INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

MAESTRÍA EN INFORMÁTICA APLICADA

Reconocimiento de Validez Oficial de Estudios de Nivel Superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de Noviembre de 1976.

IMPLEMENTACIÓN DE PROYECTO PARA HABILITACIÓN DE ADQUISICIONES EN UNA EMPRESA DEDICADA A TECNOLOGÍA INFORMÁTICA SIGUIENDO LA METODOLOGÍA ÁGIL

Desarrollo de estudio de caso Pro-grado que para obtener el grado de

Maestro en Informática Aplicada

Presenta:

Juan Rodrigo Arroyo Ochoa

Asesor:

Mtro. Ricardo Salas Mejía

Guadalajara, Jalisco 10 de Junio de 2017

DEDICATORIA

Para Carol, Rodrigo y los que están por venir… mi inspiración y motivación.

ÍNDICES

I. Contenidos.

II. Índice de tablas

III. Índice de figuras

RESUMEN

Comenzaré este resumen con una cita de (Fowler, 2005):

> Hay una frase que he escuchado en cada proyecto problemático en el que he participado. Los desarrolladores vienen conmigo y me dicen "el problema con este proyecto es que los requerimientos siempre están cambiando". La parte que encuentro sorprendente sobre esta situación es que alguien se sorprenda de esto. En la construcción de requerimientos de negocio para software, los cambios son la norma, la pregunta es qué es lo que hacemos con respecto a esto.

El presente documento relata la experiencia que tuve al participar como líder técnico en un proyecto donde nos encontramos con este caso de requerimientos cambiantes y el cómo la metodología Ágil nos ayudó a llevarlo a buen término.

Dicho proyecto apoya a la mejora y automatización de una parte del proceso de habilitación de adquisiciones (compra de otras empresas) en una empresa multinacional dedicada a la tecnología informática. Su proceso actual se realiza en varios pasos que son ejecutados por distintos grupos de personas donde la salida de uno es entrada de otro, algunos de estos pasos son manuales y otros dentro de algún sistema.

Es importante mencionar que esta área de negocio reconoce que tiene muchos pasos manuales que podrían automatizarse, así como partes en donde solo puede participar una persona y se tiene la necesidad de convertirlo en trabajo colaborativo. Se decidió dividir el proceso y atacar primero los pasos que podrían ser implementados de una forma rápida y que aportan más valor al proceso general.

El proyecto se implementó siguiendo una adaptación hecha por la misma compañía de *Scrum,* que es un proceso de desarrollo de software, parte de la metodología Ágil y que sigue ciertas prácticas que propician la colaboración en el equipo, el correcto seguimiento a la implementación y nos periten reaccionar de manera adecuada y a tiempo en caso de ser necesario.

El lector encontrará, además de los conceptos teóricos aplicables al caso, el cómo se desarrolló el proyecto desde la etapa de descubrimiento hasta su cierre, pasando por los problemas que se presentaron durante este tiempo y el cómo se resolvieron.

Por último, se incluyen las lecciones aprendidas, oportunidades de mejora identificadas por los mismos miembros del equipo implementador y, finalmente conclusiones personales sobre lo aprendido en el proyecto y las reflexiones que generó el esfuerzo de documentación para elaborar este estudio de caso.

Palabras clave: Desarrollo de Software, Metodologías Ágiles, Scrum.

CAPÍTULO I. MARCO DE REFERENCIA

1.1 Conceptos teóricos aplicables al proyecto

Agile

El término *Agile* nace en Febrero del 2001 cuando un grupo de personas se reunió para intercambiar ideas y obtener un entendimiento común sobre distintos enfoques de desarrollo de software. (Fowler, 2005)

El resultado de la reunión fue el conocido como manifiesto Ágil para desarrollo de software:

> Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:
>
> Individuos e interacciones sobre procesos y herramientas
>
> Software funcionando sobre documentación extensiva
>
> Colaboración con el cliente sobre negociación contractual
>
> Respuesta ante el cambio sobre seguir un plan
>
> Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.
> (Beck et al., 2001)

Además del manifiesto ágil, se inició el desarrollo de los principios ágiles que fueron completados más adelante y también forman parte de la base de la metodología, en conjunto con los valores y las prácticas, estos pueden ser encontrados en http://agilemanifesto.org/iso/es/principles.html.

Previo a estas reuniones, varios grupos habían desarrollado ideas similares, la mayoría de ellas enfocadas al desarrollo iterativo, (Kawaguchi, 2013) ilustra muy bien la relación y precedencia de los mismos, así como su relación con ágil. (Ver figura 1, Ágil & Lean)

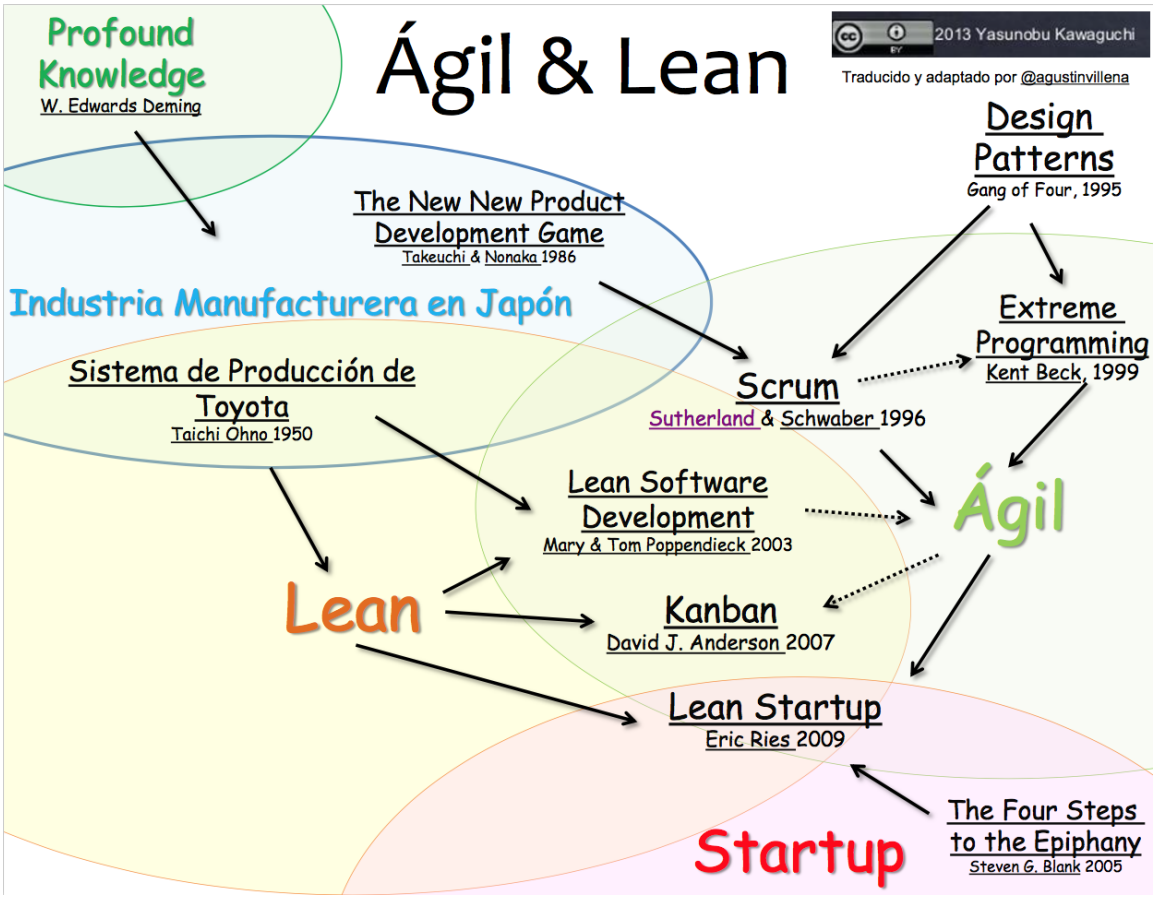

Figura 1: Ágil & *Lean*. (Kawaguchi, 2013)

Cómo nos indica Larman,

> Los métodos de desarrollo ágil aplican desarrollo iterativo y evolutivo, planificación adaptativa, promueven la entrega evolutiva e incluyen otros valores y prácticas que fomentan la agilidad: una respuesta rápida y flexible al cambio. No es posible definirlos exactamente, ya que las prácticas específicas varían, sin embargo, las iteraciones cortas con refinamiento adaptativo y evolutivo de planes y metas es una práctica básica que comparten entre ellos.
>
> (Larman, 2003)

Lo que logran estas personas que se reunieron en 2001 es muy importante, sientan las bases para unificar esfuerzos realizados por diversos grupos que ya tenían una ideología similar, construyendo un gran conjunto de principios, valores y prácticas basados en el manifiesto Ágil del cual cada equipo que lo utiliza es libre de tomar lo que le acomoda. La empresa en donde laboro resumió los principios y valores ágiles, tomando los que se adaptan a la cultura organizacional y se apoya de las prácticas para permearlos en la corporación.

Scrum

Es un *framework* o marco de referencia de la metodología ágil diseñado para proyectos complejos o con requerimientos cambiantes,

> Parte de la premisa de que vivimos en un mundo complicado, no podemos predecir qué entregaremos, cuando lo entregaremos ni cuál será la calidad y costo, sin embargo, podemos vincular el proceso empírico con criterios de monitoreo y gestionar el mismo con mecanismos de retroalimentación constantes.
>
> (Highsmith, 2002)

Dicha premisa hace mucho sentido si hemos tenido experiencias en proyectos en los que los requerimientos no están claros en la etapa de análisis y aun así se nos pide proveer un presupuesto para su implementación. *Scrum* basa sus prácticas en el siguiente proceso que es iterativo e incremental.

Figura 2: *Scrum framework.* (ScrumAlliance, 2016)

Según (Schwaber, 2004), existen solo 3 roles en Scrum, el *Product owner*, el equipo y el *ScrumMaster*, los cuales se dividen las tareas de administración del proyecto:

- El *product owner* representa los intereses de todos los participantes en el proyecto, se encarga de obtener el financiamiento inicial y continuo del proyecto, creando y manteniendo priorizada la lista de requerimientos, planeando los entregables y los objetivos de retorno de inversión.
- El equipo es auto-dirigido, auto-organizado e inter-funcional, se encarga de implementar funcionalidad incrementalmente basado en las prioridades del *product backlog*, el equipo es responsable del éxito de cada iteración y finalmente del proyecto completo.
- El *ScrumMaster* se encarga de asegurar que se siga el proceso de Scrum, de que este sea conocido por el equipo y de que entregue valor a la organización y al cliente.

La página de *Scrum Alliance* nos explica este proceso, en los siguientes pasos:

- Un *product owner* crea una lista priorizada de deseos llamada *product backlog*.
- Durante la planeación de la iteración, el equipo toma una parte pequeña de la parte de arriba de la lista de deseos, un *sprint backlog*, y decide cómo implementar esas piezas.
- El equipo tiene cierta cantidad de tiempo — una iteración (usualmente dos a cuatro semanas) — para completar su trabajo, pero se reúne cada día para revisar su progreso (*Scrum* diario).
- Durante el camino, el *ScrumMaster* mantiene al equipo enfocado en su meta.
- Al final de la iteración, el trabajo debería ser potencialmente entregable: listo para entregar al cliente, ponerlo en un estante de tienda o mostrarlo a un *stakeholder*.
- La iteración termina con una revisión de la misma y retrospectiva.
- Cuando la siguiente iteración comienza, el equipo elige otro pedazo del *product backlog* y empieza a trabajar de nuevo.

(ScrumAlliance, 2016)

La organización en la que laboro realizó una adaptación de *Scrum* en la que agrega un par de roles y cambia el nombre del rol de *ScrumMaster* por *Iteration manager*, al final es solo un cambio de nombre y sus responsabilidades son las mismas.

Los roles que agrega a la metodología son llamados miembros del equipo extendido y se componen por:

SME o Subject matter expert, son expertos en cierta área de negocio o técnica que pueden ser agregados a reuniones conforme sean necesarios.

Project manager, es un rol que aplica cuando el proyecto particular es parte de un programa o proyecto más grande, en este caso el Project manager es el facilitador a dicho nivel, hace las veces de *Iteration manager* pero con un alcance de varios sub-proyectos.

Sponsor, es la persona o grupo de personas que provee los recursos económicos y de otros tipos para la realización del proyecto.

Prácticas ágiles

Las prácticas son mecanismos que, mediante su ejecución, nos ayudan a permear los principios y valores de Ágil en el equipo y eventualmente en la cultura organizacional, ya que proporcionan un ambiente de colaboración y respeto a la vez que ayudan a no perder de vista lo más importante en un proyecto que es dar verdadero valor al cliente.

