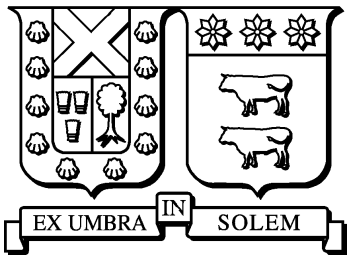
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA

SANTIAGO – CHILE



"PLATAFORMA DE REPORTERÍA AUTOMATIZADA PARA GESTIÓN DE ESFUERZOS EN PROYECTOS DE DESARROLLO DE SOFTWARE BASADOS EN METODOLOGÍAS ÁGILES"

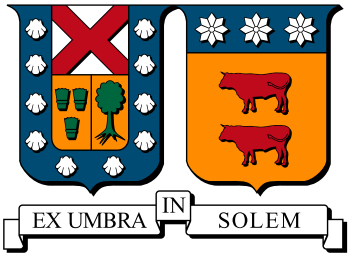
SEBASTIÁN GONZÁLEZ MARDONES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: CECILIA REYES COBARRUBIAS

ABRIL 2020

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



"PLATAFORMA DE REPORTERÍA AUTOMATIZADA PARA GESTIÓN DE ESFUERZOS EN PROYECTOS DE DESARROLLO DE SOFTWARE BASADOS EN METODOLOGÍAS ÁGILES"

SEBASTIÁN GONZÁLEZ MARDONES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: CECILIA REYES COBARRUBIAS
PROFESOR CORREFERENTE: PROFESOR CORREFERENTE

ABRIL 2020

Agradecimientos

Resumen

Abstract

Índice de Contenidos

Índice de cuadros

Índice de figuras

Glosario

Introducción

Capítulo 1

Definición del Problema

1.1. Contexto

Ágil, o *agile* en inglés, se ha destacado en las últimas dos decádas por ser una metodología que promueve una manera diferente de pensar y actuar para gestionar proyectos. Desde su aparición formal en sociedad con la publicación del Maniefiesto Ágil en 2001 hasta hoy, se ha transformado en una importante alternativa (la única alternativa en algunos casos) para gestionar todo tipo de proyectos, no solamente aquellos relacionados con informática [4].

En particular, Agile ha tenido una recepción enorme dentro del mundo de la ingeniería y el desarrollo de software, debido principalmente a principios que, según Steve Denning, miembro del directorio de Scrum Alliance, aportan con guías sistemáticas para crear productos mejores, rápidos, convenientes y personalizados [10], en contraste a la anterior metodología preferida: cascada, o *waterfall model* en inglés.

Principios básicos de la filosofía y desarrollo ágil son [6]:

- énfasis en las personas y relaciones sobre los procesos y herramientas
- énfasis en software que funcione sobre una documentación amplia
- énfasis en colaboración activa por sobre los acuerdos en contratos

- énfasis en un enfoque reaccionario al cambio por la adherencia estricta a un plan ya definido

Son estos principios los que han llevado a metodologías ágiles como Scrum a ser reconocidas como filosofías exitosas[4].

Sin embargo, como toda metodología o decisión tomada en ingeniería, existen trade-offs: concesiones que uno entrega para obtener más o mejores condiciones en otras áreas. Y son justamente los puntos de la derecha de los principios anteriormente expuestos los que más complican la gestión de un proyecto ágil.

Las principales debilidades de un filosofía ágil son la falta de buena documentación, énfasis en funcionalidad sobre usabilidad[15], falta de predictibilidad, mayor esfuerzo de los miembros de los equipos de trabajo, pobre planificación de recursos[5], y una tendencia a perder de vista las necesidades reales del proyecto por sobre necesidades ficticias.

Particularmente, los últimos tres puntos pueden re-interpretarse como una falta (o falla) en la transparencia. Esta falta de transparencia se presenta en la comunicación entre cliente y equipo, principalmente para definir las funcionalidades que realmente se necesitan implementar y en la estimación de recursos requeridos para su desarrollo. La constante respuesta al cambio de Agile y tambien la flexibilidad que posee tienden, en el mediano y largo plazo, a perder el énfasis en lo más importante, el core del proyecto. Esto puede ocasionar una mala asignación de los esfuerzos de desarrollo, trabajando en funcionalidades que aportan poco valor o no son válidas con un usuario/cliente final. Además, un mal plan de recursos suele resultar en la toma de malas decisiones, en el abandono de buenas prácticas, y en una pérdida en la calidad.

