbró que en más de una ocasión no fue asignado el recurso esperado debido a los tiempos en participación que el desarrollo necesitaba frente a los tiempos disponibles con los que contaba, por lo que era asignado otro recurso en su reemplazo con menor experiencia. En contraposición el líder de desarrollo no presentó dicho inconveniente, por lo que todos los desarrolladores fueron asignados a los proyectos acorde a su experiencia y conocimiento sobre la tarea a realizar.

Como conclusión de lo observado podemos afirmar que el equipo de negocios, en este caso facturación y cobranzas, no cuenta con los recursos suficientes para asignar los representantes a las tareas en base a su experiencia y conocimiento debido a que las asignación de dichos recursos depende de la carga operativa que con la que éste cuenta, por lo que si no cuenta con tiempo disponible no es asignado a la tarea.

Dentro del proceso de confección del backlog se observó una conexión inestable del tipo origen-destino para el Líder de desarrollo y el Repositorio de documentación. Esta conexión inestable se reflejó en el momento en que el Líder de desarrollo no consultó al repositorio donde se almacenan los informes de retrospectivas y las habilidades de desarrollo que tienen que ser considerados al momento de crear los

equipos de trabajo y asignar tareas del backlog. Esta inexistencia genera una conexión inestable y por lo tanto no es posible que el intermediario genere una correcta interacción entre los actores. Cabe destacar que los elementos restantes involucrados en el proceso, cuentan con conexiones estables por lo que todos los intereses entre los actores son relacionados correctamente.

En la figura 9, se observan los elementos que intervienen en la confección del backlog según la observación pasiva.

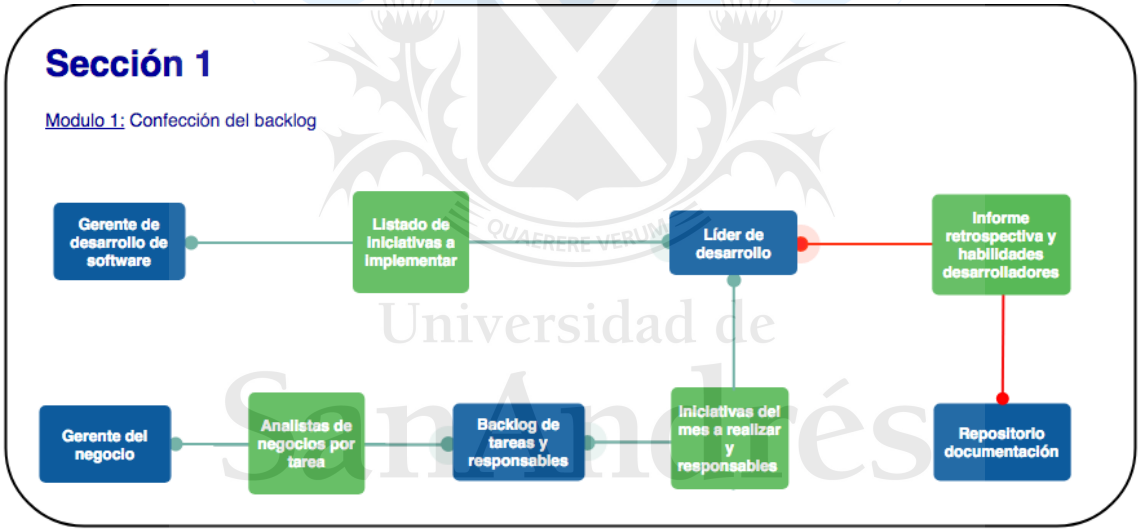

Figura 9: Diagrama de actor-red para la confección del backlog según observación pasiva.

Fuente: Elaboración propia en base a la observación pasiva.

Definición funcional y requerimientos técnicos

Todas las reuniones fueron realizadas semanalmente los días Lunes junto a un representante del equipo de Facturación y Cobranzas. Las reuniones eran lideradas por el

desarrollador asignado a la tarea quien consultó sobre los requerimientos de la nueva funcionalidad a desarrollar. En escasas ocasiones el líder de IT asistió a las reuniones.

Para la gran mayoría de los encuentros, el representante del equipo de Facturación y Cobranzas se encontraba con poco tiempo para la realización de la misma por lo que intentaba simplificar la definición de los requerimientos abstrayéndose de los detalles con el fin de lograr el mínimo entendimiento necesario para desarrollar.

En estos casos en que se simplificaron las funcionalidades el desarrollador se limitaba a realizar una rápida comprensión y definición de los requerimientos. Para poder lograr esto, los desarrolladores se abstrajeron del objetivo del negocio y su relación con otros procesos para enfocarse en las funcionalidades que sistema debía realizar. Dichas actividades, estaban subdivididas en tareas que eran descriptas con vocabulario técnico y eran documentadas en archivos privados.

Algunas reuniones de definición de requerimientos del software se realizaron en reiteradas ocasiones debido a que se habían definido requerimientos que no contaban con la información suficiente para poder llevar a cabo el desarrollo.

En ninguna de las reuniones se identificaron indicios de que los desarrolladores no comprendan el lenguaje contable a través del cual el representante del negocio utilizó para explicar la funcionalidad.

Como conclusión sobre las reuniones de definición funcional y requerimientos técnicos se observó que los desarrolladores contaban con el conocimiento contable

suficiente para poder comprender los requerimientos que definen los representantes del equipo de Facturación y Cobranzas.

Asimismo, se observa que el equipo de facturación y cobranzas presentó continuamente inconvenientes con los tiempos que disponía para asistir a las reuniones, esto se refleja en una conexión inestable del tipo origen-destino que da a entender que existen problemas de comunicación que afectan directamente a la calidad de los requerimientos. Esta conexión inestable afectó la manera en que se definieron los requerimientos técnicos y de negocio, debido a que los mismos no contaban con la información necesaria para poder desarrollar la funcionalidad. Por último, se observó que no existe un proceso formal para la documentación de los requerimientos ya que cada desarrollador documenta lo relevado en diferentes maneras.

En la figura 10, se observan los elementos que intervienen en la definición funcional y requerimientos técnicos según observación pasiva.

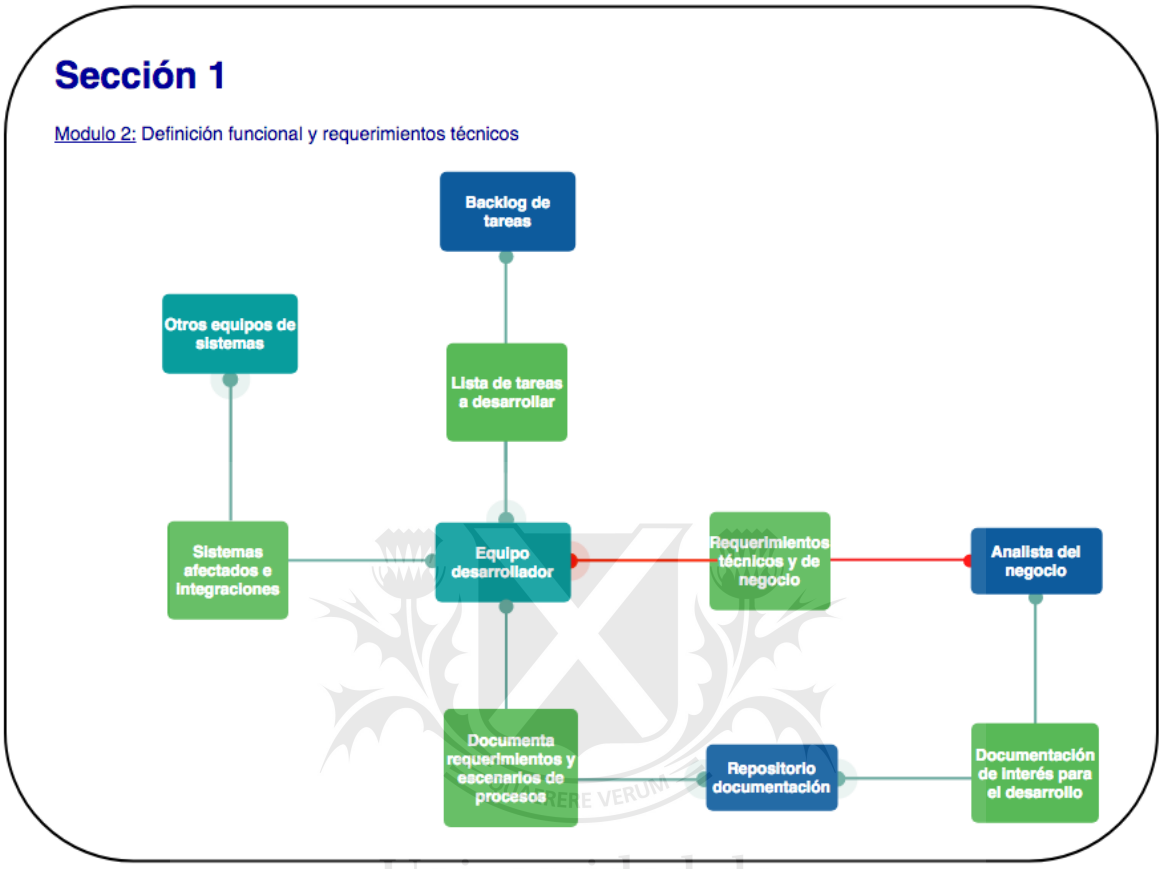

Figura 10: Diagrama de actor-red para la definición funcional y requerimientos técnicos según observación pasiva. Fuente: Elaboración propia en base a la observación pasiva.

Desarrollo de software

Durante observación pasiva se analizó la manera en que los ingenieros de software desarrollaban el mismo. Se observó que los desarrolladores contaban con buenos conocimientos técnicos sobre la tecnología para la cual estaban desarrollando, ya que no se realizaban preguntas técnicas sobre la programación o la manera en que debía confeccionar una funcionalidad.

Sin embargo, se observó en reiteradas oportunidades que aquellos ingenieros que contaban con menos experiencia tenían dudas sobre la respuesta que debía dar el software de cara a la integración con otros procesos. Esto genera una conexión inestable al momento de comprender las arquitecturas e integraciones.

Para todos los casos, los ingenieros con mayor experiencia pudieron responderles asertivamente sobre la consulta, despejando toda duda y dando a entender que contaban con el conocimiento suficiente para el desarrollo de funcionalidades en relación a la arquitectura de la empresa y la integración con otros procesos.

Adicionalmente se observó que todos los desarrolladores dentro del equipo de Cargos utilizaban la misma herramienta para desarrollar el software siendo que otros equipos que compartían el espacio, utilizaban otra tecnología.

Como conclusión de lo comentado en relación al desarrollo del software, se observó que los ingenieros de software cuentan con un gran conocimiento técnico sobre la tecnología en la que están programando.

Asimismo, se destaca que aquellos profesionales que cuentan con menos experiencia presentan dificultades al momento de desarrollar el software exhiben inconvenientes con el intermediario código de la funcionalidad generado por una conexión inestable debido a un desconocimiento de la arquitectura, mientras que los que se encuentran con más tiempo en la empresa despejan dudas de los primeros, lo que implica que los ingenieros con mayor tiempo en la empresa cuentan con un mejor conocimiento de la arquitectura que se encuentra dada por la experiencia.

En la figura 11, se observan los elementos que intervienen en el desarrollo del software según observación pasiva.

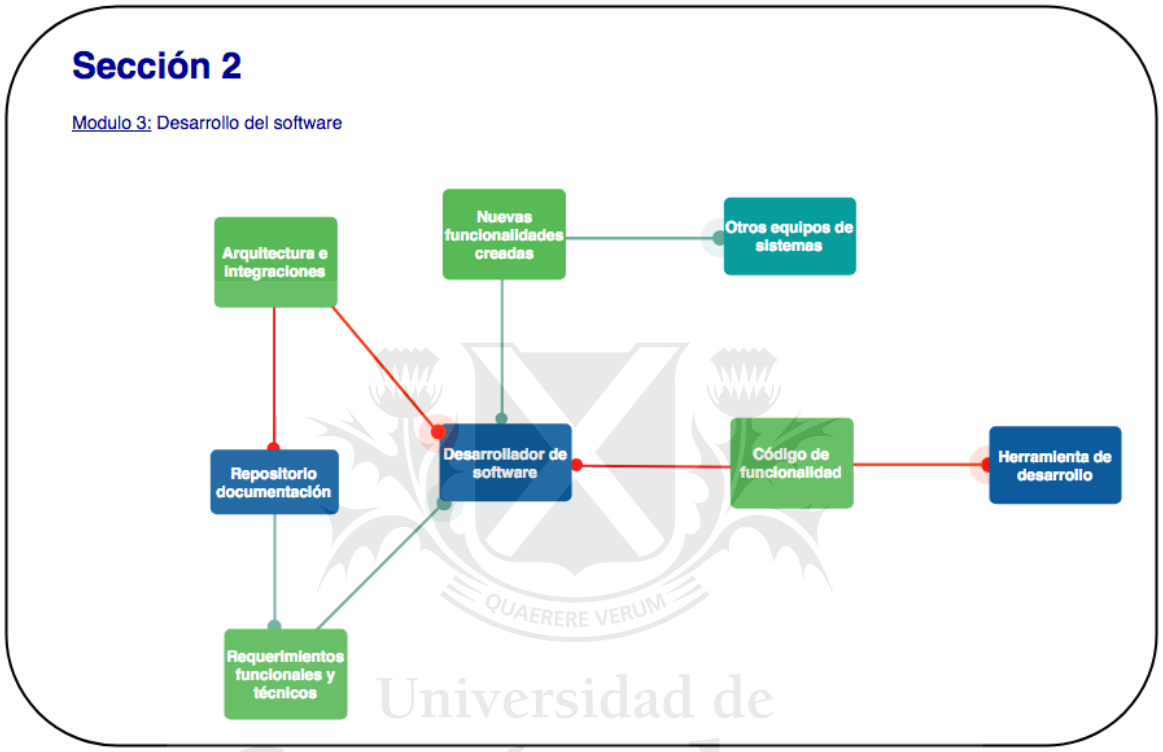

Figura 11: Diagrama de actor-red para el desarrollo del software según observación pasiva.

