UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

Profesor Patrocinante:

Yussef Farrán Leiva.

Comisión:

César González C.
Gonzalo Rojas D.

Metodología Lean-PMI para desarrollo de proyectos de software

Informe de Memoria de Título para optar por el título de: Ingeniero Civil Informático

Concepción, 27 de abril de 2017

Juan Manuel Sanhueza Gómez

Índice

Abstract

Si bien la perspectiva tradicional de desarrollo con énfasis en la planificación previa y reticencia a la modificación de requerimientos se encuentra cada día más en desuso, la idea de implementar un método ágil puede ser una tarea que choque con una cultura adversa al cambio que prefiera adoptar medidas más tradicionales frente a dificultades en el desarrollo. En este documento se realiza una revisión del método Lean de Poppendieck, generando una sugerencia de implementación validada por el marco de las buenas practicas del PMI y entregando un conjunto set sugerido de cuestionarios para comprobar su implementación, esto con el objetivo de proveer una alternativa más liviana y de menor impacto a otras metodologías ágiles o tradicionales para organizaciones que busquen agilizar o mejorar su proceso de desarrollo.

1. Introducción

El avance de las metodologías de desarrollo de software y sistemas informáticos ha continuado incansablemente desde hace ya varias décadas, sin embargo, todavía existen lugares en donde las prácticas en uso corresponden a métodos antiguos de ingeniería que, si bien pueden funcionar bien para otras áreas de la ingeniería, tienen resultados nefastos en el área de desarrollo de software.

Afortunadamente, el avance y éxito de las metodologías ágiles ha impactado a toda la industria de forma contundente, obligando a muchas empresas a tener que adoptar estas metodologías para poder seguir siendo competitivas en un mercado cada vez más agresivo. Por lo tanto, muchas empresas que antes ocupaban una metodología de cascada o, peor aún, que no contaban con ninguna metodología establecida (llamado en adelante "metodología *ad hoc*") se encuentran en la necesidad de implementar prácticas e ideas ágiles sin tener la cultura o apoyo necesario como para poder sustentar éstas (1). La encuesta realizada por VersionOne en el año 2013 muestra las inquietudes y las razones de fracasos detrás de los intentos de implementación de metodologías ágiles.

Dado que el apoyo por parte de las jefaturas tiende a ser el mayor factor de éxito en la ejecución de un proyecto (2), es natural pensar que si la jefatura se opone a la implementación de metodologías ágiles, entonces intentar realizar un proyecto de forma ágil resultará en un proyecto fallido. Por otra parte, la tasa de fracaso de los proyectos con metodología ad-hoc (3), puede conllevar a implementar una metodología de cascada como respuesta ya que, tanto como

menciona Poppendieck como en la experiencia del autor, la respuesta a dificultades en el desarrollo suele aumentar la formalidad, es decir, aumentar la documentación, congelar los cambios al alcance del proyecto y optar por un modelo de Cascada. Esta decisión, a su vez, reduce la tasa de éxito de los proyectos (4), (2), (3).

Es con estos elementos en mente que nace la motivación para realizar el actual documento. Es decir, ante constantes problemas en el desarrollo de proyectos de software ¿Es realmente conveniente reaccionar por medio de la formalización y el aumento de rigidez del proceso de desarrollo de proyectos de software? Si bien intuitivamente se puede pensar que sí lo es, se ofrece una contrapropuesta: agilizar el proceso de desarrollo. Como se mostrará más adelante, la formalización y "rigidización" de los procesos de desarrollo suelen traer un aumento del riesgo y una menor tasa de éxito en los proyectos junto con un mayor costo general. Sin embargo, ya que esta contra-propuesta es poco intuitiva, con el objetivo de poder convencer a las jefaturas, se realizará la formulación de un proceso de desarrollo para proyectos de software considerando los siguientes requerimientos:

1. Debe ser ágil, tener bajo costo de implementación y ofrecer cierta flexibilidad
2. Debe ser validada con métodos o herramientas más tradicionales (PMBOK en el actual caso)
3. Se deben entregar herramientas para revisar y corroborar su correcta implementación

Finalmente, se agrega que, dado que existen una infinitud de casos posibles bajo los cuales se desarrolla software, el documento actual se centrará en el desarrollo de proyectos (nuevos) de software por parte de un equipo de desarrollo pequeño (no más de 10 personas) con un solo cliente.