Estas debilidades propias de la metodología ágil han sido experimentadas de primera mano, por la empresa de software a medida y boutique tecnológica Nursoft [1] a lo largo de múltiples de sus proyectos. El presente trabajo se realizará bajo el alero de esta empresa, y la solución propuesta será parte cotidiana de las labores de sus ingenieros de software, diseñadores, jefes de proyecto y del Head of Technology de Nursoft.

1.2. Identificación del problema

La transparencia aparece entonces como una de las más grandes amenazas a un proyecto de filosofía *Agile*. Una de sus grandes fortalezas, la aceptación y reconocimiento de la incertidumbre inherente a la naturaleza de un proyecto de software, termina siendo una gran piedra en el zapato. La misma incertidumbre que genera un énfasis de trabajo enfocado en resolver solamente lo necesario para continuar con la siguiente etapa, produce una falta de predictibilidad que siempre complica al presupuesto, a los tiempos reservados para el proyecto y a las expectativas del cliente de obtener lo que necesita.

Con este contexto, se pueden identificar problemas para los siguientes actores:

Clientes Por lo general, se produce una situación en donde el jefe de proyecto y el equipo de desarrollo son los que pueden generar una predicción, en cuanto a costos y tiempos para nuevos requerimientos, que se ajusta mejor a la realidad que los plazos del mismo cliente. Esto es debido a que los ciclos de desarrollo han ido dejando cada vez más experiencia en el equipo sobre cómo gestionar los recursos para nuevas funcionalidades. Se genera una asimetría de conocimiento de gestión y de proyección entre clientes y equipo de desarrollo.

Nursoft Resulta interesante para Nursoft centralizar este conocimiento con respecto a los esfuerzos, principalmente por tres razones importantes:

- Cada proyecto, una vez finalizado, se queda con toda la experiencia que generó en términos de costo del esfuerzo. Cualquier conocimiento derivado sobre la naturaleza del proyecto versus los esfuerzos invertidos en distintas funcionalidades implementadas por el equipo desaparece, quedando únicamente en la memoria de los integrantes del equipo.
- La categorización de los proyectos que son aceptados por Nursoft es difícil de acotar: desde proyectos relacionados con procesos de minería, pesca, medicina preventiva y telemedicina hasta inversión, educación, investigación y manejo de conflictos internacionales, por nombrar algunos.

Sin embargo, Nursoft se aproxima a todos los proyectos con el mismo proceso, y en particular, los procesos de Planificación, Desarrollo y Entrega son idénticos y homologables entre proyectos. Actualmente no se cuenta con data formalizada de gestión de esfuerzos para ninguna tipificación de proyecto, y por lo tanto, Nursoft se enfrenta a los mismos problemas, una y otra vez al iniciar un proyecto nuevo.

- Nursoft cuenta con un par de trabajos[13][12] que hacen, o pronto podrían hacer, uso de información derivada del análisis de reportes. Es primordial para la empresa mantener todos estos trabajos en sintonía y cercanía, de manera que futuros trabajos de título o trabajos, internos se conecten fácilmente a esta red de información.

Equipos de desarrollo Antes del presente trabajo, se ha intentado formalizar el proceso de la gestión de esfuerzos dentro de los equipos, no logrando el éxito esperado por diversos motivos. El proceso de registrar esfuerzos, no otorga ventajas inmediatamente visibles a los miembros de los equipos, de la manera que sí lo hace para los roles más administrativos y gerenciales. De hecho al contrario, se ha argumentado (acerca de los procesos antiguos), que el registrar el trabajo del día a día toma una parte, menor pero no despreciable, de la jornada diaria.

Este ánalisis de los problemas decanta en las siguientes preguntas clave para el desarrollo de este trabajo:

- ¿Cómo hacer más transparente para el cliente el proceso de desarrollo de software y las variables relacionadas con los esfuerzos invertidos en este proceso?
- ¿Cómo se traspasa el conocimiento de gestión obtenido por el equipo, no sólo al cliente, sino que también a la empresa?
- ¿Cómo se combinan exitosamente las necesidades de la gestión con una experiencia óptima para los equipos de trabajo?