Fuente: Elaboración propia en base a la observación pasiva.

Pruebas unitarias y revisión de código

En la etapa de las pruebas unitarias se observó que los desarrolladores, una vez finalizada una funcionalidad (tarea del backlog), se comunicaban con el equipo de negocios con el fin de definir las pruebas a realizar.

Durante una reunión informal el desarrollador confeccionó una lista de los resultados esperados en relación a los datos ingresados, por cada escenario definido por

el equipo de negocios. Adicionalmente, el desarrollador realizaba sus propias pruebas para analizar la calidad del software desarrollado y los tiempos de respuesta de la misma.

La etapa de las pruebas unitarias para comprobar el correcto desarrollo de una funcionalidad del sistema constaba de dos partes:

- Recorrer el código reiteradas veces con el fin de observar alguna inconsistencia en la declaración de las variables o funciones desarrolladas.
- Luego del análisis del código, el desarrollador procedía a verificar manualmente dado un ingreso previo de datos, que la funcionalidad devuelva un resultado esperado. Para aquellos casos en que la funcionalidad no devolvía los datos en formato o tipo correcto, el desarrollador modificaba el código y repetía nuevamente las etapas.

Luego de haber realizado las pruebas unitarias, el desarrollador notificaba al líder del equipo con el fin de que sea revisado el código y la funcionalidad por un tercero.

Las pruebas fueron realizadas usualmente por el líder, pero para algunos casos en los que el mismo no contaba con el tiempo suficiente para la realización, un desarrollador del equipo fue asignado para que lleve a cabo la revisión del código.

Una vez que el código era revisado y aprobado, el revisor realizaba un “commit” en la herramienta Github con el fin de generar una nueva versión del mismo y preparar la funcionalidad para ser probada en la etapa de las pruebas integrales.

Como resumen de las pruebas unitarias, se observa que el equipo de Desarrollo de Software – Cargos, no cuenta con una herramienta para realizar un testing automático de los cambios del negocio por lo que todas las pruebas son realizadas manualmente. Esto demuestra una conexión inexistente entre los desarrolladores y una herramienta de software automatizada para historias de usuario del negocio.

Asimismo, se observó que las pruebas relacionadas a la funcionalidad son realizadas por el equipo de negocio, lo que implicó la posibilidad de una omisión al momento de asegurar una correcta integración con otras funcionalidades por desconocimiento.

Por último se observó que la fase de revisión del código no pareció dar un valor agregado, lo que afecta a la conexión y puede causar inconvenientes en el intermediario ya que únicamente se revisa el código fuente y no existe un documento formal donde se establezcan los escenarios probados con el fin de asegurar que se están asegurando todos los aspectos del software.

En la figura 12, se observan los elementos que intervienen pruebas unitarias y revisión del código.

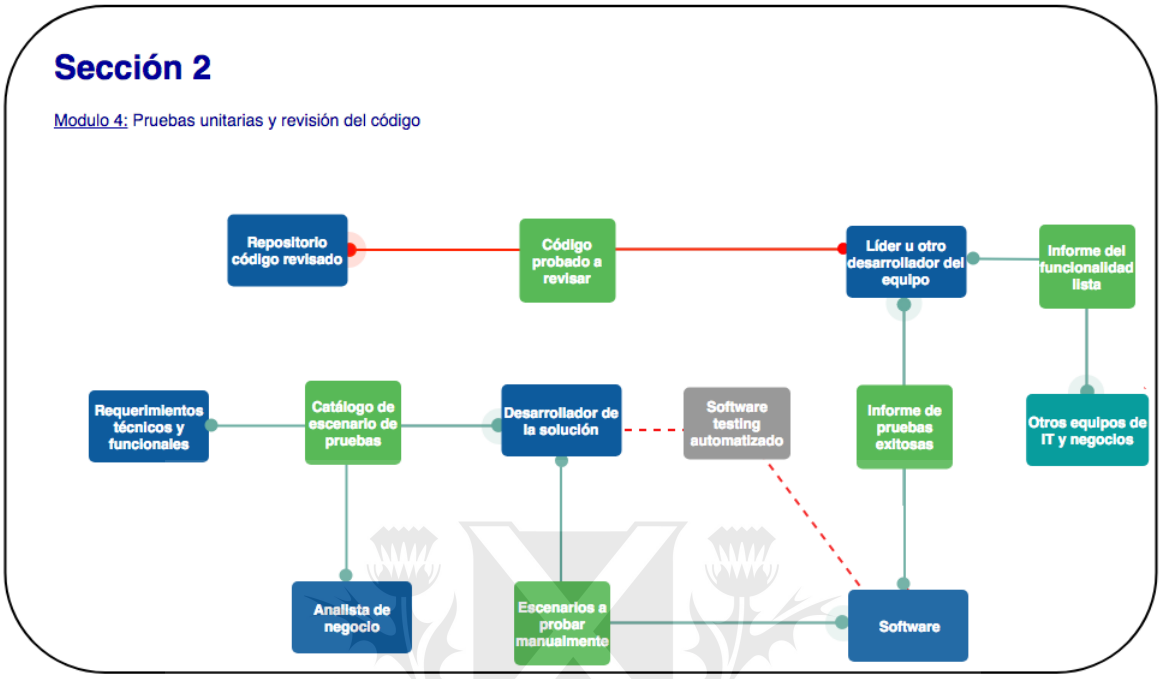

Figura 12: Diagrama de actor-red para las pruebas unitarias y revisión del código según observación pasiva. Fuente: Elaboración propia en base a la observación pasiva.

Pruebas integradoras

Las pruebas integradoras fueron realizadas entre los equipos de desarrollo que tenían alguna implicancia relacionada a la función desarrollada. Para ello, se identificaban qué equipos se iban a comunicar con dicha funcionalidad, y qué equipos esperaban recibir datos de la funcionalidad.

Una vez determinado los equipos que deben estar involucrados en las pruebas integradoras, se definieron los escenarios de pruebas con el equipo de negocios. Estos escenarios, en su mayoría eran similares a los de las pruebas unitarias. Cabe destacar,

que se realizaba uno o dos casos de prueba por escenario posible por lo que las pruebas eran realizadas por calidad y no cantidad.

Los equipos que se comunicaban con la funcionalidad, generaron en un entorno de pruebas, los datos ficticios acorde a los escenarios estipulados para que puedan comenzar con la prueba.

El equipo responsable de probar la funcionalidad recibía un mail, por parte de del grupo que se debía comunicar con la misma, confirmando que los datos se encontraban listos para probar en un ambiente de alfa, una vez que se ejecutaba la funcionalidad, el equipo de desarrollo de cargos notificaba vía mail al equipo que utilizaba la salida de la funcionalidad, que ya se encontraban los datos listos para ser consumidos.

Una vez que el último equipo recibía los datos y aseguraba que la funcionalidad funcionaba correctamente, enviaba un mail informando las pruebas exitosas y que el software se encontraba listo para ser enviado a producción.

Posterior a recibir dicho mail, el líder del proyecto procedía a realizar un “push” en la herramienta Github para subir a producción la nueva porción de software en el momento en que sea indicado.

Como resumen de las pruebas integradoras que realizó el equipo de cargos se observó una conexión inestable al momento de definir los escenarios de pruebas ya que la definición de los mismos depende directamente de la experiencia del equipo de negocios y usualmente no eran contemplados en su totalidad. Adicionalmente siendo

que se realizan pruebas de calidad y no cantidad, no fue posible probar escenarios distintos a los definidos previamente.

Asimismo se observó que si bien las pruebas integradoras son realizadas por varios equipos de desarrollo, durante el tiempo de la observación pasiva, el equipo de cargos no incluía a los otros equipos de desarrollo para la definición de las pruebas sino que eran incluidos para solicitar y verificar datos que se comunicaban con la funcionalidad. Lo que genera inconvenientes en la estabilidad de la conexión ya que otros equipos podrían haber contribuido con la definición de nuevos escenarios de pruebas en base a la experiencia de la arquitectura con la que cuentan.

Por último se observó que no existe documentación formal sobre los escenarios de pruebas realizados.

En la figura 13, se observan los elementos que intervienen en las pruebas integradoras según observación pasiva.

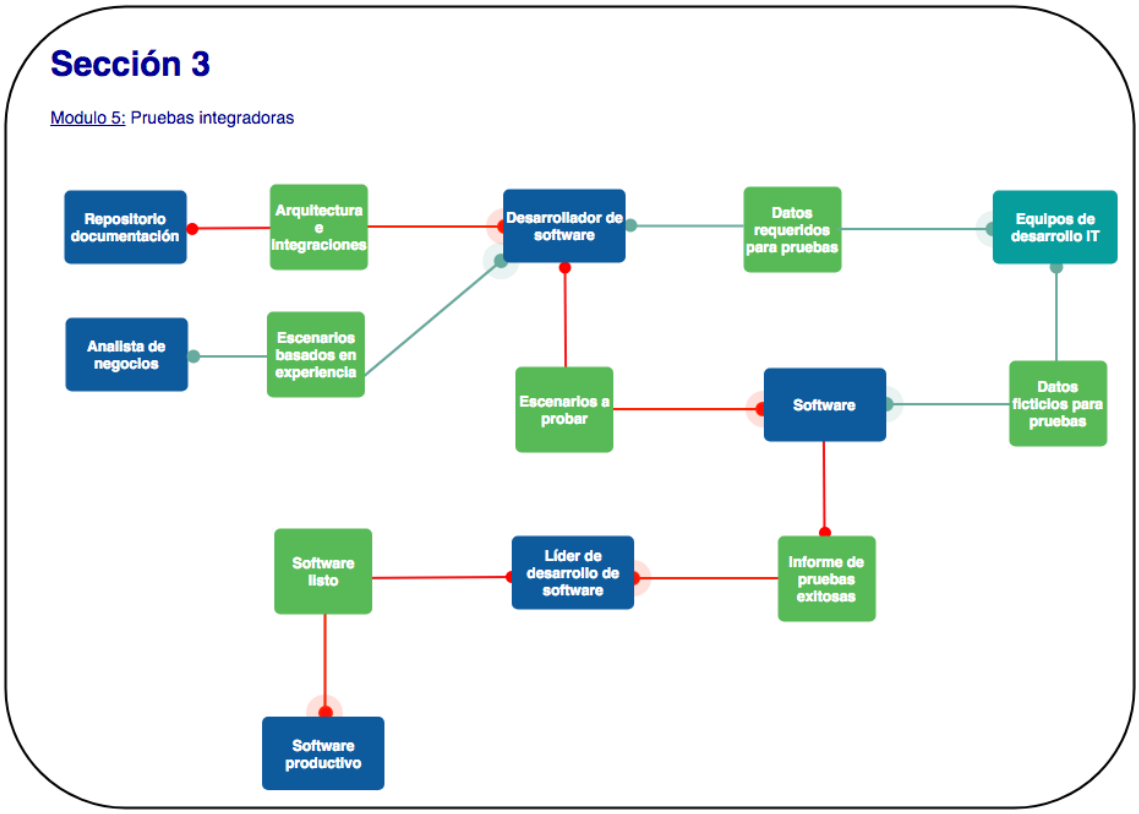

Figura 13: Diagrama de actor-red para las pruebas integradoras según observación pasiva.

Fuente: Elaboración propia en base a la observación pasiva.

Puesta en producción y documentación

Para las puestas en producción se toman un día entero y las realizan junto a uno o más miembros del equipo de negocio. Las puestas en producción usualmente suelen tardar 4 horas hasta que replican en todos los servidores.

En relación a la documentación, no se observó que se haya generado documentación alguna.

Retrospectiva

Durante el tiempo que se realizó la observación pasiva no se llevó a cabo una reunión de retrospectiva entre los miembros del equipo de cargos. Sin embargo se realizó una salida informal en un club con todos los equipos a fin de analizar cómo trabajaron los equipos durante el período. Si bien no se participó, se indagó sobre la reunión y varias fuentes comentaron que no existió un momento para realizar un análisis del trabajo realizado sino que fue una reunión para fomentar el trabajo en equipo y comunicar los próximos desarrollos a realizar por el área.

Según lo comentado por las fuentes que concurrieron a la reunión se concluye que no se realizaron reuniones de retrospectiva para analizar la razón de los desvíos durante el período ni tampoco analizar la razón por la cual ocurrieron fallas en producción con el fin de evitar las mismas.

4.3.2 Instrumento II: Equipo de Desarrollo de Software IT SAP - Entrevista equipo de créditos

Entrevista en base al contexto del equipo de IT-SAP

El equipo de IT-SAP está conformado por un líder (Diego), de él dependen 7 ingenieros de desarrollo y un sublíder técnico. Se adjunta a continuación un organigrama sobre el equipo de IT-SAP el cual fue confeccionado con información brindada por el equipo de Recursos Humanos de la empresa:

En la Figura 14, se observa el personal que compone el equipo de IT-SAP junto a su edad, antigüedad y estudios alcanzados.