El principal valor de estas prácticas es, brindarnos la capacidad para responder al cambio además de garantizar que las historias de usuario contienen el detalle necesario y los criterios de aceptación en tiempo y forma. Algunas de las prácticas principales utilizadas en el proyecto reportado son:

User stories, o historias de usuario son elementos que nos ayudan a expresar un problema o requerimiento enfocado al usuario, comúnmente inicia como una idea simple que se agrega al *Backlog* y posteriormente evoluciona con mayor detalle a través del *Backlog refinement, Iteration planning* y de la misma implementación. Se busca comúnmente que una historia de usuario pueda ser implementada por completo en una sola iteración, si se encuentra que su complejidad es mayor, lo mejor es tratar de dividirla. (IBM Corp., 2016)

Una historia de usuario tiene una descripción inicial que consta de la siguiente estructura:

Como un _____ quiero _____ para _____

En donde se identifica el tipo de usuario al cual está enfocada la historia, la funcionalidad requerida y el objetivo de la misma.

Acceptance criteria, es el set de requerimientos específicos relacionados a una historia de usuario que deben ser completados o implementados para poder considerar dicha historia de usuario como completada. (IBM Corp., 2016)

Son los puntos clave en los que un revisor se puede basar para comprobar que el requerimiento de una historia de usuario fue correcto satisfactoriamente. Esto aporta mucho valor a una historia de usuario, ya que evita ambigüedad.

Backlog refinement, consiste en reuniones periódicas en las que parte del equipo (*Iteration manager*, *product owner* y probablemente algún representante del área técnica) se enfoca en analizar, documentar y priorizar la lista de historias de usuario que se encuentran en el *backlog*. Durante esta reunión, pueden agregarse nuevas historias, eliminar algunas que ya no son necesarias y cambiar prioridad de las existentes, las historias con más prioridad se colocan arriba de la lista y deben tener suficiente detalle para que los miembros del equipo encargados de implementar puedan tomarlas y estimarlas. (*IBM Corp.*, 2016)

Story points, es una unidad de medida relativa a tamaño que se utiliza para estimar historias de usuario, cada equipo determina el valor de cierto número con base a la percepción del mismo equipo sobre el esfuerzo que tomaría implementarla. El número de *Story points* que un equipo puede implementar en una iteración se convierte en la velocidad del equipo y es utilizada como entrada para cada *Iteration planning*.

Planning poker, es una técnica de estimación en la que participa todo el equipo y nos ayuda a asignar *Story points* a una historia de usuario. El equipo primero debe definir la escala de números a utilizar, comúnmente se utiliza la serie Fibonacci, aunque también se pueden utilizar valores como tallas de camiseta (XS, S, M, L, XL).

Durante el ejercicio, se revisan los detalles de las historias de usuario a estimar y cada participante elige la complejidad de la misma de manera secreta, una vez que todos tienen un voto, estos se muestran a todo el equipo y se busca llegar a un acuerdo sobre dicha complejidad, normalmente las personas que eligieron el valor más alto y el más bajo argumentan su decisión y después de esto se puede ejecutar otra ronda de votación hasta llegar a un común acuerdo.

Es probable que, durante este ejercicio, se encuentre que alguna historia de usuario no tiene el detalle suficiente para realizar la estimación, en este caso, se solicita el detalle faltante, ya sea en descripción o criterios de aceptación al *product owner* y se continúa con las historias que sí tienen el detalle requerido. (IBM Corp., 2016)

Iteration planning, en esta reunión, el equipo toma los elementos que se encuentran en la parte superior del *Product backlog* (previamente estimados) y, basados en la experiencia y velocidad de iteraciones pasadas y en posibles restricciones como la disponibilidad de miembros del equipo, planea cuántas y cuáles historias de usuario sería posible implementar en la siguiente iteración.

El *Iteration manager* planea la reunión (por lo regular de 4 horas) y los pre-requisitos necesarios para realizarla efectivamente son, que las historias de usuario en el *Product backlog* estén correctamente

priorizadas y contengan suficiente detalle para estimarlas en caso de que no se encuentren ya estimadas. Comúnmente participan los integrantes del equipo encargados de la implementación.

Los elementos del *Product backlog* que son seleccionados para ser implementados en la iteración o *Sprint* que se está planeando, pasan a formar parte del *Sprint backlog*. (IBM Corp., 2016)

Stand-up, es una reunión que se lleva a cabo diariamente en la que participa todo el equipo, su principal objetivo es mantener al equipo enterado del estado actual del proyecto en qué está trabajando cada quién y si existe algún impedimento que deba ser atacado.

Normalmente se lleva a cabo en máximo 15 minutos y los participantes comentan qué es lo que hicieron el día anterior, cuál es el plan para el día actual y si hay algún impedimento para realizar su trabajo. Gerentes y otros *stakeholders* pueden asistir, pero deberán abstenerse de participar activamente.

Si durante la ejecución de la reunión, surgen temas a ser revisados con mayor detalle, se puede utilizar el tiempo después de la reunión o agendar una reunión por separado solo con las personas involucradas. (IBM Corp., 2016)

Wall of work, esta práctica consiste en tener un lugar en donde el estatus actual de las actividades a ser realizadas es visible para todos en el equipo.

El equipo se pone de acuerdo en cuáles son las columnas importantes a agregar al muro, estas deberán proporcionar información sobre el estado de cada uno de los elementos en el mismo y puede ser algo tan simple como: 'Pendiente', 'En progreso', 'En pruebas', 'Terminado'. (IBM Corp., 2016)

La persona ejecutando alguna actividad es la responsable de mover los elementos de una columna a otra y esta actividad se realiza común mente antes del *Stand-up* diario o durante esta reunión. (IBM Corp., 2016)

Showcase, también llamado *Playback* o *Demo*, es una reunión que nos da la oportunidad de obtener retroalimentación de los stakeholders acerca de las piezas de trabajo completadas hasta el momento. Cualquier persona interesada en el progreso del proyecto está invitada.

La preparación de esta reunión no debería tomar más de una hora ya que se basa en mostrar elementos ya implementados.

Normalmente la retroalimentación obtenida en esta reunión terminará reflejada en cambios a los elementos ya existentes en el Product backlog o generando nuevos elementos. (IBM Corp., 2016)

Work in progress, esta práctica se basa en la premisa de que es más eficiente trabajar en una sola tarea a la vez que hacer *Multi-tasking*, debido a esto se ponen límites para el número de elementos que una persona puede tener 'En progreso' en el *Wall of work*. Este límite puede ser ajustado al paso de las iteraciones basado en los resultados obtenidos. (IBM Corp., 2016)

Retrospectiva, es una reunión que ocurre por lo regular al finalizar una iteración, su objetivo final es el fomentar la mejora continua mediante la retroalimentación de todos los miembros del equipo.

La reunión inicia con 10 minutos en los que los integrantes del equipo, de manera individual, escriben lo que se piensan que realizó correctamente durante la iteración y que sería bueno seguir haciendo de la misma manera, qué podría mejorarse o debería dejar de ocurrir y también sugerencias de mejora.

Posteriormente, se agrupan y analizan los comentarios del equipo y opcionalmente se someten a votación para ver cuáles son los más relevantes y poder así tomar acciones sobre los mismos. Estas acciones pueden terminar incluso como elementos en el *Product backlog* del proyecto. (IBM Corp., 2016)

Requerimientos SMART

La palabra SMART es un mnemónico usado comúnmente en desarrollo de software para nombrar las características que un buen requerimiento debe tener, aun cuando es un término común, podemos encontrar diferentes interpretaciones, tomo la que me parece más entendible a continuación:

> ***S**pecific* (Específico): Debemos saber lo que se nos está pidiendo específicamente.
> ***M**easurable* (Medible): Debemos poder medirlo para poder monitorear el avance.
> ***A**ttainable* (Alcanzable): Debe ser algo técnicamente posible.
> ***R**elealisable* (Realizable): Debe ser lograble y realista en términos de tiempo, alcance y otras restricciones del proyecto.
> ***T**raceable* (Trazable): Debemos poder dar seguimiento a un requerimiento desde su concepción hasta su implementación.
> (IBM Corp., 2016)

Bases de datos y Administradores de bases de datos

Una base de datos es un software que se utiliza para almacenar información con las que trabaja un sistema de manera estructurada, puede ser relacional o no relacional.

Las bases de datos relacionales se componen de esquemas que tienen una estructura de tablas que representan entidades sobre las que queremos guardar información, las tablas se componen a su vez de campos o atributos que pueden tener distintos tipos y tamaños según el dato que se desea almacenar, así, un campo en el que almacenaremos una fecha será distinto a uno donde almacenamos un número. A estas tablas se les definen relaciones entre ellas para representar el cómo interactúan las diversas entidades relativas al sistema del cual guardan información.

Para trabajar con los datos almacenados en una base de datos relacional utilizamos el lenguaje llamado *SQL*[1] mediante el cual se pueden realizar consultas, inserciones, etc.

Este tipo de bases de datos es ideal para los casos en que los datos a guardar son estructurados, con entidades y sus relaciones bien definidas, un ejemplo sería una base de datos para una escuela en donde podemos determinar fácilmente entidades como alumno, profesor, aula, asignatura, etc.

Por otro lado, las bases de datos no relacionales o también llamadas *NoSQL*, nos ayudan a resolver el problema que presentan las fuentes de datos no estructurados como las redes sociales, estas generan cantidades enormes de información diariamente y no necesariamente tienen estructuras definidas, aun así, es muy importante el poder almacenar esta información y poder aplicar operaciones a la misma para poder realizar análisis sobre ella.

Un administrador de bases de datos es un software que nos ayuda a crear bases de datos y proporciona una interfaz entre el usuario y la misma, existen muchos diferentes, por ejemplo, para bases de datos relacionales tenemos *MySQL, DB2, SQL Server, Oracle*, entre otros, del lado de los no relacionales tenemos *MongoDB, CouchDB, Cloudant, Cassandra*, etc.

¿Cómo elegir entre una base de datos relaciona o *NoSQL*?, las bases de datos relacionales son excelentes para aplicaciones centralizadas cuyos datos encajan en un modelo de datos relacional.

Un factor clave que empuja a las organizaciones a buscar alternativas a sus bases de datos relacionales actuales es la necesidad de consultar en grandes volúmenes de datos, hasta 2005 aproximadamente, los

[1] Structured Query Language

problemas de desempeño eran resueltos comprando procesadores más veloces, hasta que en cierto momento esto ya no fue una opción viable y se debió optar por usar varios procesadores trabajando juntos. Muchos DBMS relacionales son mono-procesador y no pueden responder a las demandas de insertar y consultar información en tiempo real que muchos sitios web tienen. (Dan McCreary, 2013)

Plataformas de Servicios en la nube y *Bluemix*

Una plataforma de servicios en la nube es un modelo de servicios en el que el cliente puede obtener el valor diversas partes de un sistema de software como almacenamiento, servidores, etc., sin la necesidad de ser el dueño de los mismos, accediendo a ellos a manera de renta por medio de una red, evitando así los costos de adquisición, mantenimiento y almacenamiento que tendría de otra forma. (IBM Corp., 2016)

Existen varios niveles, ilustrados en la figura 3, la primera columna representa un modelo tradicional en el que el cliente tiene y mantiene todas las partes, las columnas 2 a 4 representan los distintos niveles de servicios en la nube.

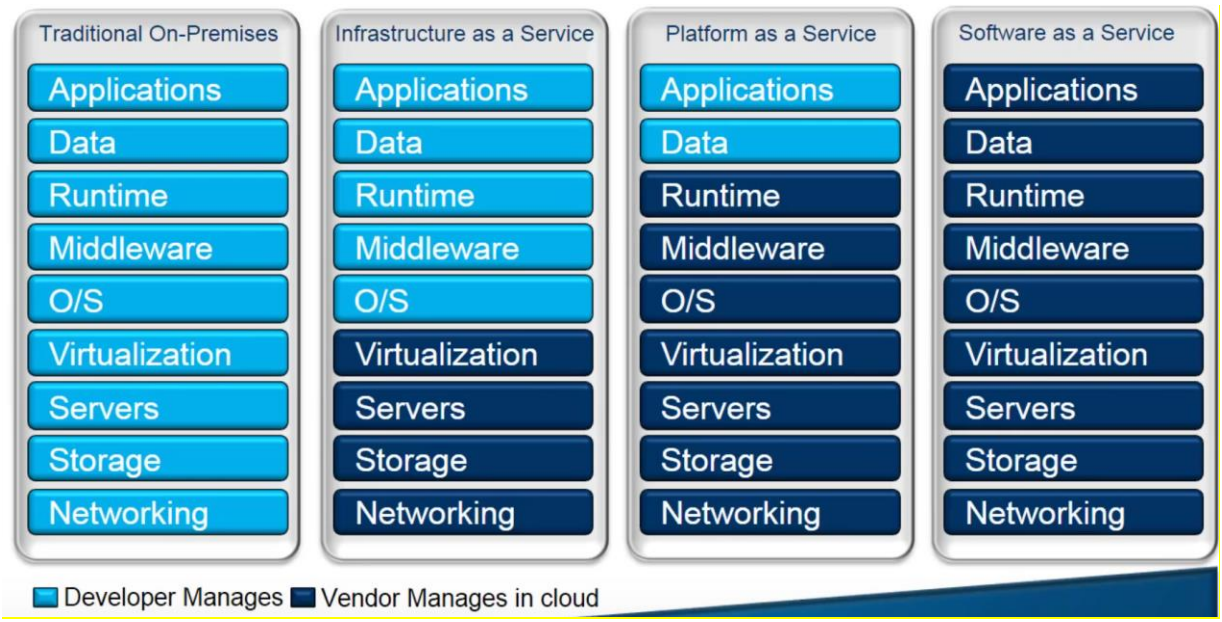

Figura 3: Servicios en la nube (IBM Corp., 2016)

Existen 3 proveedores principales de servicios en la nube, *Azure* de *Microsoft*, *Bluemix* de IBM y AWS de *Amazon*.