1.3. Objetivos de la Solución

1.3.1. Objetivo General

Implementar una plataforma que permita tanto el fácil registro como una completa gestión de esfuerzos de tiempo para los proyectos de desarrollo de software en la empresa Nursoft.

1.3.2. Objetivos Específicos

- Entregar la capacidad de herramientas básicas de Business Intelligence ara gestionar los reportes de esfuerzo de tiempo de los equipos de desarrollo.
- Permitir la visualización de los esfuerzos de manera transparente, sencilla y automatizada al cliente.
- Garantizar la flexibilidad y modularidad como requerimientos no funcionales para permitir una sencila comunicación de la plataforma con futuros proyectos internos de Nursoft.
- Generar un proceso eficiente y sencillo que permita a los equipos de desarrollo el reporte de esfuerzos sin que impacte fuertemente en sus tiempo de trabajo.

1.4. Impactos y alcances

1.4.1. Impactos

Tomando en cuenta las preguntas planteadas en la sección Identificación del Problema, a continuación se describe el impacto que la solución propuesta tiene en la organización:

- ¿Cómo hacer el proceso de desarrollo de software más transparente al cliente?

La manera de trabajar de Nursoft desde el punto de vista comercial es a través de planes de equipo. El cliente paga por un equipo de ciertas características y tamaño, lo que se traduce en horas efectivas de trabajo. Por ejemplo, un proyecto del último año contempló la construcción de una plataforma de *e-learning*[1] al estilo de Coursera[2]. El cliente contrató un plan de equipo conformado por un diseñador full-time, un ingeniero civil full-time, un ingeniero civil part-time y una hueste de otros profesionales: *DevOps*[2], ingeniero de aseguramiento de calidad, arquitecto de software y asesoría comercial. Esto se traduce directamente a un número de horas mensuales por contrato.

Nursoft entiende como transparencia informar al cliente no sólo cuántas horas se inviertieron en el proyecto, sino que *en qué* se inviertieron estas horas, con detalles que pueden ser por funcionalidad/historia de usuario o por tipo de trabajo. Adicionalmente, Nursoft también informa a sus clientes sobre la inversión en *horas complementarias*[3].

Con una plataforma dedicada al reporte y gestión de esfuerzos, este problema se reduce a agregar reportes del proyecto según filtros deseados, bajo la demanda del cliente y de manera automatizada.

- ¿Cómo se traspasa el conocimiento de gestión obtenido por el equipo, no sólo al cliente, sino que también a la empresa?

  El conocimiento de gestión aprendido durante la vida de un proyecto, se refleja directamente en los esfuerzos, y especialmente *en qué tan fielmente* se registran.

  En el contexto de un proyecto particular, teniendo el agregado de esfuerzos totales invertidos en cierto requerimiento, permiten tanto al Cliente como al Jefe de Proyecto contar con información histórica y fidedigna cuando se planifiquen requerimientos similares, entregando inmediatamente una mejor estimación de tiempos y plazos.

  En el contexto de la empresa Nursoft, contar con un histórico de esfuerzos no sólo de historias de usuarios y entregas de versiones, sino que de macro-procesos[4] permite pronosticar y planificar con mayor información y eficacia variables como plazos generales

[1] De *electronic-learning*, se refiere a la educación o capacitación a través de internet.

[2] Si bien la academia y la industria aún pelean por encontrar una definición común, se refiere a un conjunto de prácticas de desarrollo de software y de adminsitración de sistemas con el fin de levantar y automatizar infraestructura tecnológica, ambientes de desarrollo y producción, y entrega continua de valor.

de proyectos, equipos, capacidad de equipos (contrataciones nuevas) y oportunidades de ventas.

Cabe además insistir que Nursoft seguirá seguirá trabajando sobre esta plataforma, extendiéndola a otros casos de uso y en particular sobre la mina de oro que son los esfuerzos, aplicando, por ejemplo, técnicas avanzadas de aprendizaje de máquinas o post-procesamiento, generando y descubriendo nuevos conocimientos sobre sus procesos. Por lo mismo, la solución propuesta debe entregar buena flexibilidad, rendimiento y una alta reutilización de funcionalidades. Según el trabajo de Diego Plaza[14], una arquitectura basada en servicios entrega los requerimientos no funcionales requeridos para la necesidad de la empresa.

- ¿Cómo se combinan exitosamente las necesidades de la gestión con una experiencia óptima para los equipos de trabajo?