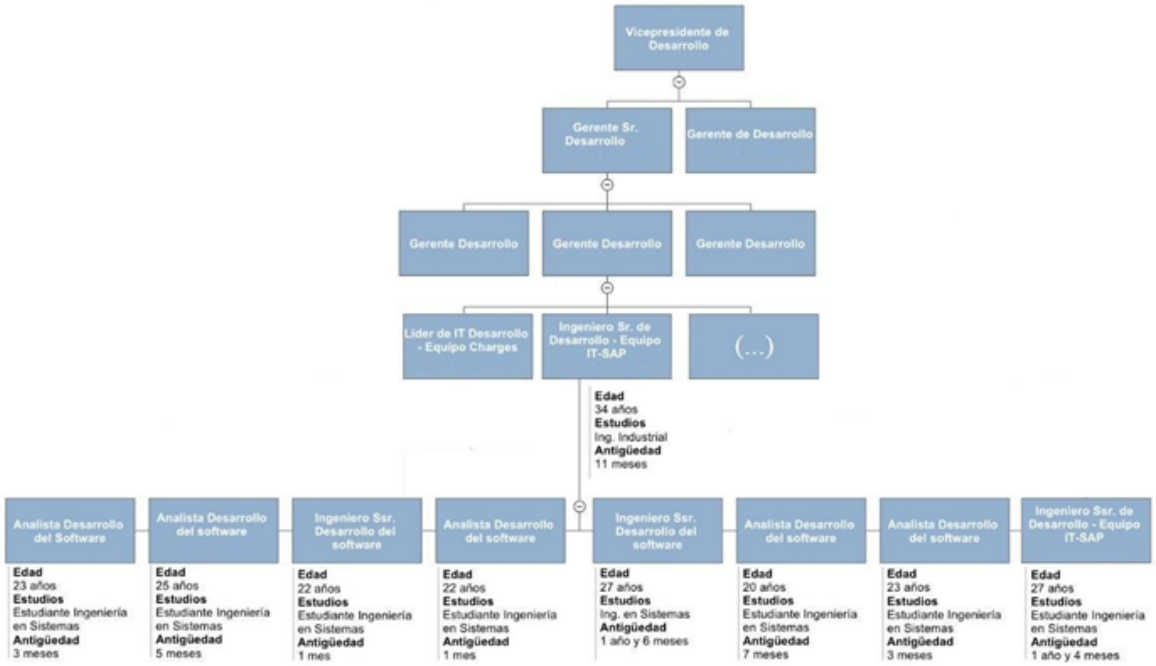

Figura 14: Organigrama y detalle del equipo IT-SAP. Fuente: Elaboración propia en base a información brindada por Recursos Humanos de Compañía de Tecnología S.R.L.

De la observación sobre el organigrama y su detalle, se destacan tres características comunes que comparten la mayoría de los ingenieros: La antigüedad en la compañía, los estudios alcanzados y el puesto organizacional.

De la antigüedad se observó que el 77% del equipo cuenta con menos de un año en la compañía. Esto puede dar a entender que solo el 23% del equipo puede llegar a tener un mayor conocimiento sobre la arquitectura de tecnología y las características del negocio dentro de la empresa.

Por otro lado, se observó el grado de estudios alcanzados dentro del equipo, siendo que solo el 23% de los mismos se encuentran recibidos con un título de grado, mientras que el restante aún se encuentra cursando la universidad.

Contar con personal con un título de grado asegura que los profesionales tengan un mayor conocimiento en áreas en las que no se desempeñan cotidianamente, por ejemplo que un estudiante de Ingeniería en Sistemas comprenda conceptos de contabilidad y auditoría.

Durante la entrevista, Esteban comentó lo siguiente en relación al conocimiento de esquemas contables al momento de realizar una desarrollo del software: "Mira, usualmente en el esquema no se suele indagar lo que veo yo por lo menos. (…) No evalúo si lo que me estás diciendo está bien o mal. No tengo mucho conocimiento quizá Diego que es más funcional, puede tener un poco más de esa discusión. Diego es Ingeniero Industrial tiene más conocimiento, y mucha más experiencia. (…) trato de entender un poco pero cuesta, no es tan fácil" (Esteban, entrevista personal, 23 de Enero 2018).

Por último se observó el puesto organizacional de todos los miembros que componen el equipo de IT-SAP, siendo que el mismo está compuesto en un 55% por analistas de desarrollo de software, 33% por ingenieros semi-seniors y un 11% por un ingeniero senior.

Dada la baja cantidad de personal experimentado en el equipo se consultó a Diego sobre la razón por la cual el equipo cuenta con dicha distribución, comentó que

es complejo contratar ingenieros senior de desarrollo del software debido a dos principales motivos: La tecnología que utiliza la empresa es de punta y los recursos que comprenden dicha tecnología, se encuentran en empresas de tecnología donde la remuneración es elevada y es difícil la contratación. Diego comentó: "Conseguir un senior es muy difícil (…) primero porque la tecnología que utiliza Compañía de Tecnología S.R.L es muy nueva y es de punta, siendo que en muy pocas empresas lo utilizan, y segundo porque los seniors están en empresas importantes donde cobran muy bien y es difícil pasarlos, entonces lo que más se consigue son analistas, juniors o a lo sumo semi-seniors". (Diego, entrevista personal, 23 de Enero 2018).

Debido a que el 88% del equipo está conformado por analistas de desarrollo del software e ingenieros Semi-seniors, la comunicación interna del equipo se ve afectada negativamente por la falta de experiencia y el comportamiento que caracteriza a estos puestos organizacionales. Durante la entrevista a Diego y Esteban, ambos coincidieron en la existencia de problemas en la comunicación interna dentro del equipo: Según Esteban: "Yo creo que quizá en comunicación hay algunos problemas, creo que falta comunicación, más que nada para comunicar cosas que cuando uno está atrasado o necesita más ayuda. (…) creo que la comunicación falta un poco, mejoramos pero hay cosas que no se comunican del todo bien". (Esteban, entrevista personal, 23 de Enero 2018).

Mientras que para Diego, referido a la comunicación dentro del equipo aseguró: "Dentro de todo estuvimos mejorando bastante este último tiempo (…) La verdad es que el área de sistemas en sí, el aspecto de la comunicación es flojo. Son difíciles y se suelen

perder cosas en el medio, la mejor forma que creo es para mantenerse comunicado es hacer “daylies” o sino cada 2 o 3 días una reunión para que puedan comunicar como es el grado de avance. (…) Yo creo que también tiene que ver con el seniorty, no hay ningún senior y eso es un tema que creo que hay que ajustar en todos los chicos de cara al progreso personal”. (Diego, entrevista personal, 23 de Enero 2018).

Esta falta de comunicación dentro del equipo puede afectar a los proyectos de desarrollo debido a que no se informa oportunamente los desvíos en la planificación por demoras en el desarrollo del software o espera en la respuesta de otros equipos, Diego comentó: “Yo creo que digamos la dinámica de trabajo que se debería seguir (…) es enviar digamos mails diarios o cada vez que hay una actualización, avisar che mira que esto está en este estado, lo tiene esta persona o no puedo seguir por falta de esta parte. Hay algunos de los chicos que lo hacen pero creo que es muy difícil que el individuo que es de sistemas realizar esa tarea”. (Diego, entrevista personal, 23 de Enero 2018).

Durante la entrevista se indagó a los entrevistados por los beneficios que heredan por pertenecer a su subgrupo de desarrollo de software por sobre los beneficios corporativos comunes al personal de la compañía. Ambos estuvieron de acuerdo en que cada subgrupo de desarrollo dentro de Compañía de Tecnología S.R.L define sus propios beneficios, así como también la metodología de trabajo y las herramientas que utiliza. Durante la entrevista con Esteban comentó que cada subgrupo cuenta con sus propios beneficios que son definidos por los jefes de cada uno, y que están por encima de los beneficios corporativos: “Son flexibles mis jefes, los días de trabajo desde casa que me quiera tomar, me los tomo. Eso se maneja dentro del equipo, digamos que dentro

del equipo tu jefe es el que define. Suele ser así, pero depende del equipo (…) yo entre en Septiembre, y me tenía que tomar 3 días más de vacaciones y no pasa nada, lo hable con ellos directo pero eso es dentro del equipo, es mejor". (Esteban, entrevista personal, 23 de Enero 2018).

Asimismo Diego, cuando se indagó sobre el mismo tema, comentó que la razón por la cual cada subgrupo define su propios beneficios se debe a que el área de sistemas es muy particular y debido a la gran demanda de ingenieros de software que tiene la industria, es necesario contar con cierta flexibilidad para poder mantener a los recursos dentro de la empresa. Esto se ve reflejado cuando comentó "Sistemas es un área muy particular. En un área común como administración la gente como que es más estructurada por lo que la oferta es más grande que la demanda de laburo, en sistemas pasa al revés. Compañía de Tecnología S.R.L necesita más gente de la que hay, y para sostener eso necesitas un ambiente dinámico más aún cuando la rotación (en términos de egreso e ingreso) es muy alta. Entonces necesitas tener cierta flexibilidad y en general se labura por proyectos por lo que si necesitas entrar a las 10 hs e irte a las 17 hs no hay problema siempre y cuando termines el proyecto". (Diego, entrevista personal, 23 de Enero 2018).

Durante la entrevista Diego comentó que los beneficios no solo se reducen a flexibilidad horaria y más cantidad de días de vacaciones, sino que también definen sus propios sueldos, utilizan sus propias herramientas y puede reclutar al personal técnico que ellos quieran. Cuando se le consultó sobre este tema, explicó: "Si vos estás pensando en que tenes que ir a la facultad y no sabes si llegas a tiempo vas a bajar el

rendimiento (…) lo mismo con el sueldo, intento ver si cada uno está a gusto con el sueldo, si vos no llegas a pagar el alquiler es un problema más que tenés en la vida y te desgasta mentalmente (…) La selección de herramientas (para programar y documentar) es totalmente libre según las necesidades y la dinámica del subgrupo (…) usamos lo que nos parece más cómodo y lo que mejore nuestra productividad (…) nosotros tenemos "hiring" (contratación de personal) abierto (…) vemos que perfil para que equipo puede ir, y puedo decidir yo si entra, así como también puede decidir otra persona (…) si le va bien le mandamos un examen técnico, si lo pasa bien se le define un seniority, un sueldo y entra". (Diego, entrevista personal, 23 de Enero 2018).

De lo comentado anteriormente se observó que cada subgrupo crea y mantiene sus beneficios, los cuales aplican para los individuos del mismo y están por encima de los beneficios corporativos. Asimismo, se puede vislumbró que los privilegios incluyen mayor flexibilidad horaria, aumento de los días de vacaciones con goce de sueldo, aumento de sueldos propios (con aprobación del sector de recursos humanos), contratación directa personal para el desarrollo del software y definición y uso de herramientas productivas acorde a las comodidades de cada subequipo.

Al momento de indagar sobre las rotaciones a nivel interno de Compañía de Tecnología S.R.L se observó que existen diferencias al momento de definir el grado de rotaciones dentro de la empresa entre miembros del equipo de IT-SAP.

Por un lado Esteban, comentó que existe rotación dentro de la compañía pero que la misma está reservada a niveles jerárquicos por encima de los desarrolladores debido

ya que cuentan con un nivel de especialización más elevado y un mayor grado de conocimiento de la arquitectura. Según explicó, "La rotación es a niveles más altos. Es gente que me parece que tiene más conocimiento del negocio y que puede ser más funcional a darte una mano. Porque quizá a mí me moves al equipo de cargos (otro sector) y quizá no puedo darle una mano mucho mayor y tengo que ir aprendiendo. Quizá alguien que está hace más tiempo, entiende más del negocio y moverlo es más fácil porque es más fácil adaptarse". (Esteban, entrevista personal, 23 de Enero 2018).

Por otro lado Diego, el Supervisor de Esteban, comentó durante la entrevista que dentro de su equipo existieron rotaciones a niveles bajos jerárquicos, debido a problemas dentro del grupo o porque a los desarrolladores no les parecía desafiante el trabajo que realizaban. Esto se observó en los siguientes comentarios sobre las rotaciones durante la entrevista: "En un momento creo que éramos 5 o 6 personas que conformábamos un equipo bastante friccionado en ese momento. Mucha fricción por lo que básicamente se disolvió y quedamos Walter y yo. Dos de los chicos se fueron a un equipo de datos fiscales y otro se fue para un equipo de tecnología. Eso a nivel rotaciones que ocurrieron, ahora hay una de las chicas que tiene ganas de cambiarse a otro equipo y estamos trabajando en eso, (…) en alguno de los casos puede llegar a ser que existan tareas operativas o tareas un poco aburridas para algunos sin tanto contenido técnico como busca un ingeniero en sistemas". (Diego, entrevista personal, 23 de Enero 2018).

Se observó que existe una diferencia al momento de definir el grado de rotación dentro del equipo ya que para Esteban la rotación está directamente relacionada con el grado de conocimiento del negocio y la arquitectura, por lo que es el desconocimiento de

la misma una barrera de salida hacia otros equipos impidiendo la rotación. Mientras que para Diego, la rotación se realiza cuando alguno de los miembros del equipo está disconforme con sus compañeros o con el trabajo que desempeña. Si bien hay diferencias ambos coinciden en que existe rotación dentro de la compañía.

Se indagó a los entrevistados por la distribución del trabajo en relación a las tareas cotidianas que realizan, para ello se les solicitó que distribuyan en términos porcentuales la porción de trabajo que requiere cada tarea usualmente. Se adjunta a continuación una tabla que resume lo descripto por los entrevistados.