IBM Bluemix es una plataforma que combina *Infrastructure as a Service* (IaaS) y *Platform as a Service* (PaaS), adicionalmente tiene un amplio catálogo de servicios en la nube que pueden ser integrados con IaaS y PaaS para el desarrollo rápido de aplicaciones. (IBM Corp., 2017)

CAPÍTULO II. DESCRIPCIÓN DEL PROYECTO REPORTADO

2.1 Antecedentes del proyecto reportado

Compañía y áreas involucradas

El presente estudio de caso se centra en un proyecto ejecutado en el último cuarto del año 2016 en una compañía multinacional del ramo de la tecnología informática enfocada principalmente en fabricación y comercialización de hardware y software, así como servicios y consultoría en dicho ramo.

La empresa cuenta a la fecha con alrededor de 380,000 empleados y presencia en 175 países, trabaja con el 90% de los bancos más grandes del mundo, 9 de las 10 compañías petroleras más importantes, 40 de las 50 compañías líderes en venta al detalle y 92 de las 100 organizaciones de cuidado de la salud más importantes.

Una de las áreas de la empresa llamada *Transformation and Operations (T&O)* está enfocada en implementar cambios en la forma de trabajo interna de la compañía para alinearla a la estrategia de la misma, buscando reducir complejidad en los procesos y teniendo siempre en mente las necesidades del cliente final.

En otras palabras, *T&O* transforma continuamente la forma de trabajo de la compañía para convertir las ventajas competitivas en nuevas fuentes de creación de valor.

Una de las sub-áreas de *T&O* es la llamada *Enterprise services*, esta transforma y ejecuta procesos de soporte global para *stakeholders* internos y externos, proveedores, asociados de negocios y empleados. Esta a su vez tiene un área encargada de soporte a pre y post venta llamada *Sales transactions support* y debajo de ella se encuentra *SmarterProcess* que es la encargada de ejecutar el proyecto aquí documentado.

Por otro lado, el área para la cual se implementó la mejora por medio del proyecto reportado es denominada *'Systems'*, esta proporciona soluciones enfocadas a clientes externos, buscando ayudar a las empresas a responder a sus retos tecnológicos, su oferta está basada en tres pilares:

- Tecnología diseñada para una mejor gestión de los datos.
- Apuesta por entornos definidos por software que permitan la combinación de diferentes tecnologías y sean más flexibles, seguros y escalables.
- Tecnologías más abiertas y flexibles, que fomenten la colaboración y permitan adaptarse a los constantes cambios del mercado.

El área favorecida por el proyecto reportado es una de las sub-áreas de *Systems*, llamada *Acquisitions Integration* y se encarga de habilitar las soluciones y productos ofrecidos previamente por empresas recientemente adquiridas por la compañía y que seguirán comercializándose con las mismas o distintas condiciones.

Cabe señalar que es una parte importante del negocio, ya que cada adquisición busca incrementar el portafolio de productos y capacidad servicios de la compañía, tan solo en 2016, la compañía tuvo 56 adquisiciones con una inversión de casi 6 mil millones de dólares y la tendencia es similar para 2017.

Proceso de adquisición

Acquisitions Integration tiene un tiempo de ciclo estimado de 18 a 20 semanas para la habilitación de productos de una adquisición mediana (hasta 75 productos) y su proceso consiste en 6 pasos:

1. Categorización de producto (4 a 8 semanas), consiste en la creación de un inventario, identificación estructuras de producto y en la revisión de contratos con clientes y proveedores actuales.
2. Nombrado de producto (4 a 6 semanas), consiste en nombrado de producto, revisión de propiedad intelectual y derechos, así como ciertas aprobaciones.
3. Hoja de habilitación (ES) (4 semanas), Elaboración de una hoja de cálculo con características como tipo, métricas de cobro, cargos adicionales, de cada producto y sub-producto.
4. Códigos de parte (1 día), Obtención de números de parte, códigos y nombres legales.
5. Creación de partes (4 semanas), Creación y carga de registros en sistema.
6. Precios (2 semanas), Proceso mediante el cual se determina el precio unitario y los rangos de precios para cada producto.

Necesidad de negocio y beneficios esperados

En Agosto de 2016, personal ejecutivo del área *Acquisitions Integration* se puso en contacto con personal de *Enterprise services* para solicitar el análisis de su proceso, proponer mejoras y posibles automatizaciones al mismo, ya que están conscientes de que tiene muchos pasos con trabajo manual, repetitivo y propenso a errores.

Se realizó un *workshop* (detallado más adelante) en el cual se reunieron personas clave del proceso, analistas de negocio y personal técnico, encontrando las siguientes condiciones y necesidades:

- El tiempo de ciclo para la habilitación de productos o *Time to market* es muy largo.
- El proceso actual consta de muchos pasos, ejecutados por distintos equipos, con distintas necesidades, algunos de estos equipos no consideran que valga la pena la inversión en un sistema.
- El proceso de creación de la hoja de habilitación recae en una sola persona para obtener los datos necesarios y capturarlos en dicha hoja.

- No existe un repositorio o sistema central que ayude a compilar la información para alimentar la hoja de habilitación.
- Existen muchos pasos del proceso que se ejecutan manualmente y son propensos a errores.

Así mismo, se definieron los beneficios esperados de la implementación de un *Minimum valuable product* o *MVP*:

- Tener un repositorio central de información y convertir así el trabajo de una sola persona en trabajo colaborativo.
- Aumentar el control sobre el proceso mediante la habilitación de roles con diversos accesos.
- Reducir en 50% el tiempo necesario para la creación de la hoja de habilitación y el proceso de precios.
- Servir como un *showcase*, que, mediante la entrega de valor en corto tiempo, convenza a los dueños de otros pasos del proceso a mejorarlo.

Esto debido a la necesidad de obtener dichos beneficios en corto plazo (3 meses), después de esto se planea ir aumentándolos de manera iterativa con el desarrollo de futuras fases sin una fecha límite aún definida.

2.2 Objetivo del proyecto reportado

El objetivo directo del proyecto reportado es reducir en 50% el tiempo necesario para ejecutar dos de los pasos de proceso identificados, el primero, a ser reducido en 2 semanas es el crear la hoja de habilitación o *Enablement Spreadsheet* en la habilitación de productos, ofreciendo una plataforma en la cual más de una persona pueda colaborar para la creación de dicho archivo, el segundo es el proceso de agregar precios a los productos ya definidos, se busca reducir el tiempo de proceso en 1 semana para este último. Esto debido a que estos pasos del proceso de habilitación fueron elegidos como los primeros a mejorar.

El éxito del proyecto en términos de reducción de tiempo invertido en la creación de la hoja de habilitación y en agregar precios nos ayudará al logro de dos objetivos más importantes:

1. La percepción de valor en corto tiempo ayudará a convencer a algunas personas que dudan de la necesidad general de mejorar / automatizar el proceso.
2. También servirá como ejemplo de cómo la metodología ágil ayuda a responder al cambio, evitando conflictos y sensibilizando a todas las partes sobre la complejidad del negocio y la dificultad de responder a cambios constantes en el mismo.

2.3 Descripción de la metodología empleada

Se empleó una adaptación de la metodología *Agile* hecha por la organización y que en su fase de *delivery* sigue las etapas de *Scrum*, ya que la misma está haciendo un esfuerzo para que se siga en todas sus áreas, no solo en desarrollo de software. Este es uno de los primeros proyectos ejecutados en el área *SmarterProcess* con ella, por lo que implica un desafío, ya que la mayoría de los involucrados están acostumbrados a métodos tradicionales, así que buscaremos contribuir con el cambio de cultura organizacional que se requiere para su adopción exitosa.

Como se menciona en el marco teórico, *Agile* ayuda a los proyectos a corregir la dirección durante la ejecución, esto se logra con ciclos acotados que entregan valor al cliente y después de los cuales se revisan los posibles puntos de mejora.

Existen dos puntos que siempre deben estar presentes en un proyecto ágil; la búsqueda por mejorar y la apertura al cambio.

Principios y valores de *Agile*

La metodología se basa en muchos principios y valores, de los cuales la organización (IBM Corp., 2016) tomó los siguientes para su adaptación propia:

- Principios
  1. *Claridad del entregable*, indica que todo el equipo debe tener claro lo que se espera como entregable de su trabajo y tenerlo siempre en mente.
  2. *Iterar y aprender*, se trata de escuchar, iterar, aprender, corregir curso en vez de esperar al final esperando que el resultado sea perfecto.
  3. *Equipos auto-dirigidos*, construir equipos con las habilidades correctas e incentivar la innovación y el empoderamiento.
- Valores
  1. Confianza
  2. Empatía
  3. Apertura
  4. Respeto
  5. Coraje

Fases de la metodología

(IBM Corp., 2016) la divide en dos grandes fases, *Discovery* y *Delivery*. La primera se encarga de asegurar que haremos el trabajo correcto y la segunda de hacer correctamente el trabajo.

Durante la etapa de *Discovery*, se recomienda realizar un *Discovery brief* o *Workshop* en el que las personas clave colaboren entre sí para cumplir los siguientes objetivos:

- Lograr un entendimiento compartido
- Caso de negocio o beneficios esperados
- Explorar diferentes opciones de solución
- Asignar los recursos correctos
- Priorizar y administrar el trabajo

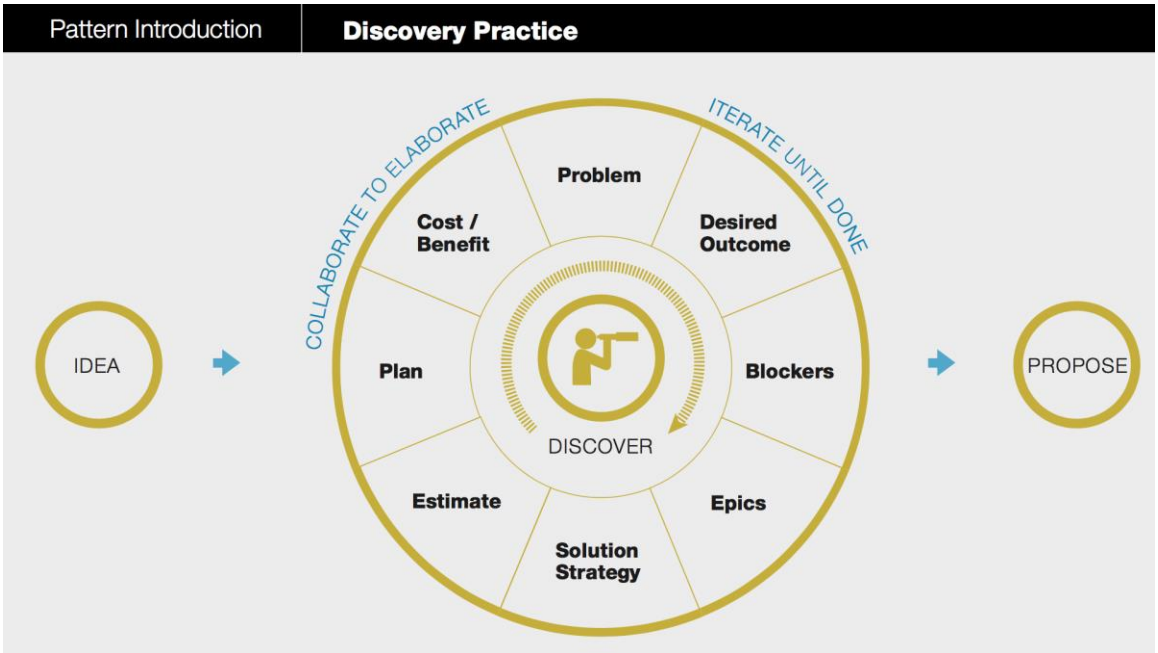

Figura 4: La práctica *discovery* (*IBM Corp*., 2015)