  Hasta este punto se tienen claras las necesidades y ventajas para la mirada gerencial de esta propuesta. A través de un proceso de levantamiento de necesidades desde los equipos, tanto desde las funcionalidades como de la UX[5], la solución propuesta apunta a ser asimilada dentro de la rutina de trabajo, no solamente para registrar esfuerzos sino para que entregue valor diariamente a los equipos.

1.4.2. Alcances

La solución permite la posibilidad de una especie de ecosistema para aplicativos relacionados con la gestión de esfuerzos. Sin embargo, este ecosistema potencial escapa del contexto de este trabajo, más allá de que ya existan trabajos funcionales en Nursoft relacionados con análisis de reportes de esfuerzos.

Las intenciones centrales del presente trabajo son primero, armar la base de dicho ecosistema

[3] Se refiere a trabajo que no es efectivo, no es cobrable al cliente, pero crucial para una correcta gestión de un proyecto ágil: gestión, reuniones con clientes, presentaciones, *sprint reviews*, *daily meetings*, *weekly meetings*, etc.

[4] Los macro-procesos de Nursoft incluyen Venta, Planificación, Montaje Inicial, Desarrollo y Post-Venta.

[5] de ***U**ser e**X**perience*, se refiere a todos los aspectos de la interacción de un usuario con un producto, servicio o una compañia.

para que trabajos posteriores se alimenten de él, y segundo, entregar una primera versión de un informe automatizado de gestión de esfuerzos para el cliente. Nursoft se aproxima a este trabajo como un punto de partida o MVP[6], puntapié inicial para un proceso iterativo incremental, cuya segunda versión aún no termina por decidirse.

[6]de ***M**inimum **V**iable **P**roduct*, se refiere a la iteración mínima de un producto o servicio, que permite tanto entregar valor como validar hipótesis iniciales.

Capítulo 2

Marco Teórico

2.1. Nursoft

Pequeña introducción contextualizando la metodología de trabajo de Nursoft, la necesidad de reporar el trabajo. Un repaso por los intentos anteriores desde Nursoft, para gestionar el reporte de esfuerzo, y de generación de informes agregados de estos esfuerzos desde Nursoft hacia el cliente.

Habla de la historia de Nursoft

2.2. Metolodgía de trabajo

Nursoft se ha desenvuelto a lo largo de su vida en más de 20 proyectos de desarrollo de software. Los rubros de cada proyecto se extienden desde la minería, pasando por *retail*, industria agropecuaria, mercado inmobiliario, *e-learning* hasta en tele-medicina.

Una mayoría importante de los proyectos caen en lo que es **desarrollo de software clásico**, y el resto son entendidos como **consultorías**: estos son proyectos donde el cliente está interesado inicialmente en conocer la factibilidad, tanto comercial como técnica, de alguna

propuesta o solución tecnológica generada por terceros o ellos mismos.

El primer grupo de proyectos se puede ceñir a una estructura informal de procesos como la siguiente:

- Cliente presenta su necesidad/problema
- Nursoft indaga el estado actual del cliente para contextualizar la necesidad/problema
- Nursoft propone una solución
- Se realiza la planificación (junto con diseños necesarios)
- Se realiza la implementación
- Se realizan entregas periódicas

Por otro lado, las consultorías se ciñen a la siguiente estructura de procesos:

- Cliente presenta su necesidad/problema junto con su propuesta
- Nursoft indaga el estado actual del cliente para contextualizar la necesidad/problema y la propuesta entregada
- Nursoft propone una contra-propuesta o modifica la propuesta entregada
- Nursoft valida la factibilidad comercial
- Nursoft valida la factibilidad técnica

La naturaleza diversa de los proyectos, tanto en dominio de conocimiento como en los procesos envueltos para llevarlos a cabo exitosamente, promueven una metodología flexible, pero bien definida; modularizada y capaz de adaptarse a las diferencias de cada proyecto. Esta metodología se presenta en la Figura 2.1.

Figura 2.1: Diagrama de macroprocesos para proyecto de desarrollo de software

La estructura general es adaptable a todo tipo de proyecto que Nursoft ofrezca, a modo de ejemplo, la Figura 2.2 ejemplifica la metodología de trabajo para un proyecto de consultoría.

Figura 2.2: Diagrama de macroprocesos para proyecto de consultoría

A continuación, se contextualiza brevemente cada macroproceso.