Tabla 5: Distribución de trabajo operativo según entrevistados y puesto.

Tarea / Participante	Según Diego - Líder IT	Según Diego - Desarrolladores	Según Esteban - Desarrolladores
Confección del backlog y definición funcional	50%	20%	10%
Soporte al equipo	50%	-	-
Desarrollo de software	-	40%	25%
Soporte y mantenimiento	-	30%	25%
Pruebas unitarias y revisión de código	-	5%	5%
Pruebas integradoras	-	5%	35%
Puesta en producción y documentación	-	Bajo	Bajo
Retrospectiva	-	-	-

Tabla 5: Distribución del trabajo operativo en términos porcentuales según entrevistados y puesto. Elaboración propia.

Entrevista en base a la investigación forense

Con el fin de estructurar los temas que fueron tratados durante la entrevista con el equipo de IT-SAP a fin de poder reconstruir los hechos que sucedieron el pasado Julio del 2017 donde se introdujo software con error en el ambiente productivo acorde al informe R.C.A, se utilizaron las fases de la metodología de desarrollo Scrum analizadas en el marco teórico del presente trabajo con el objetivo de facilitar el análisis y comprensión de los hechos.

En la figura 15, se observan las secciones que componen el marco de trabajo de

la metodología de Scrum y su descomposición en módulos.

Figura 15: Etapas de la metodología Scrum y su división en tareas. Fuente: Elaboración propia en base a (Rubín, 2012).

Definición trimestral de iniciativas y asignación de iniciativas

Trimestralmente los directores y gerentes de Compañía de Tecnología S.R.L realizan una reunión en conjunto entre las áreas de desarrollo y negocio con el fin de compartir las iniciativas que deberán ser llevadas a cabo durante los próximos 3 meses y determinar qué equipos serán encargados de desarrollar la solución.

Estas iniciativas incluyen nuevos productos como parte de una innovación, modificaciones a productos existentes para mejorar el rendimiento de los mismos o modificaciones de un producto por cuestiones de contexto como temas impositivos, legales o de la competencia.

En la reunión del segundo trimestre del 2017, se definió continuar con el desarrollo de Créditos Especiales, y se solicitó como iniciativa la creación de tres variantes de un proceso existente con el fin de obtener un ahorro impositivo en tres

escenarios en particular. Acorde a lo que comentó Diego en la reunión: "(…) habían definido un esquema contable (…) utilizaron este mismo esquema contable para registrar otros tipos de actividades (extensión de movimiento) y eso hacía que se generarán asientos en cuentas incorrectas". (Diego, entrevista personal, 23 de Enero 2018).

Como conclusión de lo expuesto anteriormente, no se observan conflictos a la hora de la definición de iniciativas, ni de comunicación en base a lo relevado. Por este motivo no hay indicios de que las reuniones trimestrales para la definición de iniciativas puedan haber influido negativamente en el desarrollo de la iniciativa asociada a los esquemas contables.

Figura 16, se observan los elementos que intervienen en la definición trimestral de iniciativas y asignación de iniciativas.

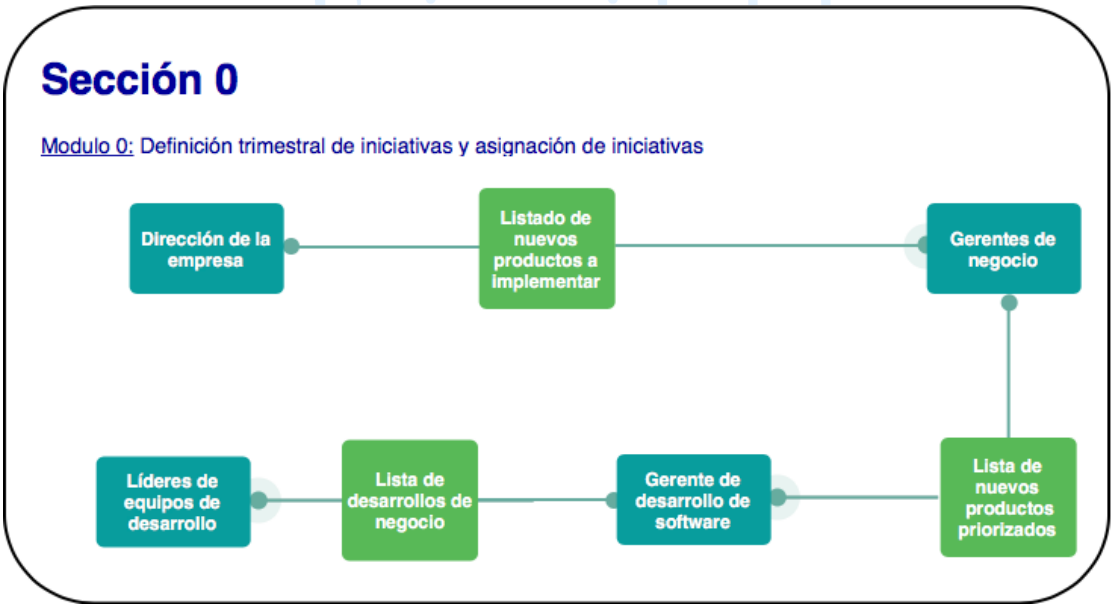

Figura 16: Diagrama de actor-red para la definición trimestral de iniciativas y asignación de iniciativas según entrevistas realizadas. Fuente: Elaboración propia en base a las entrevistas realizadas.

Confección del backlog

Todos los principios de mes, los equipos de negocios y tecnología se reúnen para confeccionar el backlog de tareas a realizar. Estas reuniones son precedidas por el equipo de negocios, y buscan analizar qué tareas deben ser realizadas.

Durante la reunión se priorizaron los temas que fueron definidos como iniciativas y, dependiendo la carga de trabajo que tienen ambos equipos, planifican el orden y los temas que deben ser desarrollados en el mes. Adicionalmente, se verifica si quedó alguna iniciativa sin realizar que fue planificada en el mes anterior, y en caso de que exista se reconfirma para la planificación del mes. Por último, los gerentes del negocio y líderes de los equipos de desarrollo seleccionan por tema qué recursos van a ser los responsables para la realización de la solución.

Esta selección de recursos no se realiza mediante un proceso establecido sino que es dinámico. Se tienen en cuenta las variables de capacidad del trabajo en términos de conocimiento, demanda de carga operativa del sector y complejidad de la tarea a realizar. Según Pablo, Gerente de Facturación y Cobranzas, no siempre es posible asignar a la persona más capacitada para la realización de dicha tarea ya que por alguna de las tres variables no puede tomar la misma. Pablo comenta: "(…) es complejo, vemos la gente más capacitada que tenemos y a los más simples (por los proyectos) se los

asignamos a los que no tienen tanta experiencia (…) hay veces que te sale perfecto (…) y otras que tenés que hacer malabares para que quede lo mejor organizado posible (…) hay veces que hay gente que te gustaría que tome el proyecto y no la termina tomando porque esa persona está colapsada por otras tareas (…)

Como conclusión de lo comentado por Pablo, se observaron problemas a la hora de definir los recursos a los proyectos, debido a la carga operativa que sus recursos tienen en relación a las tareas que desempeñan cotidianamente en la empresa. La asignación de personal sin experiencia para la definición de los requerimientos puede desestabilizar la conexión y que existan problemas en el intermediario ya que posibilita la definición incorrecta de esquemas contables, o falten requerimientos para la realización de la funcionalidad y de este modo atrase al proyecto.

Durante el mes de Junio 2017 el equipo de desarrollo Facturación y Cobranzas se reunió con el equipo del IT SAP para analizar las tareas definidas que debían ser desarrolladas para el mes. Dentro de las tareas identificadas se encontraba la realización de una extensión de movimientos como parte de una modificación de uno de los procesos de Créditos Especiales con el fin de lograr el ahorro impositivo.

Como conclusión de lo comentado por los entrevistados, no se observó conflicto alguno en términos de comunicación entre los equipos de trabajo al momento de priorizar las tareas y asignar recursos a las mismas. Sin embargo, se vislumbró que existen problemas a la hora de asignar recursos a los proyectos por parte del equipo de negocio debido a que sus recursos no cuentan con los tiempos operativos necesarios para

que desempeñen la tarea. Asimismo se observa que durante la reunión no existe una conexión estable para el análisis de retrospectiva en relación a cómo funcionaron los equipos de trabajo en el mes anterior, y si existió algún problema que deba ser contemplado para realizar las modificaciones necesarias y asegurar que no ocurra nuevamente.

Figura 17, se observan los elementos que intervienen en la confección del backlog según las entrevistas realizadas.

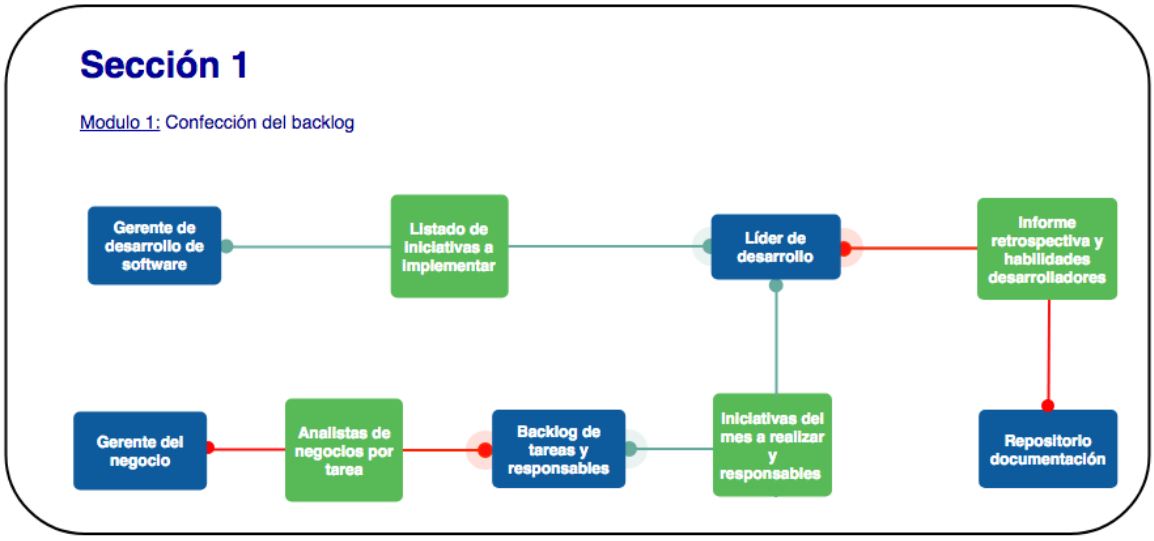

Figura 17: Diagrama de actor-red para la confección del backlog según entrevistas realizadas. Fuente: Elaboración propia en base a las entrevistas realizadas.

Definición funcional y requerimientos técnicos

Durante la entrevista se indagó sobre las reuniones de definición funcional y requerimientos técnicos a los entrevistados que ocurrieron en el proyecto de los esquemas contables de IT-SAP para Créditos Especiales.

Diego comentó que durante el mes de Julio se realizaron cuatro reuniones entre su equipo y el equipo de negocios para la definición de los requerimientos. Al momento de indagar sobre los participantes involucrados en las reuniones, Diego afirmó que él no pudo asistir a todas las reuniones pero que usualmente si asiste a esta clase de reuniones debido a dos principales factores en relación a su equipo:

1) Los desarrolladores de su equipo usualmente no conocen el objetivo del negocio por el cual es necesario el desarrollo. Por lo tanto la definición de los requerimientos esperados puede ser incorrecta o insuficiente, aumentando la probabilidad de errores en la realización del software. Diego comentó: "Para poder desarrollar funcionalidades digamos en buen estado para Compañía de Tecnología S.R.L, es necesario tener tantos conocimientos del negocio como aspectos funcionales. Imaginate que yo me pongo a armar un Excel con una columna que no tengo idea de que se trata, puede dar bien o puede dar mal". (Diego, entrevista personal, 23 de Enero 2018). Según Diego, la razón por la cual los desarrolladores no conocen el objetivo del negocio, es debido a la falta de interés que tienen ya que no son tareas específicamente de tecnología y requieren habilidades de comunicación y otras disciplinas para poder realizarlas correctamente, comenta: "Para algunos chicos puede resultar aburrido porque no tienen una carga de sistema que es la que ellos quieren, entonces miran solo la parte superficial. En esto suelo absorber yo un poco la parte funcional, (…) y contarles a ellos de que se trata el desarrollo(…) yo trato de que sepan que están haciendo y depende la persona algunos se interesan más y otros no".

En contraposición Esteban, desarrollador de IT-SAP, comentó que la razón por la cual no comprenden en su totalidad el objetivo del negocio, no es por una falta de interés sino por falta de experiencia, conocimiento del negocio, arquitectura y otras disciplinas como contabilidad. Esteban comenta que debido a su perfil académico le es difícil comprender lo que se está definiendo, "No evalúo si lo que me están diciendo está bien o está mal, no se mucho de contabilidad, (…) no tengo mucho conocimiento, quizá Diego que es más funcional puede tener un poco más de esa discusión, él es Ingeniero Industrial y tiene más conocimiento y mucha más experiencia que yo. (…) trato de entender [por el objetivo del negocio] pero cuesta (…) los del equipo también, todos lo intentan por lo menos." (Esteban, entrevista personal, 23 de Enero 2018).

2) Debido al seniority con el que los miembros del equipo cuentan (analistas y analistas semi-seniors) no cuentan con la experiencia suficiente y habilidades de comunicación para poder presidir una reunión de relevamiento de requerimientos.