Es recomendable contar con los siguientes puntos al iniciar el *Discovery brief* o elaborarlos cuanto antes una vez iniciado:

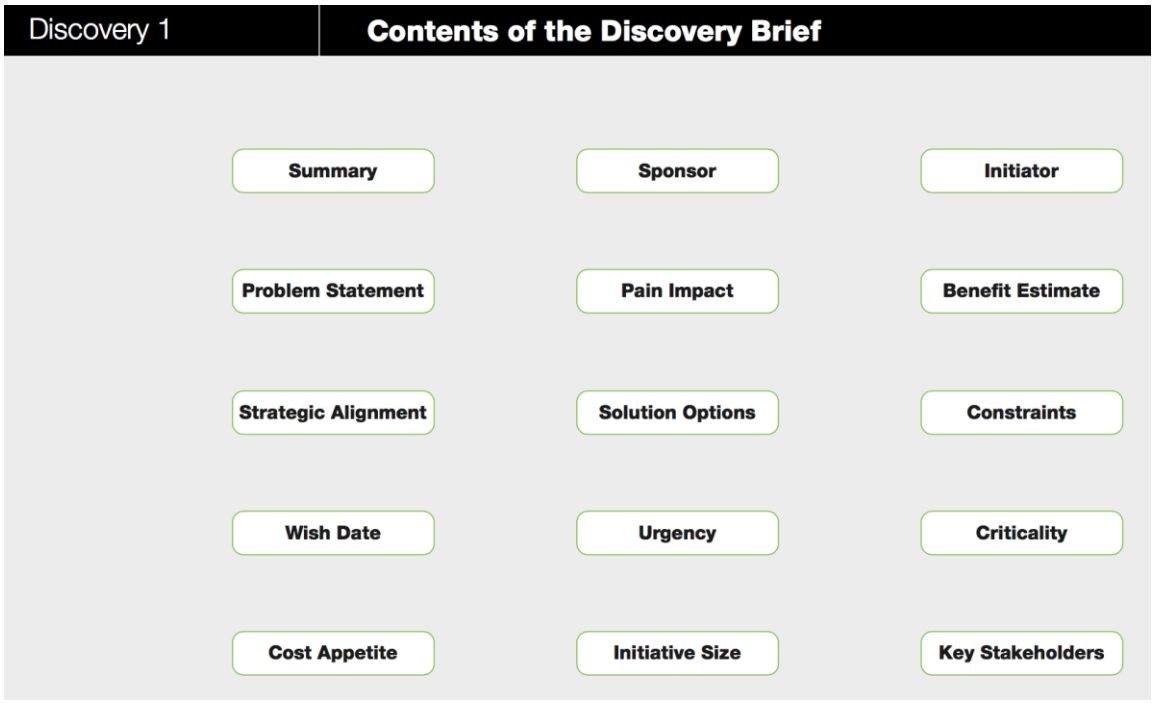

Figura 5: Contenidos del *Discovery brief (IBM Corp., 2015)*

Con este input, se ejecutan diversas prácticas para obtener información o elaborar sobre:

- Análisis del problema
- Análisis de *Stakeholders*
- Resultado deseado (Requerimientos SMART)
- Beneficios
- *Blockers*
- Alcance
- *Epics*[II], Historias de usuario, MVP
- Riesgos y dependencias
- Opciones de solución
- Estimación de alto nivel
- Planeación

[II] Conjunto de historias de usuario relacionadas entre sí

Una vez concluida la etapa de *Discovery*, inicia la etapa de *Delivery* con el siguiente ciclo de vida:

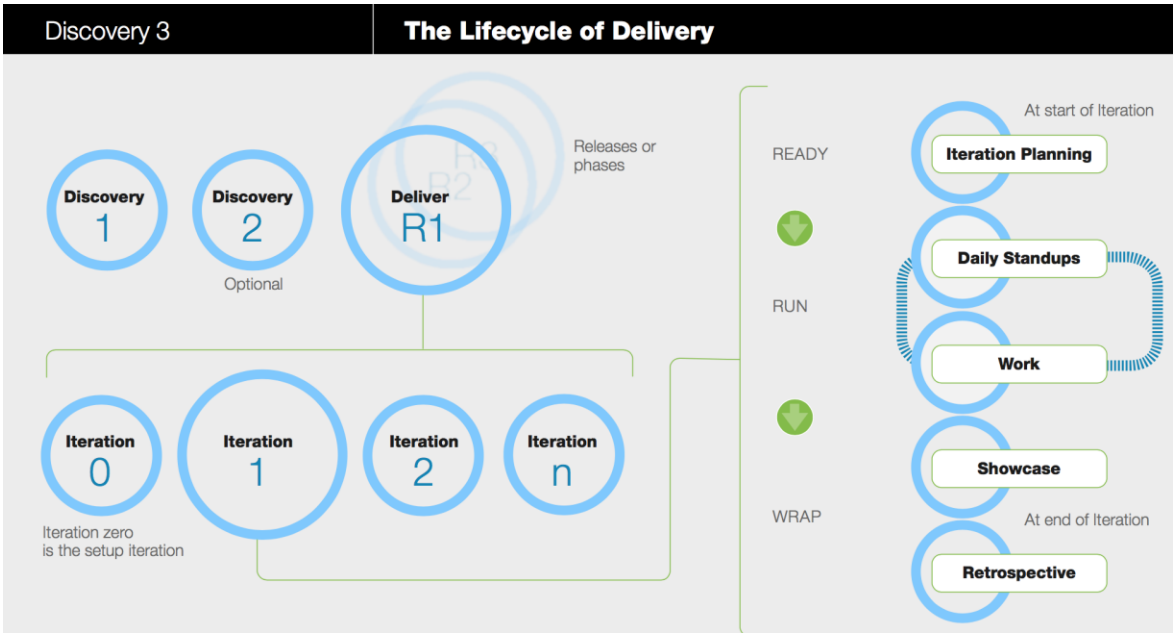

Figura 6: Ciclo de vida del *Delivery (IBM Corp., 2015)*

Iniciamos con una iteración 0 en la que se prepara todo lo necesario para poder iniciar con la implementación, se elabora sobre los siguientes puntos:

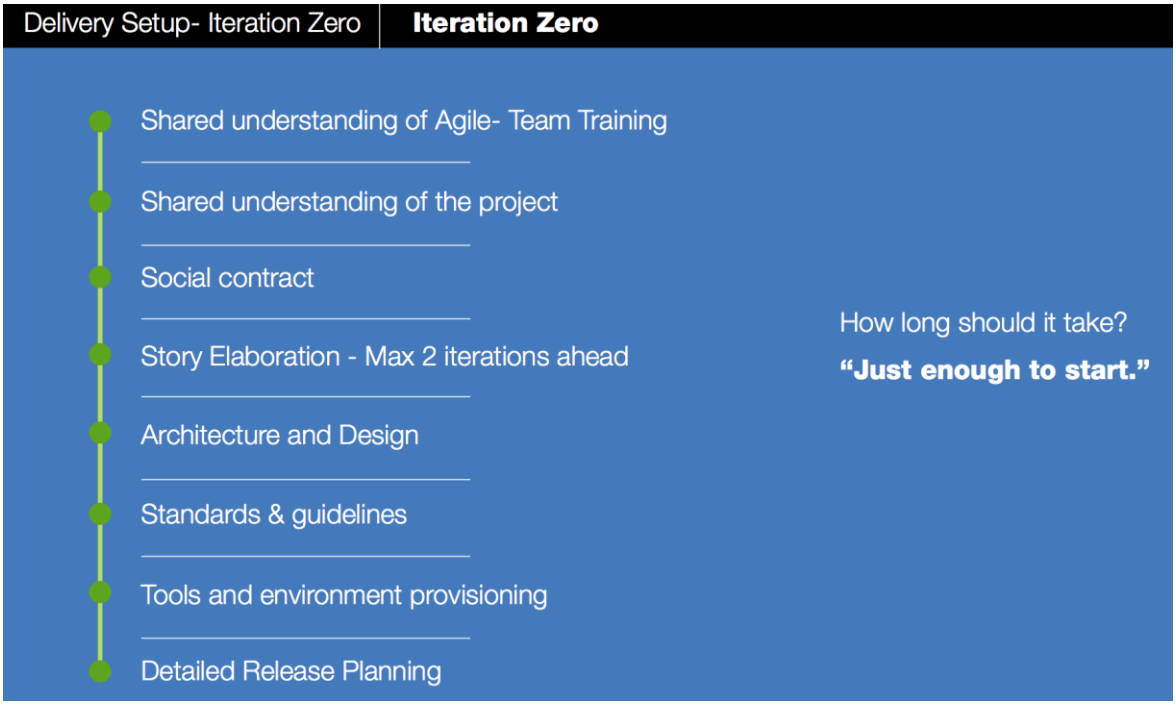

Figura 7: Iteración cero *(IBM Corp., 2015)*

Una vez terminada la iteración 0, se elabora un *release plan* para poder definir las metas e hitos del proyecto y compartir proyecciones sobre estos con los *Stakeholders*. Esto se logra mediante los siguientes pasos:

- Priorizar Historias de usuario
- Estimar esfuerzo
- Estimar la velocidad del equipo
- Determinar lo que se puede incluir en cada iteración
- Guardar espacio para contingencias
- Preparar el muro del *release*

Una vez terminado el *release plan*, ejecutamos el proyecto siguiendo algunas prácticas recomendadas para asegurar su correcto seguimiento y que se tomen las decisiones correctas durante dicha ejecución: (ver marco teórico en la sección 'Prácticas ágiles' para encontrar detalles de cada una).

- *Backlog refinement*
- *Planning poker*
- *Standups*
- *Continuous integration & deployment*
- *Showcases*
- *Retrospectives*
- *Big Visual Charts*

Es importante mencionar que el plan se revisa al terminar cada iteración, ya que es necesario planear la siguiente tomando como entrada los resultados de la iteración previa, es aquí donde se pueden hacer ajustes a la ejecución y posiblemente al plan general. Esto nos brinda un gran valor agregado, ya que no esperamos al final del proyecto o del *release* para darnos cuenta de que algo no va bien o de que va mejor de lo esperado.

- Re-estimar historias de usuario no cubiertas en la iteración
- Re-estimar velocidad del equipo
- Tomar en cuenta primero las historias de usuario no implementadas en la iteración anterior
- Actualizar el *release* plan en base a los cambios
- Actualizar el muro del *release*

2.4 Planeación o Cronología del proyecto llevado a cabo.

A continuación, se presenta el plan original y la cronología real del proyecto por mes:

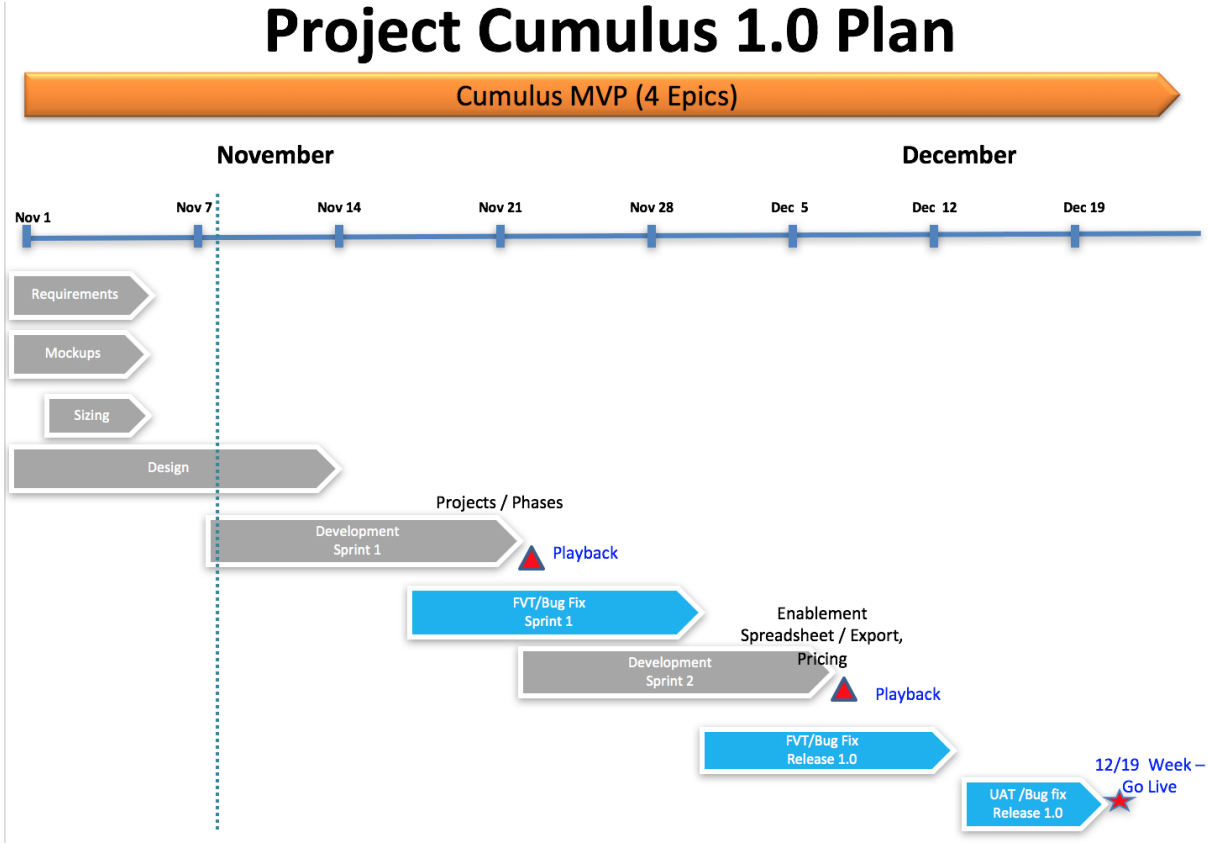

Figura 8: Plan original

Septiembre 2016

Workshop de Discovery dividido en dos fases:

- Una para entendimiento de necesidades y requerimientos.
- La segunda para la definición del proceso de desarrollo a seguir y establecer la arquitectura y tecnologías a emplear.