2.2.1. Desarrollo de negocio

El desarrollo de negocio en Nursoft, es el proceso que en una empresa más tradicional se llamaría Venta. La diferencia es sútil, puesto que Nursoft deja en claro que no vende software como un ítem, vende el desarrollo de software como una experiencia exclusiva y un servicio altamente flexible y con atención extra al detalle. El desarrollo de un negocio implica necesariamente crear una instancia donde ambas partes busquen la mejor parte de trabajar en conjunto.

En esta etapa se nivelan las expectativas de ambas partes. También se mide la compatibilidad del cliente con la empresa, con un potencial escenario donde el proyecto no sea aceptado por Nursoft.

Los puntos de mayor importancia de este macroproceso son el levantamiento de necesidades

(una versión preliminar de requerimientos), la creación de la propuesta económica y tecnológica y la firma de contrato. Para el caso de consultorías, el levantamiento de necesidades se posterga para un macroproceso posterior.

2.2.2. Estrategia

Si la fase de **Desarrollo de negocios** fue éxitosa, este proceso sigue en el ciclo de vida de un proyecto.

En Nursoft se entiende estrategia como la define Chandler[8]:

> Estrategia es la determinación de las metas y objetivos de largo plazo de la empresa, y la adopción de caminos de acción y de asignación de recursos para alcanzar dichas metas...

En este macroproceso se trabajan cuatro pilares en pos de una visión macro del proyecto:

- El cliente
- El negocio del cliente
- Los usuarios
- El software

En otras palabras, se define formalmente cuál es el problema o la necesidad del cliente, contextualizada a elementos internos y externos.

En términos prácticos, en esta etapa participa tanto el departamento comercial como el departamento de diseño y experiencia de usuario (UX). El primer departamento trabaja para definir directrices de planificación estratégica a la empresa del cliente (si así lo lo requiera), realizar estudios financieros para analizar los impactos en el negocio que tendrá el proyecto. Al mismo tiempo, el departamento de UX investiga bajo que tipo de interacciones las

hipótesis comerciales del cliente son válidas. Para esto se utilizan diversas metodologías de investigación del usuario: focus groups, entrevistas y encuestas.

Este proceso termina con un entregable: la creación de la protopersona[7], resultado que tendrá un rol protagónico en el siguiente macroproceso.

2.2.3. Planificación

La **planificación** es el proceso en el cuál se responde qué solución se dará al problema definido en la **estrategia**.

Esta solución se propone, se investiga y se argumenta tanto desde lo técnico hasta de lo visual.

En su defecto, la planificación cuenta con cuatro fases:

- **Propuesta de solución inicial** En la primera propuesta se realizan las mitigaciones técnicas necesarias, y según se necesite se desarrollan los Diagramas de procesos
- **Desarrollo de marca** En donde Diseño realiza benchmarking e investigación de referentes, el *naming*, la paleta de colores y tipografía y finalmente la creación de marca. Esta etapa es opcional, siempre se le sugiere al cliente y sólo se realiza bajo su mandato.
- **Diseño y experiencia de usuario** En donde aparecen los entregables más importantes para la posterior etapa de desarrollo. Diseño genera el flujo de navegación, *wireframes*[8] preliminares y el *basic design system*[9]. Estos son validados a través de pruebas de usabilidad con usuarios *target*. El entregable final de esta etapa es un prototipo no funcinal hacia el cliente.
- **Propuesta de solucion técnica** En este último paso, los requerimientos son transformados en Historias de usuarios. Las historias se estiman utilizando la metodología de

[7] La proto-persona es el resultado de contrastar las visiones y expectativas de un usuario ideal por parte de los *stakeholders* con las expectativas de potenciales usuarios reales que coincidan con ciertos criterios definidos de los *stakeholders*, por ejemplo rango etario o nivel socioeconómico.

Planning Poker[10]. Utilizando como insumos los entregables de la propuesta inicial, se construye y presenta la propuesta de Arquitectura.

2.2.4. Infraestructura

El proceso de Infraestructura es uno de los procesos más cortos de todo el flujo de desarrollo de un proyecto de software en Nursoft. Sus entregables y responsabilidades son casi totalmente internos.