Se indagó a los entrevistados sobre cómo fue la comunicación entre ambos equipos en términos de respuesta, predisposición y grado de conocimiento durante las cuatro reuniones de relevamiento que se realizaron en Julio del 2017 para la extensión de esquemas contables.

Diego comentó que existieron dos principales problemas que ocurrieron durante dichas reuniones y, que ocurren continuamente en los desarrollos actuales:

El equipo de negocio usualmente detalla menos requerimientos de software que los necesarios para la construcción de las funcionalidades. Por lo que al no contar con

todos los requerimientos bien definidos se atrasan los proyectos, ya que es necesario tener mayor información para continuar con la construcción de la funcionalidad.

Diego comentó: “Eso pasa continuamente, los equipos como Facturación y Cobranzas pasan menos requerimientos de los totales (…) y capaz que termina saliendo con errores porque idas y vueltas de información se puede perder algo” (Diego, entrevista personal, 23 de Enero 2018).

Cuando se le consultó al gerente del Facturación y Cobranzas sobre la comunicación al momento de definir los requerimientos del software afirmó que usualmente el personal de desarrollo del software cuenta con otra estructura y no conocen tanto el negocio, por lo que es complicado la comunicación para la definición de los requerimientos. Pablo, Gerente de Facturación y Cobranzas comenta al momento de consultarle por las reuniones semanales de definición funcional: “Eso lo vamos trabajando con IT, obviamente tienen otra formación, otra estructura y demás por lo que no conocen tanto el negocio como nosotros (…) siempre buscamos que lo que desarrollan sepan que es lo que están haciendo porque eso favorece al desarrollo (…) evita que se equivoquen o tomen acciones incorrectas (…) ha pasado a veces, que hay errores en el código porque la persona no entiende para que esta codeando (…) por eso es importante que entienda del lado del producto para que están haciéndolo”. (Pablo, entrevista personal, 9 de Febrero 2018).

Como conclusión del primer problema recurrente identificado por Diego, se observó que se definieron menos requerimientos funcionales debido a la falta de

conocimiento del producto por parte del equipo de desarrollo para indagar más asertivamente sobre la funcionalidad a realizar, y asegurar que no se omita ningún detalle indispensable para la realización de la misma. Asimismo se observó que el equipo del negocio no demuestra una predisposición para identificar todas las aristas relacionadas con el desarrollo del producto debido seguramente a una falta de tiempo o conocimiento del producto a realizar.

El otro problema de comunicación con el equipo de negocios que identificó Diego, fue que durante la etapa en la que se definieron las tareas a incluir en el backlog mensual, se planificaron iniciativas a realizar de las cuales aún no se encuentran en condiciones de ser realizadas pero sí se sabe la fecha en que debe salir a producción debido a la competencia. Por lo que la tarea es reemplazada por otra hasta tanto se encuentren las condiciones de comenzar a desarrollarlo y se dispone de menos tiempo para la realización de la funcionalidad ya que varía la fecha de inicio del desarrollo pero no la fecha de finalización. Diego afirmó: "Se pueden definir iniciativas que todavía no están definidas las condiciones y después durante el mes no están y terminas no haciendo o reemplazando por otras (…) a veces hay iniciativas que tienen que salir sí o sí (…) y todavía no tenemos la información (…) si estás sobre reloj, no podes hacer pruebas y no tenés tiempo, y lo obligas a pasar a producción. Hay un grado de probabilidad de que pase un error, para el caso de esquemas contables paso algo similar". (Diego, entrevista personal, 23 de Enero 2018).

Durante la entrevista con Pablo, Gerente de Facturación y Cobranzas, se le consultó si considera que su equipo tiene los recursos necesarios para entender la

demanda de la compañía en términos de definición de nuevos productos, y tareas operativas que caracterizan al sector. Pablo comentó que está seguro de que le faltan más recursos sin duda, para poder tomar más temas y llegar más en profundidad con algunos tópicos.

Como conclusión del segundo problema identificado por Diego, se observa que dado el riesgo inherente del negocio en el que Compañía de Tecnología S.R.L se encuentra inmerso, es inevitable que ciertos productos tengan una fecha de lanzamiento estipulada por fines estratégicos frente a la compañía, y si el equipo de negocios encargado de definir las funcionalidades del producto no cuenta con el tiempo suficiente para realizar dicha tarea, es probable que se retrase el desarrollo del software obligando al equipo de desarrollo a suprimir alguna de las etapas en la metodología de desarrollo y por lo tanto aumente la probabilidad de que existan errores en producción.

Ambos entrevistados coincidieron en que la documentación realizada en esta etapa es poca o nula. Según Diego no es común generar documentación en todas las reuniones que se realizan con el equipo de negocios, y dependiendo la criticidad se puede sacar una foto a la propuesta de la solución pero no es lo común en su equipo, ni tampoco en los otros equipos de desarrollo de Compañía de Tecnología S.R.L. Diego afirmó: “No somos muy buenos documentando, a veces el resultado de una reunión si es importante, mandamos una foto con la definición de la arquitectura, pero no solemos mandar mails. En general, no vi equipos que hagan documentos, hay muy pocos (…). Quizá a lo sumo, mandamos un mail si es muy importante, sino siempre después de una

reunión se ponen a laburar (por los desarrolladores) entonces no se pierden sobre eso". (Diego, entrevista personal, 23 de Enero).

Asimismo, Esteban en coincidencia con Diego afirmó que no es frecuente la documentación pero que se suele enviar algún mail con los requerimientos para formalizar, de lo contrario todo se realiza vía chat.

Particularmente al momento de reconstruir el caso para la investigación forense se observó que durante las 4 reuniones realizadas por el equipo de desarrollo y el negocio (etapa de definición funcional y requerimientos técnicos), existió un problema de comunicación debido a que se decidió utilizar un esquema contable que ya se encontraba en uso para un proceso, el cual resultaba incorrecto para el nuevo proceso en que se estaba definiendo.

Por un lado se detectó una conexión inestable originada por el equipo de desarrollo que no contó con los conocimientos contables necesarios para darse cuenta de que el esquema contable definido no tenía correspondencia con el nuevo proceso a desarrollar, posiblemente sea por la combinación de seniority de los miembros del equipo de desarrollo, sumado a su falta de conocimiento en la arquitectura y contabilidad.

Por otro lado, el equipo de Facturación y Cobranzas solicitó reutilizar incorrectamente el esquema contable que estaba configurado para otro proceso seguramente debido a la falta de tiempo para analizar el proceso y tomar una decisión.

Figura 18, se observan los elementos que intervienen en la definición funcional y requerimientos técnicos según entrevista realizada.

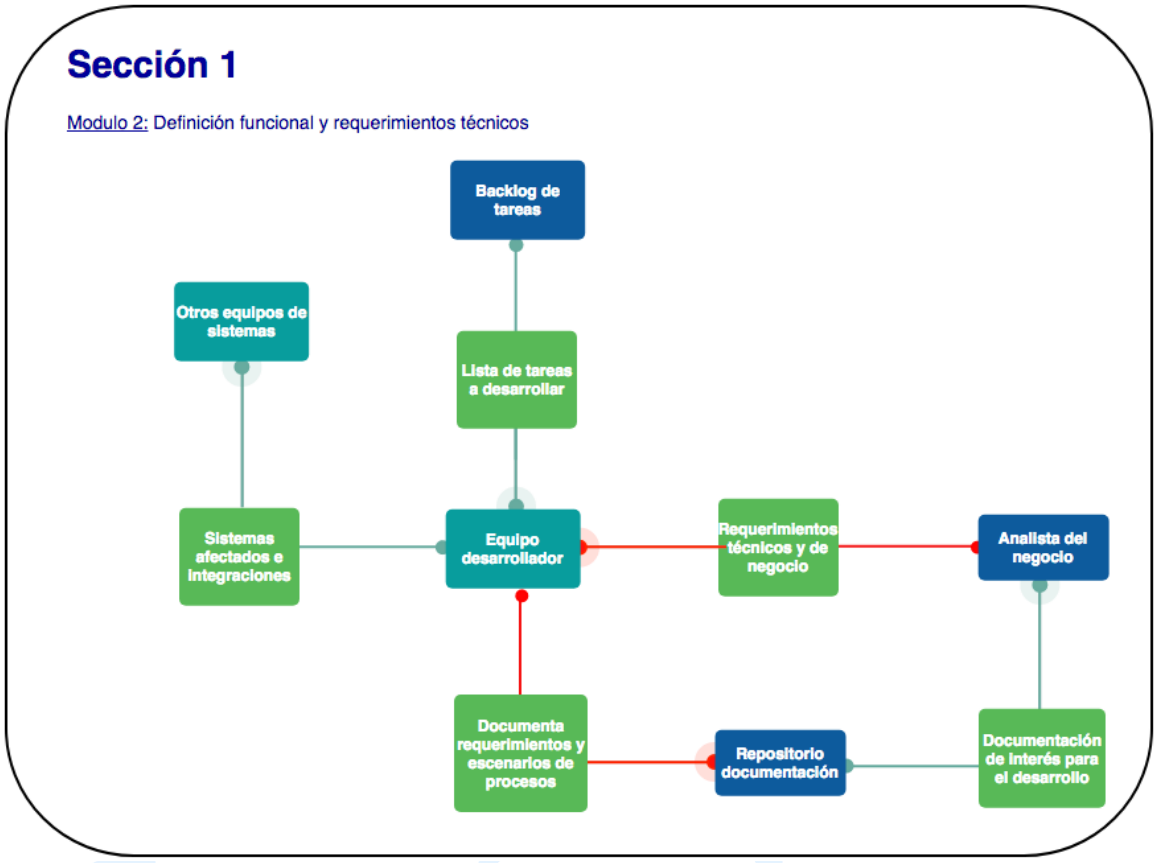

Figura 18: Diagrama de actor-red para la definición funcional y requerimientos técnicos según entrevistas realizadas. Fuente: Elaboración propia según entrevistas realizadas.

Desarrollo del software

Durante esta etapa, el desarrollador utilizó la herramienta de desarrollo del software que utilizan dentro del equipo de IT-SAP y, según lo relevado durante la reunión de definición funcional y requerimientos técnicos, procedió a desarrollar dichas funcionalidades acorde a los esperado.

Según Esteban, al momento de llevar a cabo el desarrollo se tiene en cuenta la salida esperada que tiene que generar la funcionalidad para que otros equipos puedan utilizarla e integrar el proceso.

Durante la entrevista con los miembros del equipo de IT-SAP no se observó que existiese una complejidad o problemática al momento de realizar el desarrollo del software siendo que el mismo es desarrollado acorde a los requerimientos relevados en las reuniones y acorde a lo esperado.

Figura 19, se observan los elementos que intervienen en el desarrollo del software según entrevistas realizadas.

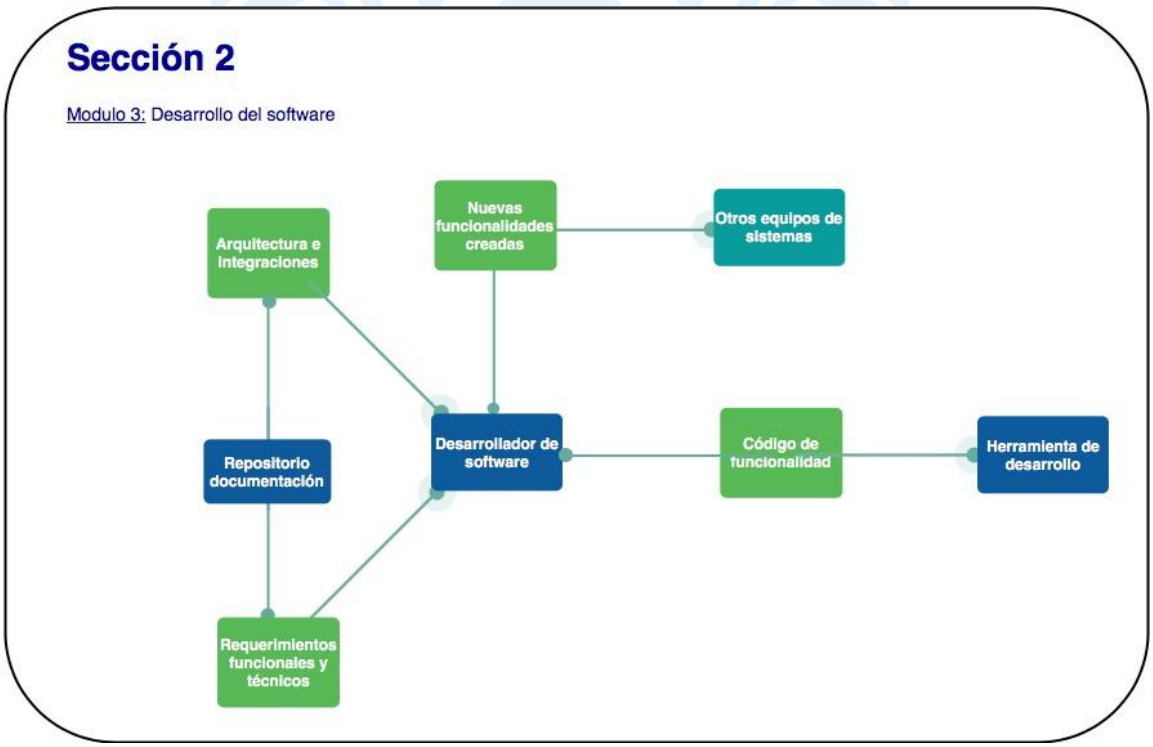

Figura 19: Diagrama de actor-red para el desarrollo del software según entrevistas realizadas. Fuente: Elaboración propia en base a la entrevistas.