Octubre 2016

Release 1 (Oct. 3 – Nov. 4):

- Iteración 0
    - POT (*Proof of technology)*
    - Implementación de requerimientos no funcionales[III]

[III] Requerimientos no relacionados directamente con alguna funcionalidad del sistema sino con aspectos como su usabilidad, desempeño. Normalmente el diseño de un sistema debe considerarlos.

Noviembre 2016

Release 2 (Nov. 7 – Dic. 2):

- Iteración 1 (Nov. 7 – Nov. 18)
    - Administración de Proyectos
    - Administración de Fases
    - *Playback session*
- Iteración 2 (Nov. 21 – Dic. 2)
    - Administración de Métricas
    - Administración de Catálogos
    - *Playback session*

Diciembre 2016

- *Release* 3 (Dic. 5 – Dic 30):
    - Iteración 3 (Dic. 5 – Dic. 16)
        - Administración de Productos
        - Administración de Chargable Components
        - Administración de Additional Components
        - *Playback session*
    - Iteración 4 (Dic. 19 – Dic 30)
        - *Offerings grid*
        - *Playback session*

Enero 2017

- *Release* 4 (Ene 2 – Ene 13):
    - Iteración 5
        - Solución de defectos
        - Exportar hoja de habilitación
        - Retrospectiva

2.5 Descripción de actividades

Etapa de *discovery*

Mi participación en esta etapa consistió en formar parte activa de las reuniones, proponiendo posibles soluciones a los problemas encontrados y buscando comprender el proceso actual para también proponer mejoras al mismo, iniciamos en Septiembre de 2016 con un Workshop en la ciudad de Raleigh, Carolina del Norte, al cual acudieron personas clave en la ejecución del proceso de adquisiciones, personal técnico, analistas de negocio y habilitadores de la metodología ágil. Se definió y ejecutó una agenda de trabajo en la que participó todo el equipo arriba nombrado (Ver fig. 6, *Workshop Agenda*).

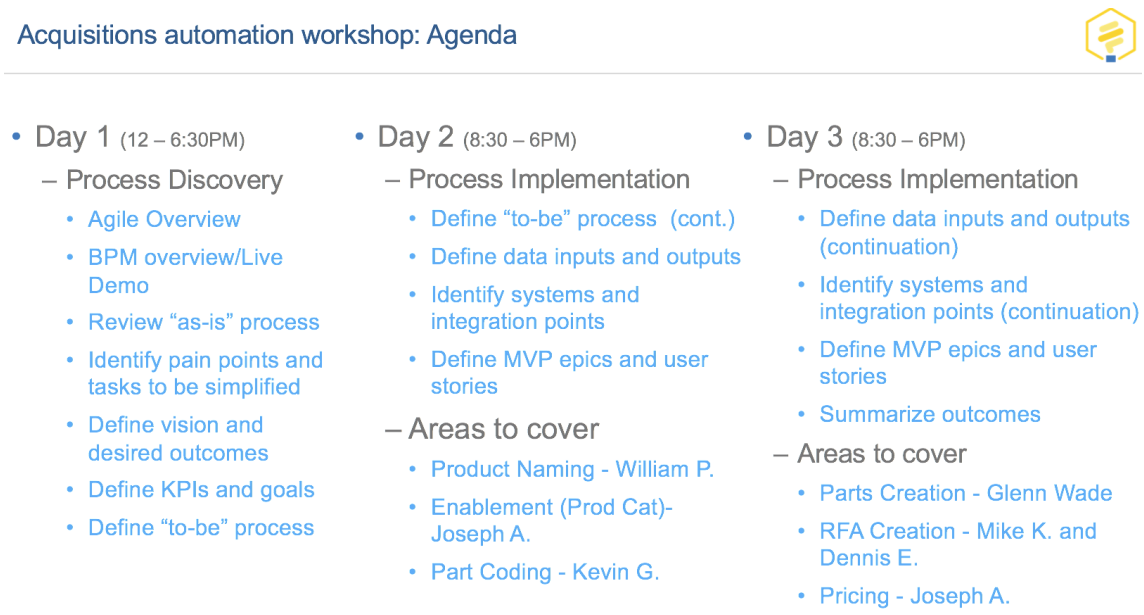

Figura 9: *Workshop agenda*

En dicho workshop identificamos el proceso actual y cuáles pasos serían afectados por el proyecto una vez culminadas todas sus fases (Ver fig. 7, Proceso actual), las necesidades de negocio y los beneficios esperados con la implementación (Ver sección 2.1 Antecedentes del proyecto reportado), y por último definimos un *release plan* para el proyecto (Ver fig. 5, Plan original), la necesidad de obtener beneficios en corto tiempo nos llevó a definir un *Minimum Valuable Product* o *MVP* que pueda ser entregado en 3 meses, este contiene la infraestructura base para la aplicación y una solución para acelerar la creación de la hoja de habilitación o *Enablement Spreadsheet*.

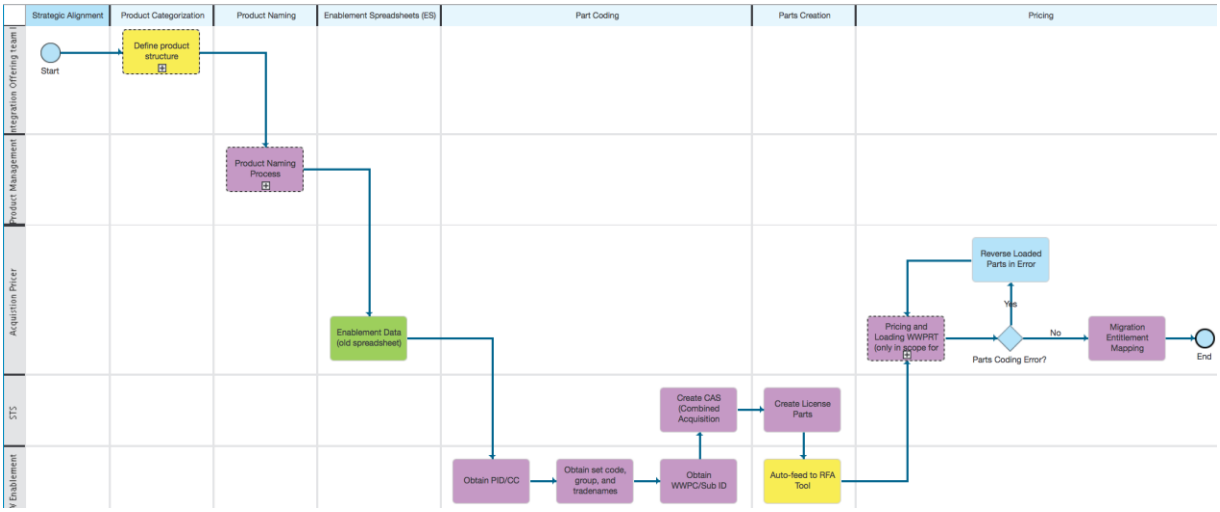

Figura 10: Proceso actual

Iteración cero

Una vez terminado el workshop de *discovery*, los miembros del equipo técnico nos mantuvimos reunidos por una semana más, iniciando con esto la iteración 0, entre los presentes contamos con gerente del área técnica, un arquitecto de software, dos desarrolladores, un líder técnico de otro proyecto como apoyo y yo como líder técnico del proyecto específico. Nos planteamos los siguientes objetivos:

- Definir el proceso de desarrollo a seguir
- Definir el repositorio a utilizar y el mecanismo para la administración del proyecto
- Definir la arquitectura, tecnologías y herramientas a utilizar
- Diseñar algunas partes de la aplicación como la base de datos (Ver fig. 8, Modelo de datos)
- Determinar el número de recursos y las habilidades necesarias para cumplir con el plan a 3 meses.

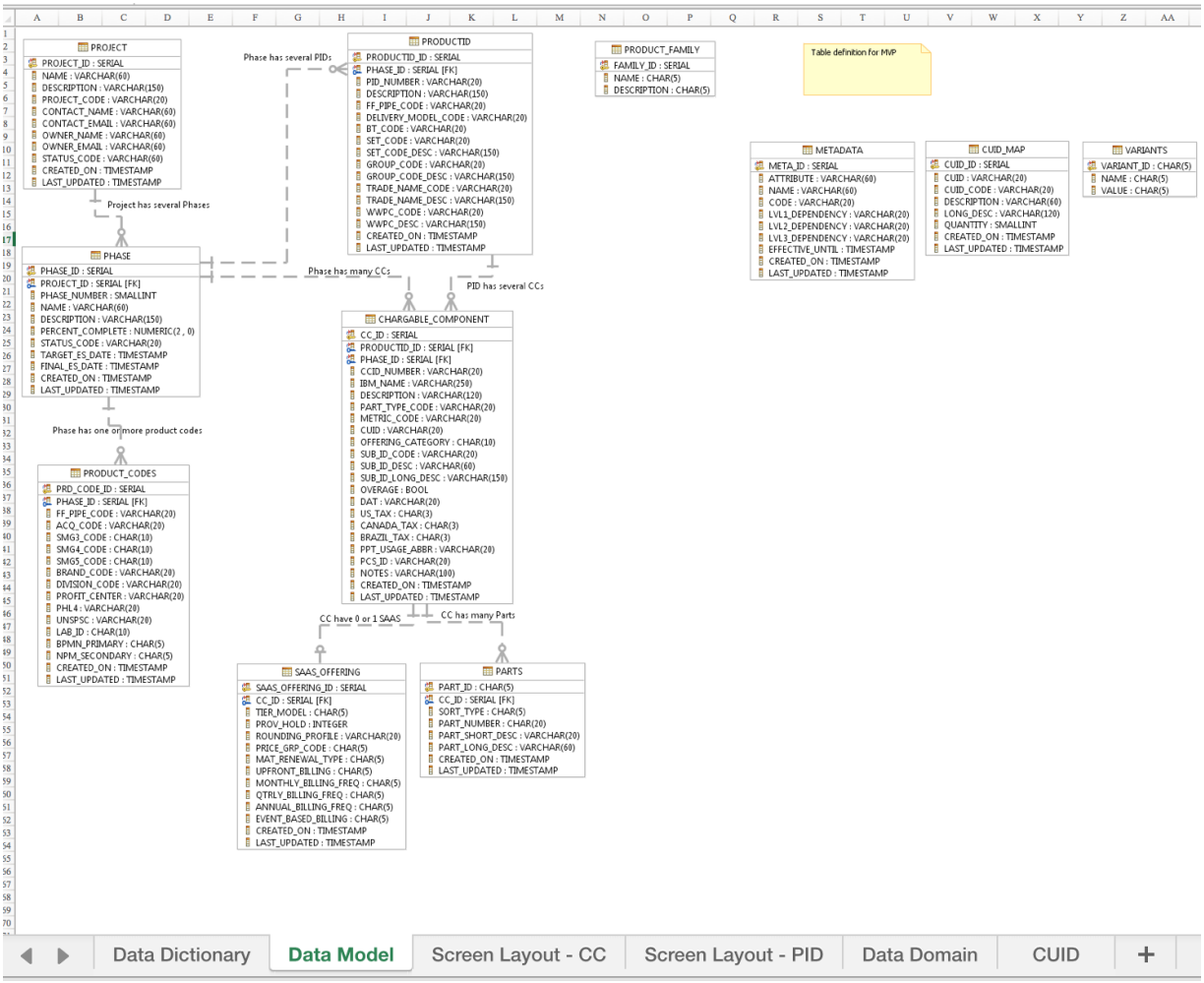

Figura 11: Modelo de datos

Sostuvimos varias reuniones en las que revisamos las prácticas más comunes de la metodología Agile y con base en ellas definimos un proceso de desarrollo a seguir durante el proyecto y eventualmente a ser implementado en el área completa (Ver fig. 9, Proceso de desarrollo).