En este proceso se levanta la infraestructura tecnológica necesaria para el desarrollo del proyecto y la mantención del mismo. Tomando como insumo la propuesta aceptada de Arquitectura, el equipo de Operaciones configura los servicios que serán utilizados a través de plantillas automáticas. También se configuran los procesos automatizados de integración y despliegue continuo (*CI* y *CD*, respectivamente).

Finalmente, se configura el entorno de *staging*, una versión de prueba de la plataforma final, utilizada para los subprocesos de *Quality Assurance*[11] y de revisión de Diseño.

2.2.5. Desarrollo

El proceso de desarrollo en Nursoft es un proceso iterativo, siguiendo una versión propia y modificada de *SCRUM*.

La unidad mínima de trabajo en este proceso es el *sprint*: es un período donde se trabaja en pos de un objetivo de negocio claro. En este período el equipo completa una serie de funcionalidades acordadas al comienzo del *sprint* y relacionadas al objetivo.

La definición del objetivo generalmente quedo a criterio del equipo y su *Project Manager*,

[8] Un *wireframe* es una esquemática o guía visual que muestra el esqueleto de una estructura.[7]

[9] Un *basic design system* es la única fuente de verdad que agrupa todos los elementos que permitirán al equipo identificar e implementar un producto, a nivel visual.[11]

[10] *Planning Poker* es una técnica lúdica basada en consenso para estimar esfuerzo necesario para implementar una funcionalidad en desarrollo de software[9]

pero una regla fundamental es que cada *sprint* debe generar valor para el cliente y/o el usuario.

Cada sprint se divide en tres etapas, como se muestra en la Figura 2.3

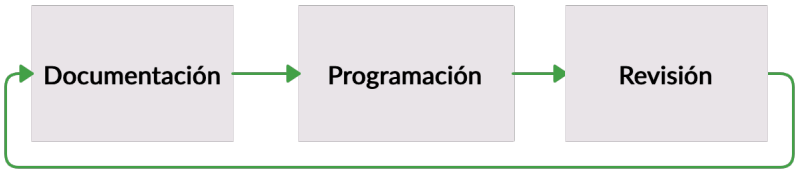


Figura 2.3: Etapas de un sprint

- **Documentación** Cómo se mencionó anteriormente, al comienzo de cada *sprint* se define el objetivo de negocio, y con éste, las funcionalidades a implementar. Se discute y actualiza o completa la documentación de las historias de usuario correspondientes.

  ¿Por qué revisitar la documentación en cada sprint si ya se definieron al final de la Planificación?

  Es común en la experiencia de Nursoft vivir de primera fuente uno de los fundamentos de la metodología ágil en cada *sprint* mencionado en el capítulo anterior:

  > Énfasis en software que funcione sobre una documentación amplia

  Esto es particularmente cierto para proyectos de desarrollo de software a medida. La documentación inicial fue útil para entender el alcance preliminar de la funcionalidad (además de cómo la arquitectura facilita su ejecución), pero a medida que el proyecto avanza, el cliente mismo va mejorando su conocimiento del contexto para cada funcionalidad, y al mismo tiempo refina sus expectativas de éste: plazos, integraciones, prioridad, etc. Por eso es relevante este subproceso de documentación al comienzo de cada *sprint*: permite afinar al conocimiento disponible de todos los *stakeholders* para la funcionalidad.

- **Programacion** Puntos relevantes para este subproceso son:

[11] *Quality Assurance* es un término que engloba manera de prevenir defectos o fallas en un producto, con el enfoque en entregar confianza de que los requerimientos serán cumplidos.[3]

- La gestion del código en todos los equipos se hace a través de una versión modificada de *Gitflow*, un flujo de trabajo colaborativo para software de herramientas de control de versiones, como Git.
- El entregable de este subproceso es hacer el despliegue del código al ambiente de *staging*.
- En la gran mayoría de los casos, el desarrollo de funcionalidades implica la generación de pruebas para este mismo. Particularmente, la mayoría de los equipos siguen el proceso de *Test Driven Development* o **TDD**.

- **Revisión**

  El proceso de revisión se constituye de dos tipos de revisiones: la manual y la automatizada.

  La **revisión automatizada** se realiza de manera continua durante el subproceso de programación, a través del proceso de Integración continua. Con cada sección de código enviada al repositorio central, se gatillan las pruebas automatizadas generadas en TDD. Si y sólo si las pruebas pasan, la sección de código enviada se mezcla con el repositorio.