Pruebas

Se consultó a los entrevistados sobre cómo es la realización de las pruebas para los desarrollos de software realizados por ingenieros del equipo. Los entrevistados coinciden en que se realizan pruebas unitarias e integradoras para todos los desarrollos. Diego da una explicación del porqué las pruebas son tan importantes: "Las pruebas se hacen siempre, no se sube nada a producción sin probar porque históricamente cada cosa que se subía a producción (sin probar) fallaba, y cualquier falla afecta al cierre. (…) el cierre contable es hito importante para la empresa porque es el momento en que Compañía de Tecnología S.R.L le presenta la información financiera a los accionistas (…) cualquier desviación en los números (…) genera un problema". (Diego, entrevista personal, 23 de Enero del 2018).

Pruebas unitarias y revisión de código

Durante cada prueba unitaria el desarrollador prueba su software según el requerimiento solicitado por el negocio. La complejidad que se le da a la prueba unitaria está dada por el grado de conocimiento que tiene el desarrollador sobre el proceso que modifica y los procesos que se relacionan con él, por lo que si el desarrollador no tiene en cuenta estas relaciones, sólo se estará asegurando que el código cumpla con lo esperado pero no evalúa el impacto con otros procesos que se relaciona.

Según los entrevistados debido a la complejidad de la arquitectura de Compañía de Tecnología S.R.L y, la cantidad de escenarios que dispone, no es posible probar ante

cada desarrollo de funcionalidad cada uno de los escenarios para asegurar que el nuevo código funcione correctamente. Por este motivo los casos de prueba se limitan a asegurar que el proceso definido y sus relaciones, funcionen acorde a lo esperado.

Esta limitación en los escenarios de prueba generó un riesgo que es aceptado por el equipo de desarrollo y consta de la posibilidad de que no se haya considerado un escenario que si sea afectado por el nuevo código y éste deje de funcionar o funcione incorrectamente.

Con el fin de minimizar la probabilidad de que este error sea materializado al no ser detectado por las pruebas unitarias, se realizó una revisión del código por parte de alguno de los miembros del equipo con el fin de asegurar que el código no afecte a algún escenario no contemplado en las pruebas. Esteban explica: "(…) el sistema es muy complejo, y tiene miles de movimientos (…) no podes pasar todos los movimientos cada vez que queres hacer una prueba porque no terminas más, (…) los tiempos se hacen larguísimos. No escala eso, se pide (probar) únicamente lo que se está cambiando. (…) se corre el riesgo de que se pueda tocar otra cosa, obviamente dentro del equipo hay review (por revisión del código) de lo se tocó no afecte a otro lado, pero siempre se corre el riesgo" (Esteban, entrevista personal, 23 de Enero 2018).

Asimismo, Diego haciendo referencia a la definición de pruebas unitarias mencionó: "Son casos de pruebas que manda la gente de administración según el esquema contable, (…) si hay algún movimiento que genera un asiento, agarramos ese

caso y lo pasamos por la interfaz (…) para probarlo. Nosotros probamos así, no probamos en volumen". (Diego, entrevista personal, 23 de Enero 2018).

Como conclusión de las pruebas unitarias que realiza el equipo de desarrollo, se observó que las pruebas fueron realizadas de manera acotada por los desarrolladores quienes definieron los escenarios a probar según el conocimiento que contaban sobre el proceso y sus relaciones.

Esto posibilita la existencia de la probabilidad de que no se estén considerando relaciones asociadas al proceso modificado dada la poca experiencia dentro de la compañía que cuentan la mayor cantidad de desarrolladores del equipo.

Adicionalmente, según la evidencia obtenida y observado el comportamiento de Diego durante la entrevista, la fase de revisión del código por parte de un miembro del equipo distinto al desarrollador, no se asegura que se realice en el 100% de los casos o disminuya la probabilidad de los errores en al menos en una cantidad significativa.

Particularmente referido a la investigación forense sobre los esquemas contables, no se observó una incorrecta implementación de esta etapa, ya que lo desarrollado fue probado y funcionaba acorde a lo definido en la etapa de definición funcional y requerimientos técnicos. Por último, se observó que el código fue revisado por otro desarrollador del equipo donde la prueba resultó satisfactoria no evidenciando que el esquema contable se encontraba incorrectamente utilizado.

En la figura 20, se observa los elementos que intervienen en las pruebas unitarias y revisión del código.

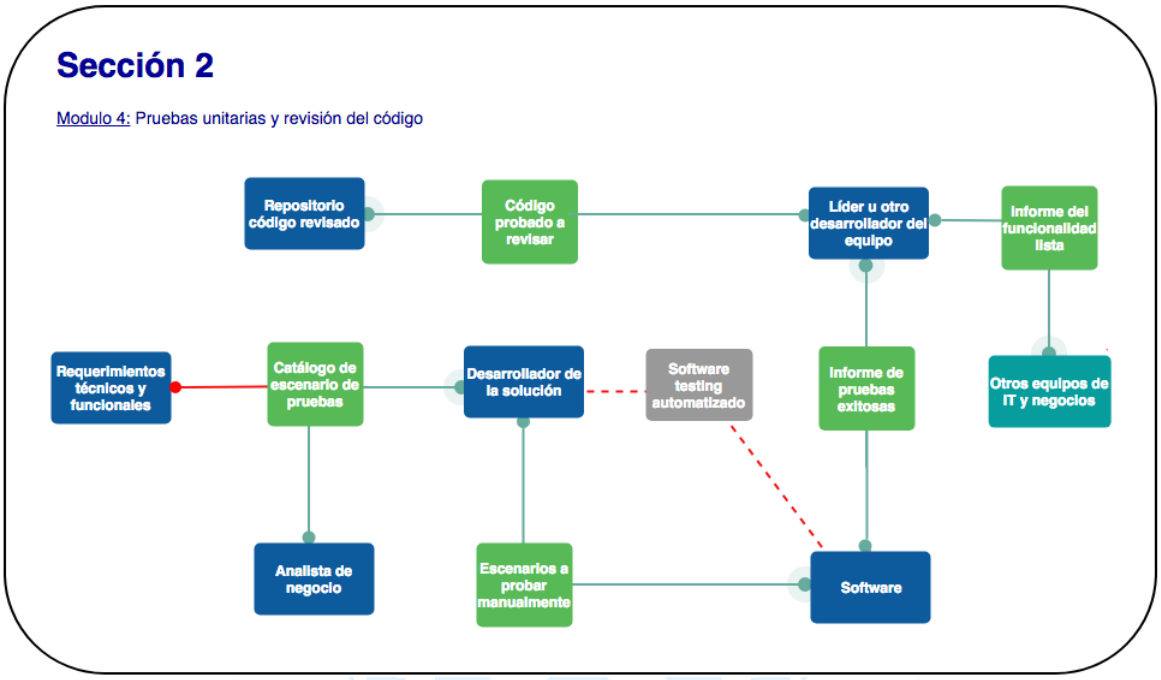

Figura 20: Diagrama de actor-red para las pruebas unitarias y revisión del código según entrevistas realizadas. Fuente: Elaboración propia en base a las entrevistas realizadas.

Pruebas integradoras

Para realizar las pruebas integradoras, las funcionalidades desarrolladas por los equipos de desarrollo son transportadas a un ambiente llamado “Calidad” en el que se ejecutan una serie de casos de prueba para asegurar que el proceso funcione correctamente.

El universo de casos a probar es fue definido durante la reunión de seguimiento realizada junto al equipo del negocio.

Según Pablo, Gerente de Facturación y Cobranzas, en la reunión semanal junto a la definición de definición funcional y requerimientos técnicos, ambos equipos definen

los casos de prueba según el proceso que se modifica. Esta decisión es tomada entre las personas que participan de la reunión y no existe una manera específica de acordarlas, por lo que se definen en base a la experiencia de cada participante.

Durante la realización de pruebas integradoras, el equipo de desarrollo habilitó el entorno de pruebas y se verificó el resultado del proceso para cada caso de prueba. En caso de que exista un error, se analizó la razón del mismo y se procedió a corregirlo para nuevamente comenzar con las pruebas.

Durante la entrevista con Pablo se indagó sobre si ha pasado que las pruebas integradoras hayan concluido exitosamente y luego se detectó un error en producción y nos comentó: “Si, ha pasado. A veces las pruebas no se definió bien el universo de pruebas a realizar, no se consideró un escenario o había algo en el ambiente de calidad que estaba configurado de una manera y en producción otra (…) generalmente pasa porque a veces es difícil prever todos los posibles escenarios (…) pero siempre por un tipo de caso, por la cantidad de usuarios y combinaciones pueden dar y eso hace que pinche (genere un error en producción)” (Pablo, entrevista personal, 9 de Febrero 2018).

Como primera conclusión de lo expuesto anteriormente, se observó que no existe una manera clara de definir las pruebas integradoras debido a la cantidad de escenarios posibles que cuenta la arquitectura de Compañía de Tecnología S.R.L. Los escenarios fueron definidos en base a la experiencia de los participantes de la reunión por lo que los factores como nivel de conocimiento del negocio, arquitectura y experiencia dentro de la

empresa resultan críticos para la definición los escenarios de pruebas que aseguren que el software funciona correctamente.

Por otro lado del análisis de la Tabla 4, se observó que difiere porcentaje de distribución de trabajo para las pruebas integradoras según el líder del área (5%) y los desarrolladores de software (35%). Según Esteban: "En pruebas integradoras se pierde mucho tiempo, en realidad son pruebas nuestras que como trabajamos con otros equipos por ejemplo el de Facturación y Cobranzas dependemos de ellos, y quizá tardan mucho en dar una respuesta (…) por ejemplo para el modelo contable de Brasil que lo estoy haciendo con ayuda de Diego para la parte funcional, mande las pruebas el Viernes o Lunes pasado y el Lunes de la otra semana recibí la prueba, digamos que estuve una semana esperando una respuesta, y ahí se pierde tiempo porque me para todo el laburo y no puedo hacer nada hasta que me confirmen que las pruebas están correctas. Ellos tardan y yo pierdo mucho tiempo ahí, hay mucha ineficiencia en esto.". (Esteban, entrevista personal, 23 de Enero 2018).

Diego, durante la reunión realizó una mención sobre las pruebas integradoras junto a otros equipos: "Después, bueno el control (sobre lo desarrollado como pruebas integradoras) de Facturación y Cobranzas suele tardar un tanto". (Diego, entrevista personal, 23 de Enero 2018).

Según Pablo al momento de consultarle por las demoras que comentó el equipo de desarrollo, coincide en que la demora está relacionada con la carga de trabajo que

tiene su equipo para las demandas de la compañía (tareas operativas esperadas del sector), y adicionalmente tareas de desarrollo de nuevos productos.

Referido a la investigación forense sobre los esquemas incorrectamente definidos para una variante en el proceso de Créditos Especiales, se observó que si bien se han realizado las pruebas integradoras sobre el proceso final, las mismas no fueron correctamente realizadas ya que se consideraron como correctas las cuentas contables a las que se estaba imputando cuando en realidad no lo estaban.

Según Diego, referido a esta funcionalidad incorrectamente implementada: "Esto pasa más seguido, porque los modelos contables empiezan a tener demasiadas aristas. Ponele si tengo un modelo contable que le impacta a un usuario común por ejemplo, ahora ese usuario común pasa a ser uno especial cambia el modelo contable y puede ser que no lo detectes y lo detectes como un problema (…) es muy difícil detectar (…) porque el modelo contable te lo definen, (…) se envían a controlar y si están bien se suben a producción y así y todo, le podemos errar (…)" (Diego, entrevista personal, 23 de Enero 2018).

Diego nos comentó las razones por las cuales ocurrió este tema en base al informe R.C.A. identificado anteriormente: "Ocurrió por dos cosas: primero porque a veces es un chino (refiriéndose a la complejidad) (…) tenés que estar muy fino y a veces con los tiempos que se maneja, estar fino no es compatible. (…) hay muchas cosas que pueden fallar ahí, o del desconocimiento del negocio (…) hay veces que no se da cuenta la gente de Facturación y Cobranzas por el tiempo que está o porque no tiene

conocimiento de más cosas (…) uno de los chicos que estaba en Facturación y Control se fue (…) porque no le gustaba la parte de producto del negocio, y si no te gusta no estas fino (…) y puede haber problemas que pueden ser errores en producción o problemas de tiempos" (Diego, entrevista personal, 23 de Enero 2018).

Como conclusión de lo expuesto anteriormente, se observó que la red sobre pruebas integradoras cuenta con varias conexiones inestables generadas por falta de tiempo, una incorrecta planificación y disponibilidad de recursos en el equipo de negocios y, a la falta de conocimiento por parte del equipo de Facturación y Cobranzas para definir los escenarios que deben ser verificados. Por último la discrepancia en la asignación de tiempos operativos diarios dentro del equipo de desarrollo da a entender la existencia de un problema en la comunicación interna del equipo de desarrollo.

Figura 21, se observan los elementos que intervienen para las pruebas integradoras según entrevistas realizadas.