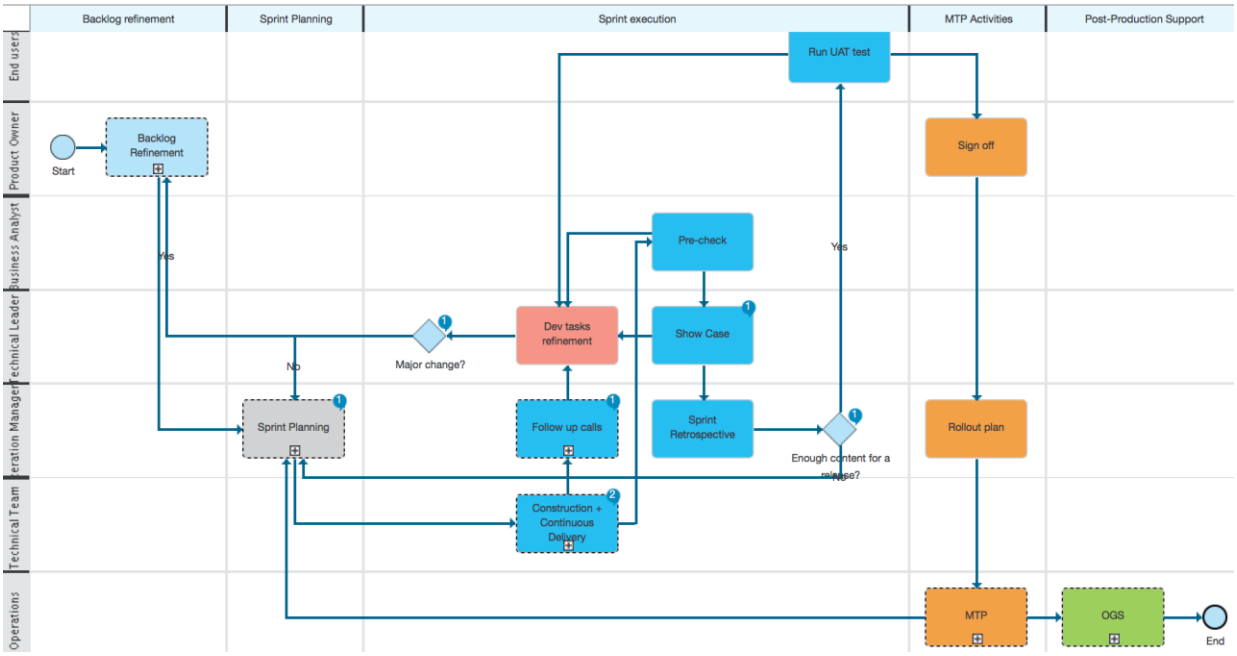

Figura 12: Proceso de desarrollo

Posteriormente revisamos el tema de la administración de proyecto, acordando utilizar *Github* como repositorio de código y *Zenhub* como herramienta de monitoreo y control, esto como un piloto para posteriormente evaluar su uso extendido en el área *SmarterProcess*.

Sobre la arquitectura, determinamos utilizar una plataforma de servicios en la nube llamada *Bluemix*, para albergar la aplicación y consumir servicios de base de datos ya disponibles en la misma, esta es proveída por la empresa y tiene un costo interno para unidades de negocio que la utilizan. (Ver fig. 10, Arquitectura).

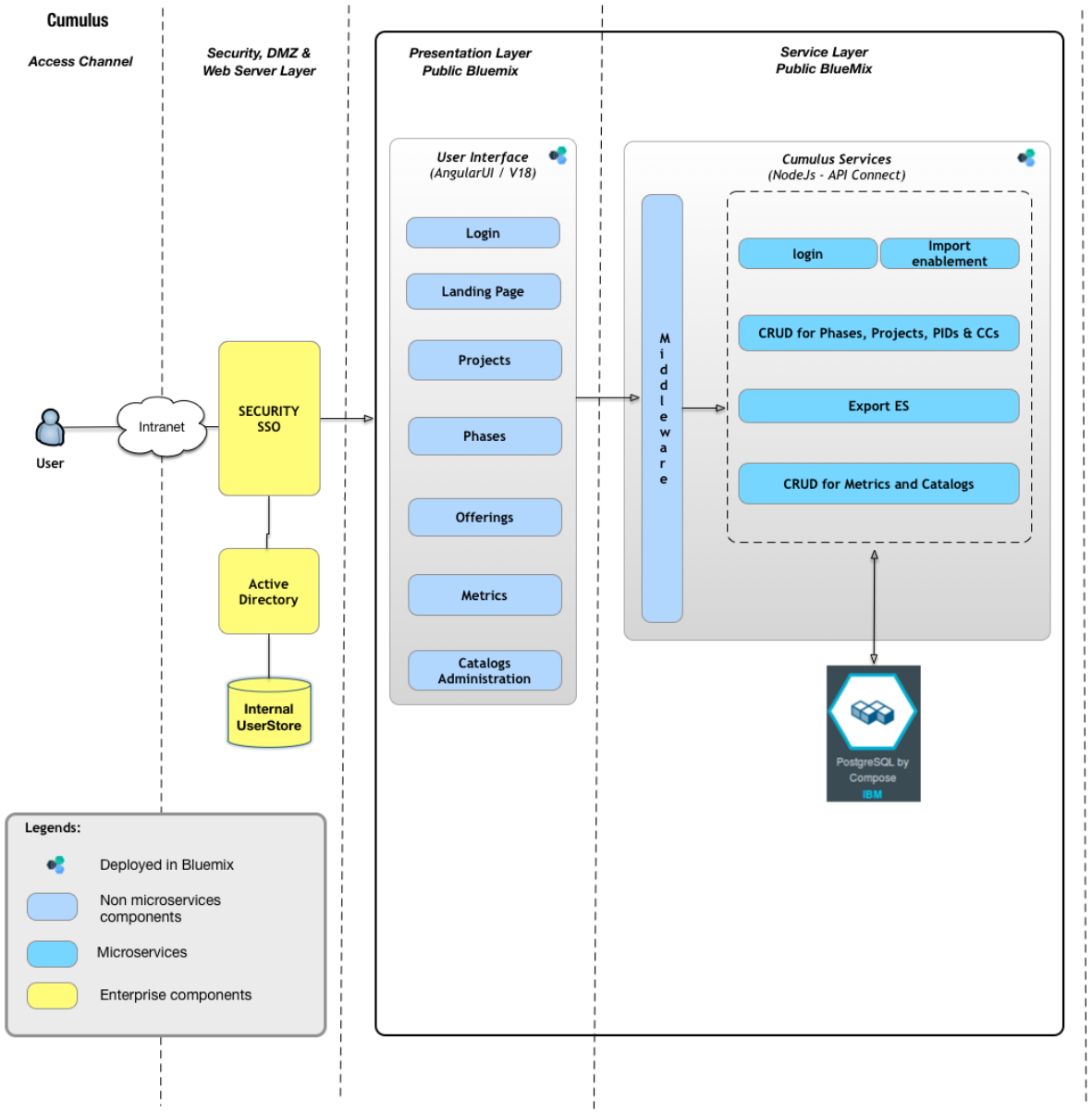

Figura 13: Arquitectura

Para determinar el lenguaje de desarrollo a utilizar se revisó el listado de recursos disponibles y sus habilidades, encontrando que sería mejor opción implementar el *back-end*[IV] en *NodeJS*[V], ya que todos dominan esta tecnología, se decidió también utilizar un *framework*[VI] existente para este lenguaje que facilite y agilice el desarrollo. Otra opción que teníamos era el lenguaje *Java*, pero uno de los desarrolladores no tenía experiencia en éste. Esta fue la base para el análisis posterior para elegir: *NodeJS framework* (*API Connect* o *Express*), administrador de bases de datos (*DB2, PostgreSQL* o *Cloudant*) y lenguaje para el *front-end* (*Angular* o *Angular II*).

Decidimos ejecutar una prueba de tecnología o *POT* con duración de una semana que consistió en la implementación de un *CRUD*[VII] simple para el catálogo de proyectos que consiste solo de un identificador numérico, nombre corto, nombre, creador y fecha de creación usando las distintas opciones de tecnologías y realizar un análisis posterior para elegir las más viables.

Ésta fue ejecutada por los desarrolladores de la siguiente manera:

- Aplicación 1: *Express, DB2* y *Angular*.
- Aplicación 2: *API Connect, PostgreSQL* y *Angular*.
- Aplicación 3: *API Connect, Cloudant* y *Angular II*.

La ejecución de dicha prueba de tecnología nos dio los argumentos suficientes para elegir *API Connect* como *framework* de desarrollo, *PostgreSQL* como administrador de bases de datos y *Angular* como lenguaje para implementación del *front-end*. (Ver tablas 1, 2 y 3).

NodeJS Framework	**Curva de aprendizaje**	**Creación automática de *endpoints*[VIII]**	**Configurable**	**Documentación automática de *API*[IX]**
Express	Pequeña, casi nula	No	Si	No
API Connect	Moderadamente alta	Si	Si	Si

Tabla 1: Comparativa de *Frameworks* de *NodeJS.*

[IV] Se refiere a la parte de un sistema que se ejecuta del lado del servidor

[V] Lenguaje de programación basado en JavaScript

[VI] Piezas de código a manera de esqueleto utilizadas como base para desarrollar un sistema

[VII] Acrónimo creado a partir de las palabras *Create, Read, Update, Delete*, se refiere a la capacidad para realizar estas acciones sobre una entidad específica en un sistema.

[VIII] URL correspondiente a un servicio

[IX] Acrónimo para *Application Program Interface*, un conjunto de métodos que se exponen para ser utilizados por un tercero.

DBMS	**Requerimientos adecuados para tipo de base de datos**	**Disponible en *Bluemix***	**Implementación viable con *NodeJS framework***	**Precio**
DB2	Si	Si	Si	$$$
PostgreSQL	Si	Si	Si	$
Cloudant	No	Si	Si	$

Tabla 2: Comparativa de administradores de bases de datos (*DBMS*).

Front-end framework	**Curva de aprendizaje**	**Compatibilidad con otras librerías**	**Compatible en todos los navegadores**
Angular	Nula	Si	Si
Angular II	Alta	Insuficiente	Si

Tabla 3: Comparativa *Angular vs Angular II*.

Una vez terminada la prueba de tecnología y teniendo definidas las herramientas a utilizar, se definieron tareas a realizar como parte del entregable de la iteración cero, conteniendo el código base funcional para la aplicación, capacidad para los usuarios para firmarse en la misma y la implementación de los estilos estándar de la empresa.

Se determinó también que el equipo estaría conformado por un arquitecto de software, un *Iteration manager*, un analista de negocios y un experto de experiencia de usuario, todos a tiempo parcial, también un líder técnico (mi rol principal) y 3 desarrolladores, uno en México, uno en estados unidos y uno en Bielorrusia.

Al mismo tiempo que el equipo técnico estaba reunido revisando los puntos antes mencionados, el analista de negocios sostenía reuniones con el cliente para detallar las historias de usuario a ser implementadas.

Implementación

La implementación se llevó a cabo entre Octubre y mediados de Enero, ya que requerimos de algunos días extras al plan inicial para estabilización y completar algunas tareas pendientes (Ver figura 5 para conocer el plan original).