  La **revision manual** cuenta con tres fases: *code review*, revisiones de QA y revisión de diseño. Las revisiones de QA se ejecutan cuando las funcionalidades fueron completadas, y se llevan a cabo en el ambiente de *staging*. Su principal aporte es verificar una calidad mínima de la funcionalidad. Se realizan a través de un conjunto de casos de prueba o escenarios de casos de uso, generados a partir de la documentación contextual de la historia de usuario. También son consideradas partes de la documentación.

  El proceso de *code review* se realiza idealmente posterior a la revisión de QA, pues se busca calificar la calidad del código y la legibilidad entre pares.

  Finalmente, las revisiones de diseño aplican donde se haya generado una interfaz visual, y verifican que haya una fiel correlación con los *mockups* o *wireframes*, donde corresponda, y que la experiencia de usuario con la nueva interfaz sea óptima.

  Para poder continuar al siguiente proceso, un *sprint* debe aprobar sin fallas severas todas las revisiones anteriores.

2.2.6. Entrega

Este es el proceso mas movedizo en el proceso de desarrollo de Nursoft, puesto que se ha modificado muchas veces durante la historia de la empresa. Esto ocure debido a que como se ha mencionado en anteriores ocasiones, Nursoft busca que la experiencia del desarrollo de software sea exclusiva, por lo tanto la entrega es una oportunidad para entregar un valor agregado al servicio.

Qué es exactamente valor para el cliente (obviando las funcionalidades) es un tema que Nursoft ha investigado y probado desde sus inicios, y a medida que su cartera de clientes ha ido cambiando a empresas cada vez más grandes, esta discusión se ha hecho más sútil.

Históricamente en este proceso se han agregado entregables como manuales de usuarios con fuerte énfasis en lo gráfico, reuniones más extensas para re-discutir requerimientos, discusión de post-mortem e incluso entregas de estadísticas. Ninguna de estas iniciativas se ha mantenido en el tiempo, porque no todos los clientes validan la necesidad de estos entregables, pero Nursoft continuamente sigue generando ideas para agregar valor a la experiencia.

Desde el punto de vista técnico, durante el establecimiento inicial del *sprint* se establecen qué funcionalidades se completarán al final de éste. Esto se logra a través del versionamiento semántico, o la creación de versiones de entrega.

El equipo acuerda cuántas versiones se entregarán, y cuales serán sus codigos indentificadores y su nombre. Un *sprint* puede generar muchas versiones y una misma versión puede necesitar más de un sprint para ser completada. La naturaleza de las funcionalidades definirá de manera exclusiva el código de la versión, mediante versionamiento semántico.

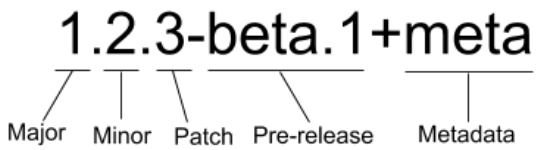


Figura 2.4: Diagrama de una version en versionamiento semántico.

Considere la Figura 2.4. Si la versión sólo arregla errores de la versión anterior o alguna

previa, entonces la nueva versión debe ser la misma que la inmediatamente anterior, pero aumentando en una unidad el término *patch*.

Si la versión introduce nuevas funcionalidades, entonces el identificador de la nueva versión debe ser el mimso que la inmediatamente anterior, pero aumentando en una unidad el término `minor`.

Finalmente, si la versión introduce un cambio que no es retro-compatible que la versión anterior, entonces se aumenta en una unidad el término `major`. Esto ocurre pocas veces, y generalmente implica cambios a nivel estructural del proyecto, ya sea en su *API*, en su *stack* de tecnologías o algún cambio de requerimientos severo. Los términos *pre-release* y *metadata* son tan poco utilizados que pueden ser omitidos para efectos de esta memoria.

La primera entrega del proyecto siempre debe ser `0.1.0`.

Una vez que la entrega ha sido defini, implementada y testeada, están todas las condiciones para hacer la entrega. La entrega se completa cuando el código ha sido desplegado al ambiente *beta* y se ha notificado al cliente.

El ambiente *beta* es un ambiente de pre-producción: todas sus configuraciones son como las del ambiente de *production*, pero la interacción es sólo con el cliente y algunos usuarios especiales externos (generalmente otros *stakeholders*). El ambiente de *production* por otro lado, es en dónde los usuarios finales interactúan con la plataforma.