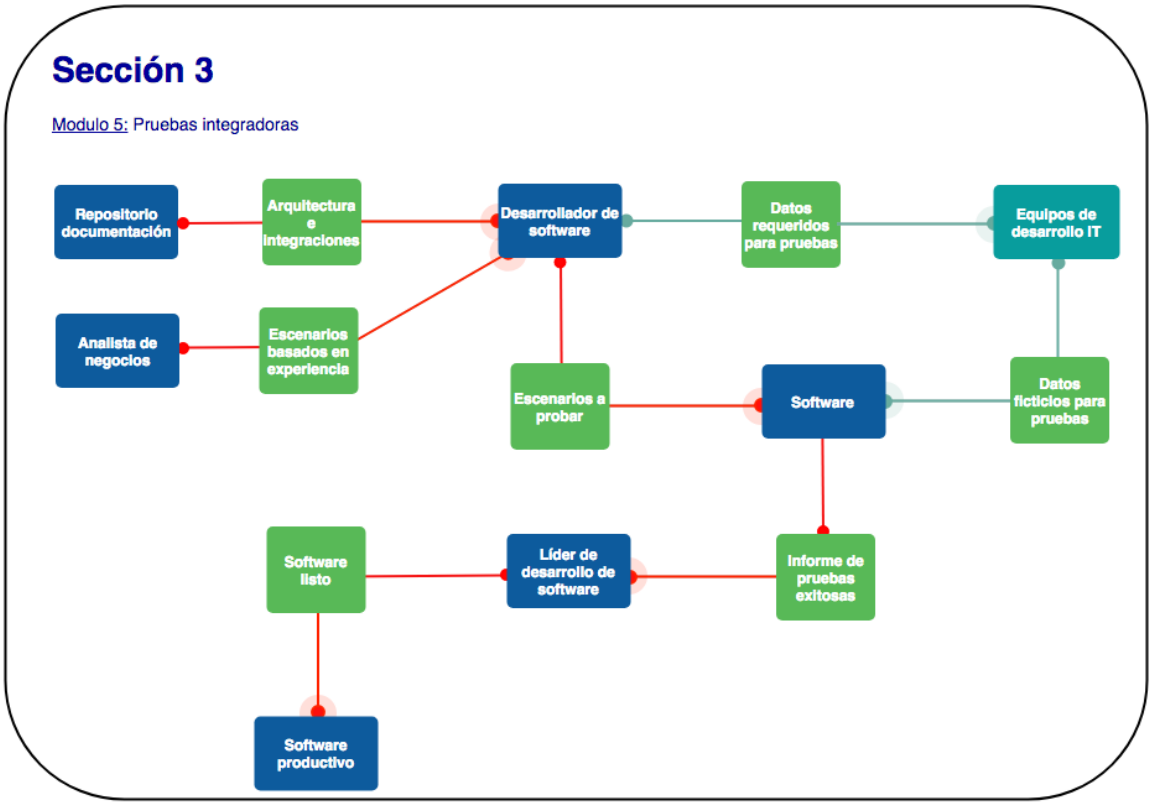

Figura 21: Diagrama de actor-red para las pruebas integradoras según entrevistas realizadas. Fuente: Elaboración propia en base a las entrevistas realizadas.

Puesta en producción y documentación

Durante la etapa de puesta en producción y documentación, se observó acorde a lo detallado por los entrevistados, que el equipo de desarrollo que luego de haber realizado las pruebas integradoras procede a subir a producción las nuevas funcionalidades.

Referido a temas de documentación, acorde a lo expuesto anteriormente, no se realizaron tareas de documentación sobre las funcionalidades desarrolladas con el fin de nutrir una base de conocimiento sobre las nuevas funcionalidades que cuenta la

arquitectura de Compañía de Tecnología S.R.L, así como tampoco se documentaron los escenarios de pruebas utilizados junto a sus resultados, y documentación posterior sobre el correcto funcionamiento de la funcionalidad, y en caso de que cuente con algún problema el detalle del mismo para ser considerado en futuros desarrollos.

Durante la revisión documental sobre la investigación forense del esquema contable incorrectamente implementado, no se halló información de importancia siendo que lo único que existía documentado sobre dicha funcionada era una tarea que compartía el líder del desarrollo con el desarrollador sobre las tareas que debía contemplar la nueva funcionalidad acorde a lo estipulado en el backlog.

Como conclusión se concluye que al no documentar las nuevas funcionalidades que se generan, así como tampoco los casos de prueba y posteriores problemas que ocurrieron en producción no es posible contar con una base del conocimiento que pueda ser consultada por los desarrolladores al momento de definir las funcionalidades y escenarios a probar con el fin de realizar un aprendizaje de los errores cometidos anteriormente, aumentar el conocimiento sobre la arquitectura en la que se encuentran desarrollado el software y considerar escenarios de pruebas similares para funcionalidades que comparten características.

Retrospectiva

Durante la entrevista realizada, ninguno de los líderes o recursos de desarrollo hicieron mención a la realización de un análisis de retrospectiva sobre cómo había sido el desempeño de sus equipos y como estos se relacionaron. Lo que nos da a entender que

no existe una etapa durante el desarrollo del software para analizar el desempeño de los equipos durante el desarrollo del producto.

Cuando se indago a Diego sobre este tema se mostró algo nervioso dando a entender que conocía de la utilidad de la retrospectiva pero que su equipo no la realizaba. Diego comentó: “Eso no, no lo hacemos. Se realiza a nivel departamental todos los trimestres (…) donde hablamos de todos los proyectos (…) como nos fue qué cosas hay que modificar (…). La verdad que el scrum teóricamente está bárbaro pero llegado a un momento (…) si haces demasiadas reuniones te rompe las bolas”. (Diego, entrevista personal, 23 de Enero 2018).

Como conclusión de lo expuesto anteriormente, se observó que no se realiza reuniones de retrospectiva para cada proyecto de desarrollo, lo que impide que se analicen las desviaciones en la planificación, rendimiento de los equipos o realizar un entendimiento de la razón por la cual un proyecto llevó un error a producción. Si bien se realizan reuniones trimestrales para tal fin, no existe evidencia alguna que indique que las mismas cumplan correctamente con su función, ya que al hacerlo departamentalmente y durante un día, no es posible analizar los proyectos de 120 empleados durante 3 meses.

4.4 Resumen de la investigación forense

Con el fin de reunir todas las conclusiones que se obtuvieron durante la desgravación de las entrevistas realizadas a los empleados de Compañía de Tecnología S.R.L, y conocer los sucesos que llevaron a que existiese un error en el ambiente productivo de Compañía de Tecnología S.R.L y genere una pérdida monetaria, se generó un diagrama de causa-efecto, el cual reúne todos los causales que ocurrieron.

El diagrama de causa efecto es denominado Diagrama de Ishikawa o Diagrama de espina de pescado, y permite representar gráficamente las relaciones múltiples de causa-efecto entre las diversas variables que intervienen en un proceso.

Figura 22, diagrama Ishikawa para comprensión de las causas identificadas en la investigación forense sobre el error en producción de los esquemas contable.

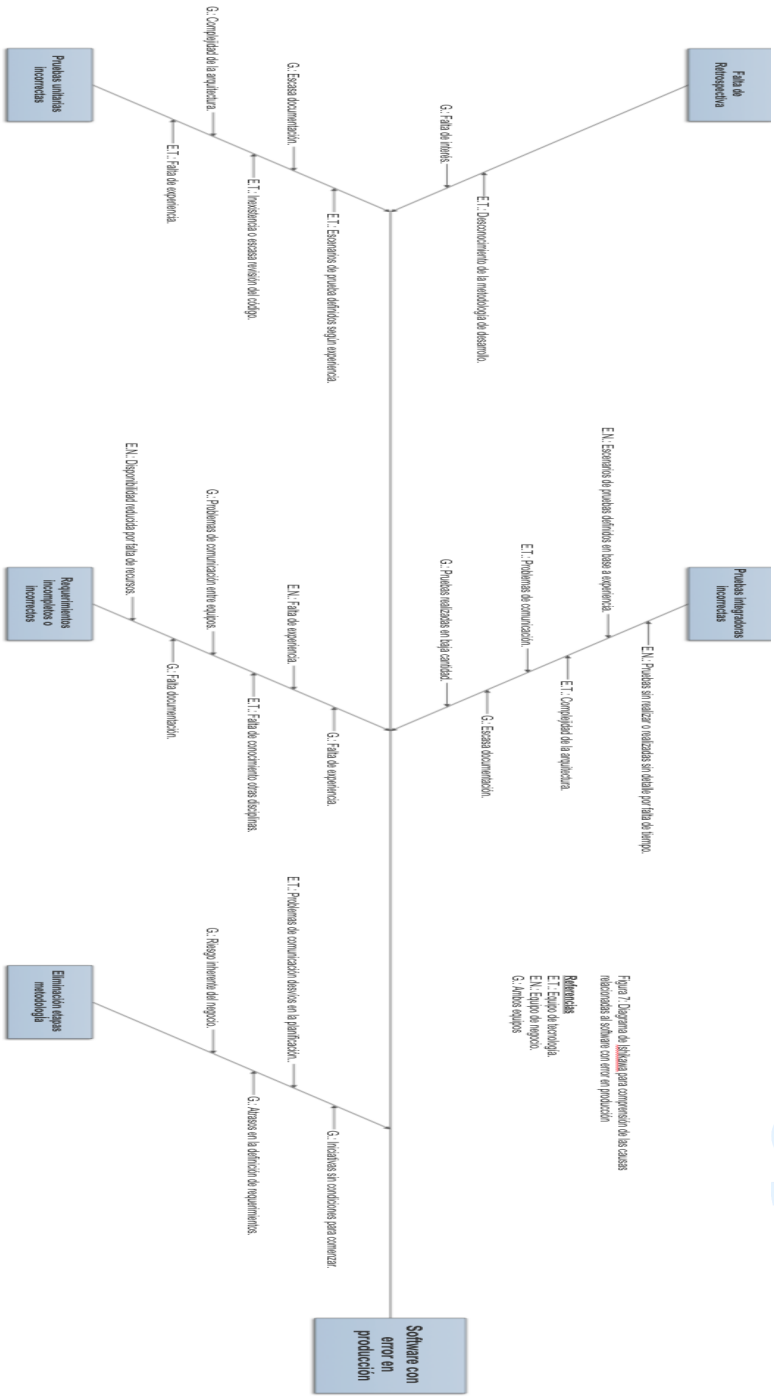

Figura 22: Diagrama de Ikishawa para causas identificadas en la investigación forence relacionado al error en producción. Fuente: Elaboración propia.

5. Conclusiones

Para comenzar a definir la conclusión a la que se aborda con este trabajo de investigación, es importante volver a analizar las preguntas que fueron el disparador del trabajo y analizar qué conceptos se encuentran interrelacionados y pueden llegar a ser la respuesta de las mismas:

1. ¿Son los errores en los productos finales del software, generados por la falta de conocimiento o incorrecta implementación de la metodología de desarrollo?
2. ¿Existieron falencias en la comunicación entre los miembros del equipo que desarrollaron el software, y la comunicación con otras área del negocio que favorecen la ocurrencia de errores en los productos finales de software?.

Ambas preguntas de investigación son parcialmente correctas. Esto es debido a que no es posible asegurar que los errores en los productos finales del software hayan sido generados únicamente por una incorrecta implementación de la metodología de desarrollo o por falencias en la comunicación entre los miembros del equipo que desarrollan el software y las áreas de negocio.

La respuesta a ambas preguntas según esta investigación, es que no es posible que los productos finales de software cuenten con errores relacionados con una incorrecta implementación de la metodología de desarrollo y falencias en la comunicación dentro y fuera del equipo de desarrollo. En otras palabras, los errores en producción ocurren cuando se implementó incorrectamente la metodología del software y existieron problemas de comunicación entre el equipo de desarrollo y otras áreas.

Con el fin de ahondar en detalle de esta respuesta, se definieron 3 conceptos (que unen una incorrecta implementación del desarrollo del software y falencias en la comunicación), que son el origen de los errores en código productivo. Asimismo, es importante destacar que existe una dependencia entre los mismos, ya que sin que ocurra alguno de los 3 conceptos no es posible afirmar que puedan seguir existiendo software con errores en el ambiente productivo.

Por último cabe destacar que no es posible asegurar que este descubrimiento sea aplicable a todas las empresas de tecnología dentro o fuera de la cultura argentina. La realización de futuras investigaciones sobre el tema desarrollado en distintas compañías tecnológicas de América Latina puede arrojar conclusiones similares y por lo tanto, identificar una correlación de hechos que permita analizar las conclusiones como factores externos a la compañía en términos de cultura y educación.

La estructura utilizada para detallar los conceptos, es la siguiente:

- Etapa de metodología afectada: Se identifica que etapas del desarrollo del software se ven afectadas.
- Falencias de comunicación: Se identifica que equipos cuentan con problemas de comunicación.
- Descripción del hallazgo: Se identifica cuáles son los causales por los cuales ocurre.

<u>Recursos insuficientes del equipo de negocios</u>

Etapa de la metodología afectada: Defunción funcional y requerimientos técnicos, pruebas integrales.

Falencias de comunicación: Equipo de negocios.

Descripción del hallazgo: A lo largo de la sección del análisis de resultados de los instrumentos utilizados, se observa una falencia relacionada a la asignación de los recursos de negocio para la definición de las características que debe tener el software.

Los recursos de los equipos de negocios, cuentan con una planificación ajustada debido a que tienen que realizar tareas operativas propias de su sector y, adicionalmente realizar tareas de consultor para la identificación de los requerimientos del software a desarrollar.