Acordamos tener las siguientes reuniones diarias durante la ejecución del proyecto:

- Llamada técnica diaria con una duración de 30 minutos dirigida por mí, en la que se encontraba invitado el equipo técnico: los tres desarrolladores, el arquitecto y el gerente, aquí revisamos dudas con requerimientos, problemas de cualquier tipo, toma de decisiones técnicas y avance en general.
- Llamada *stand-up* diaria, con duración de 15 minutos y dirigida por la *iteration manager* contaba con el equipo extendido del proyecto, esto es, *iteration manager*, analista de negocios, experto en experiencia de usuario, *product owner* además del equipo técnico y cualquier otra persona que pudiera ser requerida en cierto momento del proyecto. Cada integrante del equipo menciona qué hizo el día anterior, cuál es el plan para el día actual y si existe algún impedimento actualmente para realizar sus actividades, si surge algún tema que requiera de más tiempo de revisión o discusión se revisa posterior a la llamada solo con los involucrados o afectados.
- Reunión de *sprint planning*, ocurre al inicio de cada *sprint* o iteración y tiene una duración de una hora aproximadamente, y en ella se planean las tareas a ser implementadas durante dicha iteración, estas se toman del *product backlog* considerando prioridad, estimado y disponibilidad de los recursos necesarios.
- Se calendarizaron y realizaron también *Playbacks* o demos en cada iteración para mostrar el avance al cliente, recibir retroalimentación y así poder enderezar el curso en caso de algún problema, una vez cerrados estos puntos o con el acuerdo de implementarlos después, se realizaba la instalación de avances para revisión.
- Se acordó calendarizar reuniones extras con forme fueran necesarias para revisar puntos varios, solo invitando a los involucrados.
- Por último, se acordó tener una retrospectiva al terminar el proyecto para revisar qué se hizo bien y qué no tan bien y proponer así mejoras en diversos rubros.

Comenzamos con la implementación del proyecto, y mi colaboración inició como el punto de encuentro entre la parte del negocio y la técnica, sosteniendo reuniones con el analista de negocio para comprender y ayudar a documentar los requerimientos a manera de historias de usuario, posteriormente explicarlos a los desarrolladores y crear en conjunto con ellos tareas técnicas basados en la revisión de alternativas de implementación algunas veces en conjunto con el arquitecto de *software*, la idea es que estas tareas queden

listas para ser estimadas y asignadas a un desarrollador. Para la administración de historias de usuario y tareas se utilizó una herramienta llamada *Zenhub*, esta cuenta con una forma sencilla de documentar, priorizar y estimar tareas, así como dar seguimiento a su implementación para cuidar que se ejecuten en tiempo y forma.

Conforme avanzó el tiempo, tuvimos diversos problemas, siendo el más grave la ausencia por vacaciones del analista de negocios, esto se dio a inicio de Noviembre y a partir de esa fecha tomé su rol también.

Al presentarse este cambio, notamos que los detalles de varios de los requerimientos que habíamos recibido por parte del analista de negocios y que planeábamos implementar en próximas iteraciones discrepaban de lo que el cliente en realidad tenía en mente, por lo que el progreso se vio afectado, tuvimos que re-trabajar varias partes del sistema y planear sesiones más continuas con el cliente para evitar que esto siga ocurriendo.

Durante las revisiones de avance que se tienen después de cada iteración, se decidió agregar un nuevo desarrollador para apoyar al equipo debido a los cambios y la necesidad de entregar en el mismo tiempo, lo que eventualmente causó que la *Iteration manager* salga del proyecto por temas de presupuesto, tomando yo también ese rol.

Debido a esto tuvimos que cambiar un poco el formato de las reuniones: dejamos una sola llamada diaria con todo el equipo (que en su mayoría era técnico) y yo mantuve llamadas aparte directamente con el cliente y el experto en experiencia de usuario para revisar detalles de requerimientos, más adelante, también decidimos dejar fuera del alcance la parte de precios, todo esto acordado y validado con el *Product owner*.

Es aquí donde toma mayor importancia la metodología empleada en el proyecto, ya que por sus cualidades adaptativas más que predictivas, permitió reaccionar y tomar decisiones que incluyeron la reducción del alcance del proyecto para compensar el re-trabajo y dar cabida a los cambios y a los nuevos requerimientos.

Al realizar el último *playback* aún teníamos algunos defectos y cambios por implementar y la fecha original estaba comprometida, por lo que se negoció hacer una entrega en tiempo con el nivel de código al momento y tener un *release* más en Enero en el que se entregarían defectos y cambios resueltos y aplicados más la funcionalidad pendiente de exportar a Excel.

Cierre de proyecto

Una vez teniendo cerrados los defectos y habiendo obtenido la aceptación del usuario, se realizó una reunión de retrospectiva con el equipo para revisar qué puntos estuvieron bien, cuáles mal y cuáles se pueden mejorar y de esta forma tomarlos en cuenta en la ejecución de próximas fases (Ver figura 11, Retrospectiva).

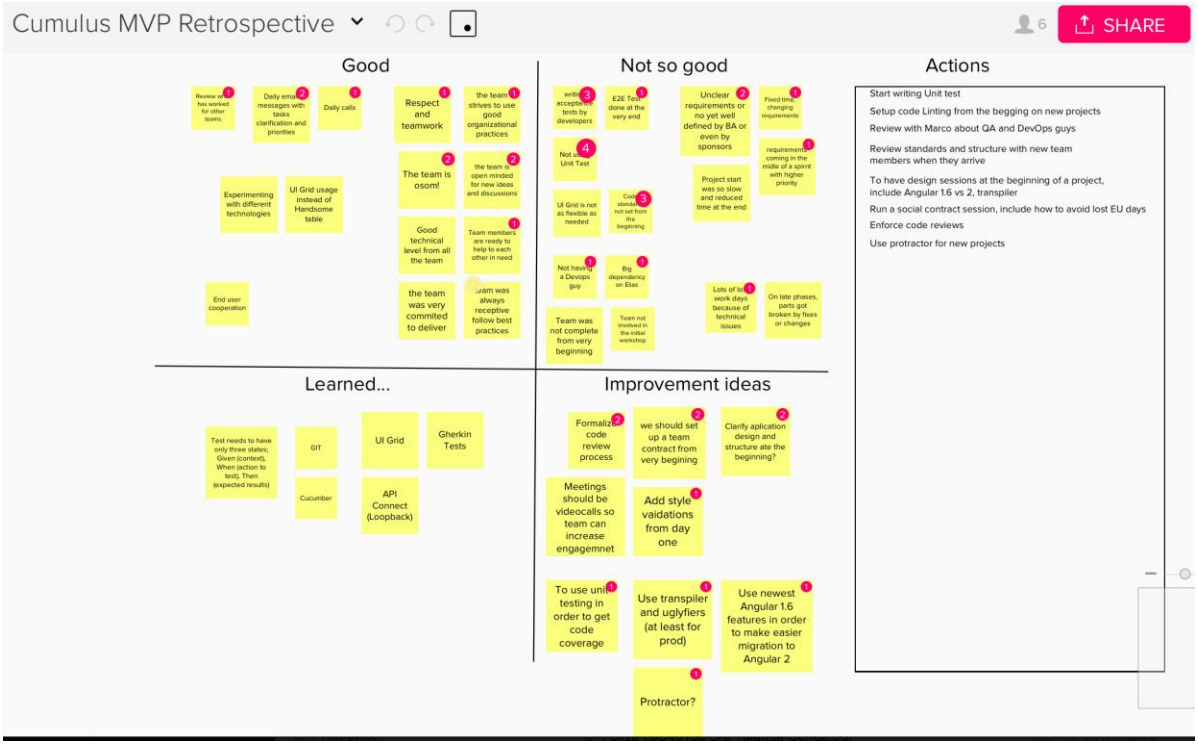

Figura 14: Retrospectiva

2.6 Resumen de la documentación e información recabada

Muestra del diccionario de datos:

	A	B	C	D	E	F	G	H	I
	Colum	Column name	Object	Spreadsheet Column	Database table	Database attribute	Responsible team (owr	Data type	Required
1	1	Sort Code	PID, CC	A				Numbers	
2	2	Sort Type	Parts	B	PARTS	SORT_TYPE	Pricer	Alphabetic	Yes
3	3	FF Pipe	PID	C	PRODUCTID	FF_PIPE_CODE	Pricer	Alphabetic	Yes
4	4	Part Type	CC	D	CHARGABLE_COMPONENT	PART_TYPE_CODE	Pricer	Alphanumeric	Yes
5	5	BT	CC	E	PRODUCTID	BT_CODE	Pricer	Alphanumeric	Yes
6	6	DAT	CC	F	CHARGABLE_COMPONENT	DAT	??	Alphabetic	Yes
7	7	US Tax	CC	G	CHARGABLE_COMPONENT	US_TAX	Pricer	Alphabetic	Yes
8	8	Canada Tax	CC	H	CHARGABLE_COMPONENT	CANADA_TAX	Pricer	Alphabetic	Yes
9	9	Brazil Tax	CC	I	CHARGABLE_COMPONENT	BRAZIL_TAX	Pricer	Alphabetic	Yes
10	10	PPT Usage ABBR	CC	J	CHARGABLE_COMPONENT	PPT_USAGE_ABBR	??	??	Yes
11	11	PPT Usage	CC	K			??	??	Yes
12	12	PCS Identifier	CC		CHARGABLE_COMPONENT	PCS_ID	??	??	Yes
13	13	PID #	PID		PRODUCTID	PID_NUMBER	??	??	Yes
14	14	PID Description	PID		PRODUCTID	DESCRIPTION	Pricer	Alphanumeric	Yes
15	15	Product Family	Product Family				??	Alphanumeric	Yes
16	16	Product Code	Product Family				??	Alphanumeric	Yes
17	17	Heritage Name	Product Family				??	Alphanumeric	Yes
18	18	Heritage UOM	Product Family				??	Alphanumeric	Yes
19	19	IBM Name	CC		CHARGABLE_COMPONENT	IBM_NAME	Pricer	Alphanumeric	Yes
20	20	CC ID#	CC		CHARGABLE_COMPONENT	CCID_NUMBER	??	??	Yes
21	21	CC Description	CC		CHARGABLE_COMPONENT	DESCRIPTION	Pricer	Alphanumeric	Yes
22	22	Metric	CC		CHARGABLE_COMPONENT	METRIC_CODE	Pricer	Alphanumeric	Yes
23	23	CUID	CC		CHARGABLE_COMPONENT	CUID	WWERB request or enablr	Alphanumeric	Yes
24	24	PA or FCT	CC				Pricer	Alphabetic	Yes
25	25	Status	CC				??	Alphanumeric	Yes
26	26	Notes	CC				??	Alphanumeric	Yes
27	27	Variable Royalties?	CC				??	Alphanumeric	Yes
28	28	Change Date	CC				??	Alphanumeric	Yes
29	29	Nature of Change	CC				??	Alphanumeric	Yes
30	30	Set Code	PID		PRODUCTID	SET_CODE	??	Alphanumeric	Yes
31	31	Set Code Desc	PID		PRODUCTID	SET_CODE_DESC	Pricer	Alphanumeric	Yes

Figura 15: Diccionario de datos

Ejemplos de solicitudes de cambio como resultado de los *playbacks* ejecutados:

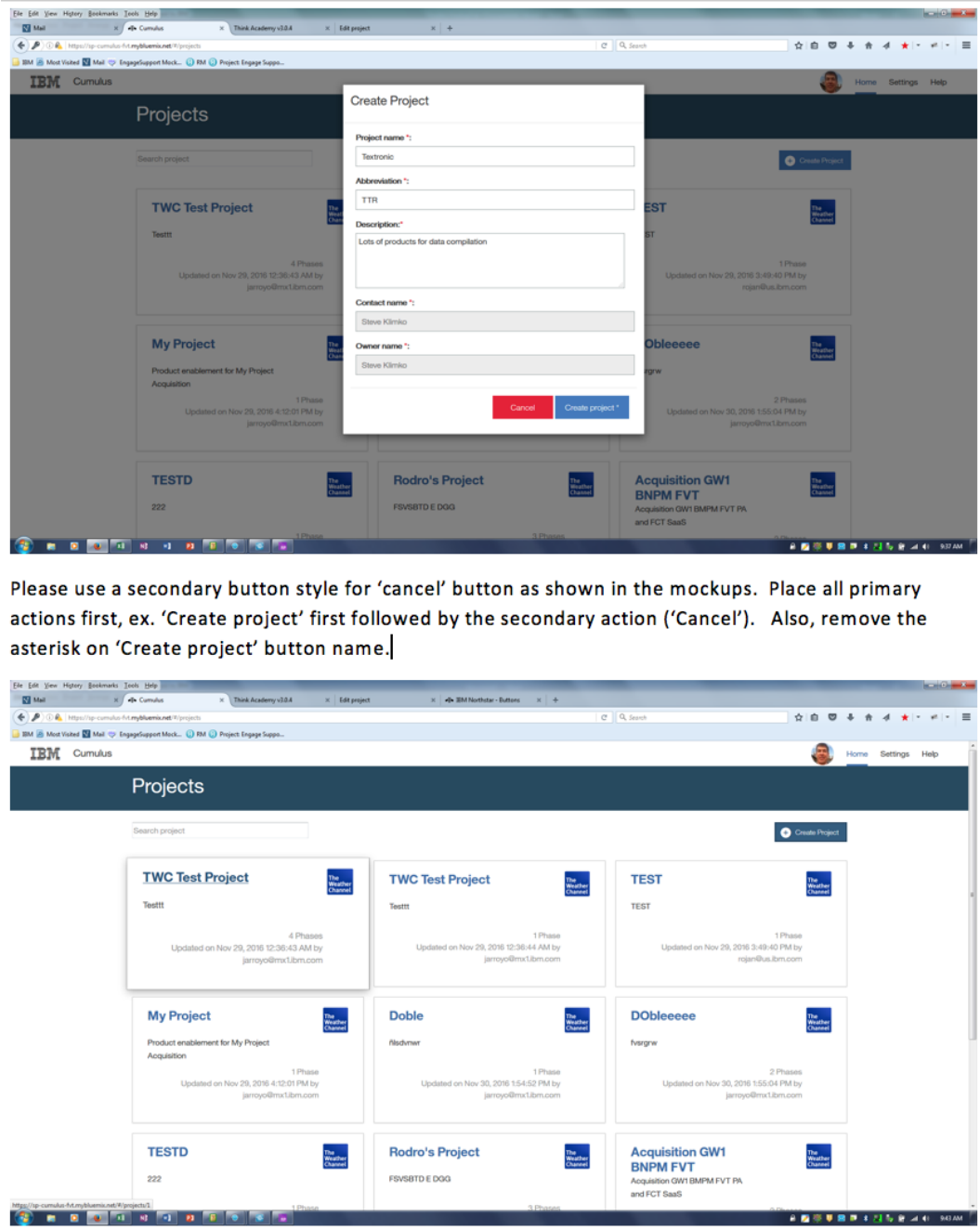

Figura 16: Solicitudes de cambio

Ejemplo de hoja de habilitación:

Sort Code	Sort Type PID/CC CC AC	FF Pipe PA FCT BMX	Part Type (see guidelines tab or embedded comment) -Must indicate if Labor Based vs Tech Based	BT PA=59 FCT=15 BMX=??	DAT	US Tax	Canada Tax	Brazil Tax	PPT Usage ABBR	General SaaS Type PPT Usage	PCS Identifier	PID #	PID Description (47) - **PID** - No Dash in PID - 4th sort in SQO
1	PID/CC		Monthly w/Spt - Tech	BT59		Z	Z	S					**Weather Company Data f**
1	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Data fo
1	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Data fo
	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Data Pa
	AC		Overage	BT59		Z	Z	S					Weather Company Data Pa
	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Data Pa
	AC		Overage	BT59		Z	Z	S					Weather Company Data Pa
	PID/CC		Monthly w/Spt - Tech	BT59		Z	Z	S					**Weather Company Alerts**
	AC		Overage	BT59		Z	Z	S					Weather Company Alerts fo
	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Alerts fo
	AC		Overage	BT59		Z	Z	S					Weather Company Alerts fo
	CC		Monthly w/Spt - Tech	BT59		Z	Z	S					Weather Company Alerts fo

Figura 17: Hoja de habilitación

Mapa de geolocalización de *stakeholders* del proyecto:

Figura 18: Mapa de localización de *stakeholders*

2.7 Resultados obtenidos en el proyecto reportado

Según palabras del usuario principal actualmente se logró reducir en 50% la elaboración de la hoja de habilitación (de 4 a 2 semanas), como se explicó en la sección 2.2, este es el objetivo directo que entrega valor al negocio, es difícil expresarlo en términos monetarios, pero si consideramos que pasó a ser un proceso de una sola persona, ejecutado en 4 semanas a un proceso en el que se puede involucrar a otros y que redujo el tiempo necesario en 50%, y multiplicamos esto por el número de adquisiciones que deben procesar anualmente, nos damos cuenta que se removió uno de los mayores cuellos de botella en el proceso de habilitación de productos y estos pueden salir al mercado en un tiempo menor.

Otra de las ventajas entregadas es el poder administrar varias adquisiciones en una sola herramienta, anteriormente esto se llevaba en archivos de Excel y se contaba con distintos archivos por cada proyecto de adquisición, ahora toda esta información se encuentra centralizada en el sistema, evitando tener archivos separados y versionados solo en la computadora de la persona trabajando con ellos.

Aunque aún falta cubrir todos los demás puntos del proceso de adquisiciones para empezar a obtener beneficios a gran escala, el verdadero éxito de este proyecto es el hecho de que sirve como *show case* para que los dueños de otras áreas comprueben que obtendrán un beneficio tangible en corto plazo, eliminando posibles dudas y ayudando con la resistencia al cambio que pudiera generarse.

Por último, el proyecto es un caso de éxito en la adopción de la metodología ágil, ya que se demostró que tuvimos capacidad de reacción ante los cambios en el equipo y en los requerimientos y se logró una sinergia con el área del negocio para llegar a acuerdos a favor del proyecto.

CAPÍTULO III. CONCLUSIONES

3.1 Lecciones aprendidas

Este proyecto trajo un buen número de lecciones aprendidas para mí, trato de agruparlas en 4 grupos principales:

Método tradicional vs *Agile*

El método tradicional de planeación de proyectos de software trata de predecir, o también se pudiera decir "adivinar" el tiempo y recursos necesarios para implementar algo que muy difícilmente está completamente definido y comprendido al momento que dicha planeación ocurre, personalmente, estuve peleando con este problema por años en un buen número de proyectos y es hasta ahora que aprendí las bases y experimenté la ejecución de la metodología ágil puedo decir que hay otra forma de hacerlo y ésta funciona.

La naturaleza adaptativa de *Agile* ayuda a centrarnos en lo que es más importante para el cliente y le puede dar más valor, comprenderlo e implementarlo de una manera acompañada que nos ayuda a reducir el riesgo de re-trabajos y malos entendidos.

Entregamos un plan inicial pero dejamos claro que no es necesariamente como terminará, la razón es sencilla, el cambio es una constante en los proyectos de software y entendemos que es muy difícil, casi imposible saber el detalle completo de los requerimientos y adelantarnos a decisiones que se tomarán, entonces, lo que hacemos es dividir el proyecto en entregas que se planean individualmente y revisar la viabilidad de ese plan al momento de planear la siguiente entrega, de esta forma corregimos el trayecto y decidimos sobre el camino.

Resistencia al cambio

Siempre hemos escuchado sobre resistencia al cambio, en este proyecto tuve que lidiar con ello a distintos niveles, primero con los ejecutores del proceso que piensan que no se pueden hacer las cosas mejor y que no vale la pena el riesgo y los recursos a invertir para intentarlo, por otro lado, personas del equipo de implementación acostumbrados a una forma de trabajo tradicional y a ciertas herramientas y cuya respuesta inicial a propuestas de cambios dicha esta forma de trabajo es 'No'.

Lo que aprendí en este tema es, que la mejor forma de convencer a alguien es mostrándole ejemplos y casos prácticos de cómo se puede beneficiar con algo nuevo, no con teoría y mucho menos imponiendo.

Herramientas

Si bien, es mucho más importante el centrarnos en los individuos que en los procesos y herramientas, también es cierto que la elección de una herramienta adecuada, que ayude a permear el método a utilizar es de gran utilidad en la adopción del mismo. En este caso, utilizamos una herramienta para administración de proyectos llamada *Zenhub*, esta se desarrolló como una extensión al repositorio de código *Github* y juntos ayudan a ejecutar las prácticas principales que adoptamos como un equipo ágil logrando así mostrar el valor de las prácticas al equipo sin mucho esfuerzo.

Sobre la documentación

Aprendí también que tiene mayor valor una pieza de software funcional que una documentación de requerimientos exhaustiva, aun cuando esta contenga *mock-ups*[X], ya que estos tienden a quedar obsoletos y generan más trabajo para mantenerlos actualizados, además de que pueden convertirse en armas para el negocio o el quipo implementador y dificultar la negociación de cambios en requerimientos y planes.

3.2 Propuesta de mejora

El principal punto que mejoraría si tuviera oportunidad de implementar el proyecto de nuevo es la revisión y cotejo de requerimientos con el cliente tan pronto como fuera posible, en este proyecto tuvimos la "suerte" de encontrarlo en una etapa temprana cuando se fue el analista de negocio que fungía como *product owner* y hubo una relación más directa del equipo de implementación con el cliente final que tomó este rol.

La metodología nos hubiera llevado ahí eventualmente, pero al menos un par de semanas después de cuando ocurrió y considerando que esto tuvo un impacto en tiempo y alcance, encontrarlo después habría complicado las cosas aún más.

Las demás propuestas de mejora para este proyecto están enfocadas al área técnica:

En este rubro, la principal es la falta de implementación de pruebas automatizadas. Este fue un punto que tuvimos en mente desde el principio, pero lo mantuvimos con baja prioridad por buscar entregar algo funcional en el menor tiempo posible, el error fue no retomar el tema hasta que la misma complejidad del proyecto nos lo recordó, pues las pruebas manuales de lo recién implementado más las pruebas de regresión de lo ya entregado tomaron cada vez más tiempo y entregaban menos certeza al ser un proceso manual.

[X] Representaciones estáticas de una pantalla que sirven como referencia o como guía para realizar la implementación de la misma

Otro punto a mejorar sería establecer los estándares de codificación a seguir desde el inicio del proyecto, en este caso lo hicimos cuando ya teníamos un buen avance y tuvimos que dedicar algo de tiempo a corregir cosas que pudieron hacerse conforme al estándar desde el principio.

3.3 Conclusiones

La conclusión a la que llego después de recapitular el trabajo realizado y los resultados obtenidos en este proyecto es que utilizando una metodología "tradicional", incluso una iterativa-incremental, el riesgo de fracaso hubiera sido mucho mayor, siendo el principal problema el alcance pactado en un inicio para requerimientos que no están claros aún.

Este no es un escenario aislado o raro, de hecho, he vivido con este problema durante los 12 años que llevo desarrollando software, y sé que es un problema general en la industria. En mayor o menor medida iniciamos un desarrollo de software sin la certeza total de los requerimientos y esto no necesariamente está mal, ya que el esfuerzo para obtener todos los detalles para implementar cada uno de los requerimientos es demasiado y aún con esta inversión, no existe garantía de que los mismos se mantendrán intactos durante el tiempo de implementación del proyecto.

La gente, el ambiente del negocio, la tecnología, son factores no estáticos y debemos estar preparados para responder a cambios en los mismos sin que esto sea una lucha entre el equipo implementador y el del negocio.

Para lograr cambiar la forma de trabajo de predictiva a adaptativa no basta con decirlo y planear el proyecto de esta forma, es necesario un cambio verdadero en la mentalidad de todas las partes involucradas, un cambio en el que el equipo se sensibilice sobre los problemas que enfrentarán y esté dispuesto a colaborar para llevar el proyecto a buen término.

Finalmente, la experiencia de compartir el presente documento con otras personas que se desenvuelven en el mismo ramo y han tenido experiencias similares o complementarias me pareció enriquecedora, tanto al recibir comentarios de posibles áreas de oportunidad y contestar preguntas que me llevaron a darme cuenta de que tal vez me faltaba ser más explícito en la redacción como al tener oportunidad también de aportar algo de mi experiencia a su trabajo.

BIBLIOGRAFÍA

Dan McCreary, A. K. (2013). *Making Sense of NoSQL: A guide for managers and the rest of us.* Manning Publications.

Fowler, M. (2005, December 13). The new methodology. https://martinfowler.com.

Highsmith, J. (2002). *Agile Software Development Ecosystems.* Addison-Wesley Professional.

IBM Corp. (2016). *Agile@IBM.* Retrieved from IBM training materials: https://agile-ibm.mybluemix.net/

IBM Corp. (2016). *Bluemix overview*. Retrieved from Youtube: https://www.youtube.com/watch?v=BDfMZjFf9s0

IBM Corp. (2017). *What is Bluemix*. Retrieved from IBM Bluemix Docs: https://console.ng.bluemix.net/docs/overview/whatisbluemix.html#bluemixoverview

Kawaguchi, Y. (2013). *La historia de Lean y Agile: Desde Deming hasta Lean Startup.* Retrieved from slideshare.net: https://es.slideshare.net/leansight/la-historia-de-lean-y-agile-desde-deming-hasta-lean-startup

Kent Beck, M. B. (2001). *Agile Manifesto*. Retrieved from agilemanifesto.org: http://agilemanifesto.org/

Larman, C. (2003). *Agile and Iterative Development: A Manager's Guide.* Addison-Wesley Professional.

Schwaber, K. (2004). *Agile Project Management with Scrum.* Microsoft Press.

ScrumAlliance. (2016). *Learn about Scrum*. Retrieved from ScrumAlliance.org: https://www.scrumalliance.org/why-scrum