Al momento de la entrega pueden suceder dos escenarios: si la funcionalidad ejecuta fielmente los requerimientos y el cliente no tiene observaciones de ningún tipo, entonces el código de la versión se despliega al ambiente de *production* y se finaliza el *sprint*. Si la funcionalidad falla debido a casos que no se contemplaron, o existen observaciones de requerimientos por parte del cliente, entonces la versión se extiende al *sprint* siguiente, donde volverá a pasar por los ciclos de Desarrollo y Entrega nuevamente.

2.2.7. Post-servicio

Esta etapa es la última en el ciclo de vida del proyecto e involucra procesos para clientes que terminan la relación laboral con Nursoft como para aquellos que desean continuarla.

En el caso de los primeros, el código fuente e instrucciones para replicar el despliegue de la plataforma tal cual lo hacía Nursoft son entregadas al cliente.

En el caso de los segundos, aparece el concepto de plan de mantención para proyectos. Este plan se desarrolla en conjunto del área comercial y del área de operaciones y se divide en tres:

- **Infraestructura** Se aloja y administra la plataforma u aplicación desde las cuentas de Nursoft.
- **Garantía** Se realiza mantención correctiva ante desperfectos.
- **Mejora continua** Se desarrollan pequeñas funcionalidades y mejoras.

A diferencia del macroproceso principal, esta fase replica la fase de desarrollo de manera muy acotada. Por ejemplo la toma de requerimientos nuevos o mejoras se acuerda por un hito informal como una llamada telefónica.

Este proceso tambíen busca potenciar el concepto de experiencia para el cliente, y ha ido cambiando en el tiempo en relación al feedback de los clientes.

2.3. Reporte

2.4. Plataforma

2.4.1. Arquitectura

Donde se define que son los microservicios, sus ventajas frente a otras arquitecturas y se argumenta por qué se seleccionaron para este proyecto

2.4.2. Experiencia

Donde brevemente se define que es la experiencia de usuario, sus orígenes, por qué es importante, cómo se levantaron requerimientos para este proyecto, y cómo afectó finalmente al proyecto.

2.4.3. Infraestructura

Donde brevemente se define que es una Infraestructura tecnológica, y se definen cuáles herramientas se utilizaron para el proyecto y por qué fueron elegidas.

2.4.4. Desarrollo

Donde se describen los frameworks utilizados para los servicios de la plataforma.

Capítulo 3

Propuesta

Capítulo 4

Implementación

Conclusiones

Bibliografía

[1] https://nursoft.cl.

[2] https://www.coursera.org/.

[3] *ISO 9000:2005, Clause 3.2.11*.

[4] *The state of SCRUM Report*. The Scrum Alliance, 2015.

[5] *What are the disadvantages of agile?*, 2019. https://leankit.com/learn/agile/what-are-the-disadvantages-of-agile/.

[6] Beck, Kent, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries *y cols.*: *The agile manifesto*, 2001.

[7] Brown, Daniel M: *Communicating design*. New Riders, 2011.

[8] Chandler, Alfred D: *Strategy and structure: Chapters in the History of the American Industrial Enterprise*. MIT Press, 1962.

[9] Cohn, Mike: *Agile estimating and planning*. Prentice Hall PTR, 2012.

[10] Denning, Steve: *Agile: The World's Most Popular Innovation Engine*, 2015. https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine.

[11] Kholmatova, Alla: *Design Systems: A Practical Guide For Creating Design Languages For Digital Products*. Smashing Magazine, 1ª edición, 2017.

[12] Mas'Ad, Rafik, Ricardo Ñanculef y Hernán Astudillo: *BlackSheep: Dynamic Effort Estimation in Agile Software Development using Machine Learning*. En *CIbSE*. Curran Associates, 2019.

[13] Morales, Felipe: *Estimación de esfuerzo en proyectos de software a partir de historias de usuario*. Memoria, Universidad Técnica Federico Santa María, 2019.

[14] Plaza, Diego: *Aplicación basada en SOA para establecer buenas prácticas de desarrollo en sistemas web*. Memoria, Universidad Técnica Federico Santa María, 2015.

[15] Satria, Deki, Dana Sensuse y Handrie Noprisson: *A systematic literature review of the improved agile software development*. páginas 94–99, Octubre 2017.