Esta planificación ajustada del equipo de negocios genera los siguientes inconvenientes:

1. Los recursos ideales con experiencia para definir los requerimientos del software a desarrollar no siempre son asignados a los proyectos debido a la falta de disponibilidad en su planificación. Por lo que ocasionalmente, son asignados recursos con menos experiencia a la definición de los requerimientos, pudiendo generar requerimientos incorrectos o incompletos.
2. Dado los tiempos reducidos que el equipo de negocio cuenta para atender las necesidades de los desarrolladores, al momento de realizar las pruebas

puede ocurrir la posibilidad de que no se contemplen todos los escenarios, y por lo tanto se introduzca software a producción sin ser probado.

3. La incorrecta defunción de los requerimientos del software, y el escaso tiempo planificado para desarrollar esta tarea puede ocasionar que se definan incorrectamente los requerimientos del software o que no se cuenten con todas las condiciones para comenzar un proyecto y de este modo retrasar el proyecto, que sumado al riesgo inherente del negocio en él se encuentra la empresa, puede ocasionar que se introduzca software en el ambiente productivo sin ser probado debido a tiempos de salida del software por motivos de la competencia.

Equipo de desarrollo y conocimiento en la arquitectura Compañía de Tecnología S.R.L

Etapa de la metodología afectada: Defunción funcional y requerimientos técnicos, pruebas unitarias, pruebas integrales.

Falencias de comunicación: Equipo de tecnología y negocios.

Descripción del hallazgo: El conocimiento de la arquitectura de Compañía de Tecnología S.R.L por parte de los equipos de desarrollo se ven afectadas por diversos factores que influyen en la identificación de requerimientos, los cuales son definidos incompletamente o incorrectamente al momento de la definición funcional y requerimientos técnicos, y durante las etapas de pruebas unitarias y pruebas integrales,

impidiendo probar todos los escenarios involucrados para asegurar la correcta integración y funcionamiento del software en las integraciones con otros sistemas.

La falta del conocimiento en la arquitectura de Compañía de Tecnología S.R.L por parte de los desarrolladores se ve afectada por:

1. La existencia de subculturas dentro de cada equipo de desarrollo se genera ya que los líderes de los equipos aseguran que si no dan una flexibilidad adicional a los desarrolladores, los mismos podrían irse a trabajar a otra empresa. Esta flexibilidad genera las subculturas impidiendo la rotación del personal hacia otros equipos de desarrollo y por lo tanto, no es posible que adquieran conocimiento sobre la arquitectura completa de Compañía de Tecnología S.R.L más allá de la que actualmente utilizan y conocen. Las subculturas de los equipos de desarrollo funcionan como burbujas dentro de la cultura de Compañía de Tecnología S.R.L dado a que cuentan con sus propios beneficios por sobre los definidos a nivel compañía generando una barrera de salida hacia otros sectores por la posible pérdida de beneficios actuales.
2. Dado la demanda actual de desarrolladores de software que experimenta el mercado, la mayoría de los ingresos en la compañía corresponden a desarrolladores analistas o semi-seniors, por lo que al asignarlos a un proyecto para el desarrollo del software puede ocasionar desvíos en la planificación debido a su falta de experiencia y comunicación

(característicos de su puesto) y por lo tanto retrasar el desarrollo del software.

3. Dado la demanda actual de desarrolladores, no es un requisito que los mismos se encuentren recibidos de un terciario o carrera de grado por lo que gran cantidad de los desarrolladores, independientemente de su seniority, no se encuentran graduados y por lo tanto desconocen de otras disciplinas por fuera de la tecnológica como la contabilidad.
4. Por la falta de conocimiento sobre la arquitectura y el bajo seniority que disponen los desarrolladores de software en la empresa, existen falencias al momento de realizar las pruebas unitarias e integradoras ya que debido al volumen de datos y sistemas que utiliza Compañía de Tecnología S.R.L, las pruebas son realizadas en base a la experiencia (calidad) y no cantidad, por lo que la omisión de un escenario de prueba al momento de definirlos puede ocasionar que se introduzca software con error en el ambiente productivo.

Inexistencia de un análisis de retrospectiva

Etapa de la metodología afectada: Retrospectiva

Falencias de comunicación: Equipo de tecnología.

Descripción del hallazgo: A medida que se realizaba la observación pasiva junto al equipo de Desarrollo IT – Cargos, se observó que durante el desarrollo de la reuniones

de equipo se interrumpía el desarrollo normal de la misma para dar lugar a discutir sobre errores que habían ocurrido con otros desarrollos.

Estos errores tenían similares características, comportamiento y tratamiento. Esto llama poderosamente la atención ya que existía una etapa dentro del desarrollo del software que su función era la de analizar cómo fue el desarrollo del software y en caso de que existiese un error, analizar los mismos con el fin de aprender y corregir lo necesario con el fin de que no vuelva a ocurrir. No se observó que el equipo de desarrollo IT de Cargos haya realizado durante la etapa en la que se desarrolló la observación pasiva un análisis de retrospectiva por los desarrollos realizados. Asimismo, esta información se corroboró con las entrevista del equipo de IT-SAP donde tanto el líder como los analistas confirmaron que no se realizaba una análisis de retrospectiva.

5.1 Implicancias y recomendaciones de acción

La introducción de un rol dentro del equipo de negocios como “Integrador”, permitiría la flexibilidad y atención que el ciclo del desarrollo del software necesita.

Este rol debería mediar entre el equipo de desarrollo del software y el equipo de negocios, y debería estar focalizado en la definición de requerimientos y escenarios de pruebas. Dicha posición debería ser ubicarse dentro del equipo de negocios con el fin de que adquiera el conocimiento del negocio y debería disponer de media jornada laboral para la realización de tareas operativas del área y el tiempo restante en la participación de proyectos para el desarrollo del software. Se recomienda que dicho rol sea ocupado por un Ingeniero en Sistemas o Ingeniero Industrial.

Asegurar que todos los equipos de desarrollo cuenten con los mismos beneficios y no se permita una flexibilidad en los mismos más allá de los acordadas dentro de la compañía puede favorecer la rotación entre equipos de trabajo, y de este modo aumentar el conocimiento de los desarrolladores.

La capacitación obligatoria en otras disciplinas como la contable, o legal para aquellos desarrolladores que no cuenten con carreras de grado, podría enriquecer la calidad de los desarrollos de software e identificar requerimientos de software que sean incorrectos o incompletos.

La realización de la etapa de retrospectiva no sólo identifica las falencias y permite corregir conceptos teóricos que cuentan los desarrolladores del equipo sino que también, incrementa la productividad del equipo en el proyecto, la calidad de la solución entregada, y potenciar el aprendizaje del equipo de manera sistemática, iteración a iteración, con resultados a corto plazo. Por último, aumenta la motivación del equipo dado que participa en la mejora del proceso, se siente escuchado, se toma decisiones consensuadas para eliminar las barreras que impiden que el equipo sea más productivo.

Por lo que la implementación de una reunión de Retrospectiva, bajo un facilitador que disponga la autoridad, mecanismos y recursos, permitirá a los equipos de desarrollo solucionar los obstáculos que se presentan en el momento de desarrollo del software y favorecer un aprendizaje continuo cuyo resultado será un software de mayor calidad y libre de errores.

6. Bibliografía

Agile Manifiesto - http://agilemanifesto.org. Octubre 2001. Web. 23 Febrero 2018.

Ahmedshareef, Z., Hughes, R., & Petridis, M. (2014). Exposing the influencing factors on software project delay with actor-network theory. *Electronic Journal of Business Research Methods*, *12*(2), 132–146.

Alexander, P., & Silvis, E. (2014). Actor-network theory in information systems research. *Information Research-an International Electronic Journal*, *19*(2). Retrieved from http://repository.up.ac.za/bitstream/handle/2263/40976/Alexander_Actor_2014.pdf?sequence=3

Bender, R. (2012). Systems Development Life Cycle : Objectives and Requirements. *Bender RBT Inc.*, 5–29. Retrieved from http://www.benderrbt.com/Bender-SDLC.pdf

Callon, M. (1986). Power, action and belief: a new sociology of knowledge? In *Some elements of a sociology of translation: domestication of the scallops and the fishermen of St Brieuc Bay* (p. 196–223–TS–EndNote Tagged Import Format).

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Dyba, T., & Dingsoyr, T. (2009). What Do We Know about Agile Software Development? *IEEE Software*, *26*(5), 6–9. https://doi.org/10.1109/MS.2009.145

Fujimura, J. H., & Latour, B. (1989). Science in Action: How to Follow Scientists and

Engineers through Society. *Contemporary Sociology*. https://doi.org/10.2307/2073372

Henderson-Sellers, B., & Serour, M. K. (2005). Creating a Dual-Agility Method: The Value of Method Engineering. *Journal of Database Management*, *16*(4), 1–23. https://doi.org/10.4018/jdm.2005100101

Highsmith J. (2002). What is agile software development? *Crosstalk*, *Highsmith*(Highsmith J. 2002. What is agile software development? Crosstalk, Highsmith, 4–9.), 4–9.

IBM Corporation. (2007). *IBM Rational Unified Process: Best Practices for Software development Teams*. *Version* (Vol. 7). Retrieved from http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Latour, B. (1996). Social theory and the study of computerized work sites. *Information Technology and Changes in Organizational Work*, 295–307.

Law, J. (1992). Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity. *Systems Practice*, *5*(4), 379–393. https://doi.org/10.1007/BF01059830

Law, J. (2009). Actor network theory and material semiotics. In *The New Blackwell Companion to Social Theory* (pp. 141–158). https://doi.org/10.1002/9781444304992.ch7

Lee, G., & Xia, W. (2010). TOWARD AGILE: AN INTEGRATED ANALYSIS OF QUANTITATIVE AND QUALITATIVE FIELD DATA ON SOFTWARE DEVELOPMENT AGILITY. *MIS Quarterly*, *34*(1), 87–114. https://doi.org/Article

Lyman, P., Latour, B., & Porter, C. (1997). Aramis, or, The Love of Technology. *Contemporary Sociology*, *26*(1), 90. https://doi.org/10.2307/2076625

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, *48*(5), 72–78. https://doi.org/10.1145/1060710.1060712

Nikiforova, O., Nikulsins, V., & Sukovskis, U. (2009). Integration of MDA framework into the model of traditional software development. In *Frontiers in Artificial Intelligence and Applications* (Vol. 187, pp. 229–239). https://doi.org/10.3233/978-1-58603-939-4-229

Pressman, R. S. (2010). *Software Engineering A Practioner's Approach*. *McGraw-Hill* (Vol. 33). https://doi.org/10.1109/6.476732

Rubin, K. S. (2012). Scrum Framework. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*, 13–28. Retrieved from http://www.amazon.com/Essential-Scrum-Practical-Addison-Wesley-Signature/dp/0137043295/ref=sr_1_1?s=books&ie=UTF8&qid=1397628062&sr=1-1&keywords=Essential+Scrum+A+Practical+Guide+to+the+Most+Popular+Agile+Process

Takeuchi, H., & Nonaka, I. (1986). The New New Product Development Game. *Harvard Business Review*, *64*(1), 137–146. https://doi.org/10.1016/0737-6782(86)90053-6

7. Anexos

Reuniones realizadas entre el equipo de Cargos

Las reuniones de seguimiento de tareas, estado de los desarrollos y nuevos desarrollos a implementar realizadas por el equipo de Cargos tienen una periodicidad semanal con un horario fijo.

Durante el tiempo observado ninguna de las reuniones contó con todos los miembros del equipo de manera presencial. Usualmente algunos se conectaban desde sus casas debido a que habían decidido no ir a la oficina, otros se encontraban en otro edificio realizando reuniones y otros, estaban con cursos de capacitación de la compañía y no era posible asistir.

La reunión siempre era precedida por Leandro, Líder del Equipo de Cargos, y para los casos en los que Diego se encontraba ausente, la misma era realizada por Victoria Alanis, Desarrolladora Senior del Equipo de Cargos.

Las reuniones de semanales estaban estructuradas por línea de negocio debido a que dependiendo el tipo de transacción que el usuario realizaba en el sitio de Compañía de Tecnología S.R.L se iba a generar un cargo para en el proceso de Créditos Especiales, Envíos Especiales, Marketplace de Compañía de Tecnología o Pagos Especiales.

Cada línea de negocio a las que se generaba un cargo estaba a cargo de uno o dos desarrolladores donde tenían las tareas para realizar, quienes durante la reunión comentaban en qué estado se encontraban las tareas que tenían planificadas para la semana, y si tuvo alguna complicación con el desarrollo de las mismas.

Los comentarios sobre las tareas realizadas, y pendientes eran registrados por el moderador de la reunión en un Excel informal al cual nadie tenía acceso.

Cada vez que existiese un nuevo desarrollo a realizar, luego la exposición del grado de avance y tareas pendientes, el desarrollador planteaba la arquitectura a nivel técnico que iba a implementar como solución. La misma era discutida entre todos los miembros de la reunión hasta que se hayan despejado todas las dudas y se haya seleccionado la mejor arquitectura. Para los casos en los que el nuevo desarrollo incluya a más de un equipo de trabajo, la arquitectura era definida por un gerente de desarrollo, y todos debían realizar sus desarrollos en base a la arquitectura planteada.

Durante la reunión se realizan interrupciones en el seguimiento de la planificación para exponer la ocurrencia de errores en los desarrollos, estos están dados en su mayoría por:

- Inconsistencias en la base de datos con respecto a registros que no fueron generados correctamente.
- Registros que no fueron absorbidos por un proceso de la compañía y por lo tanto no fueron registrados en la contabilidad.
- Fallas en la ejecución del código desarrollado lo que genera que procesos automáticos de facturación no operen correctamente.

Se observó que el equipo de desarrollo realiza herramientas

Agregar que ninguno de los miembros parece tener conocimiento en scrum salvo por la experiencia.