2. Discusión Bibliográfica

Existen múltiples metodologías de desarrollo de software, tanto ágiles como tradicionales. A continuación se realizará una pequeña descripción de las más conocidas (1):

Nombre	Tipo
Cascada	Tradicional
Scrum	Ágil
Extreme Programming (XP)	Ágil
Kanban	Ágil
Lean (Poppendieck)	Ágil
PMI	Tradicional

2.1 Cascada

El método de cascada es una metodología tradicional que se caracteriza por realizar las etapas de desarrollo de forma secuencial, idealmente, sin volver atrás durante el transcurso del proyecto (5). Se destaca que, como fue mencionado anteriormente, cuando hay problemas en el desarrollo de proyectos, la tendencia es implementar esta metodología. Generalmente contempla una etapa de Captura de Requerimientos de sistema, Captura de Requerimientos de Software, Análisis, Diseño, Implementación, Testing y Operación. Sin embargo, como fue mencionado anteriormente, trae consigo una serie de grandes riesgos. En primer lugar, realizar cambios a los requerimientos en etapas avanzadas del proyecto conlleva un gran costo. Este riesgo es exacerbado si se considera que el Testing es realizado tardíamente, riesgo reconocido incluso por el autor del paper (5) y por documentos como el Chaos Report (del cual se hablará más adelante).

Este riesgo a cambios tardíos a los requerimientos es más probable si se considera que cascada asume que: el cliente es capaz de articular el problema de forma correcta y fácil de entender, que los detalles pueden ser resueltos en la planificación y no durante la implementación y, finalmente, que el inmenso orden de complejidad de esta planificación no superará las capacidades de los involucrados. (6)

En resumen, la aplicación de esta metodología, si bien es intuitiva, es potencialmente peligrosa para casos en los que existen problemas en el manejo de requerimientos.

2.2 Metodologías Agile

Agile se refiere al conjunto de metodologías de desarrollo centradas en los principios del "Manifiesto Ágil" (7):

> 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.

Este manifiesto, escrito en el año 2001, fue la base para varias metodologías de desarrollo creadas, siendo Scrum la más conocida e utilizada con Extreme Programming como lejano segundo (1).

2.3 Scrum

Scrum (8) describe una serie de prácticas para el desarrollo de software que permiten mejorar la velocidad de desarrollo y la calidad del software producido. A grandes rasgos:

- Requiere tres roles, el "Product Owner", responsable de maximizar el retorno de la inversión del producto realizado, el "Equipo de Desarrollo", el cual es independiente y multidisciplinario y el "ScrumMaster", el cual protege al equipo de interferencia externa y lo guía en el uso de Scrum
- Sprints, es decir, iteraciones fijas con objetivos y tareas definidas.
- Reuniones de planeamiento de iteración, de avance y planificación diaria, de revisión de iteración y de retrospectiva.
- Énfasis en aprender y adaptarse al cambio.

2.4 Extreme Programming (XP)

XP es una metodología de desarrollo de software iterativa y ágil con gran énfasis en la retroalimentación y conocida por tener una gran cantidad de prácticas asociadas (9), entre ellas:

- Desarrollo orientado al testing
- Reuniones diarias de desarrollo
- Movimiento constante de gente para evitar lagunas de conocimiento y cuellos de botella
- Cliente preferiblemente in situ
- Programación a pares
- Integración continua

Si bien XP tiene un menor número de adherentes, las ideas y prácticas que ofrece son interesantes para aquellos que buscan implementar soluciones híbridas

2.5 Metodologías Lean

Las metodologías consideradas Lean son aquellas que toman inspiración de la manufactura Lean de Toyota. Según (10), Lean tiene que ver con una forma de pensar y abordar problemas en la manufactura y está basado en dos pilares: Respeto por la Gente (sean los empleados, los proveedores o los clientes) y Mejora Continua (lo cual incluye la eliminación de basura), además

de una serie de principios y prácticas complementarias a estos dos pilares (tales como los sistemas de pull, preocupación por el flujo de tareas, establecimiento de filosofías de largo plazo, entre otros.)

El éxito de esta filosofía es lo que permitió que Toyota sobrepasara a los gigantes de General Motors y Ford (en el 2008 Toyota fue la mayor empresa de producción de vehículos por ventas y con la mayor ganancia).

Incluso antes de la publicación del libro de Poppendieck, (11) muestra que los principios Lean de Toyota pueden ser aplicados al área de desarrollo de software, minimizando la cantidad de trabajo en transcurso, haciendo visibles los errores de producción, resultando en rápidas mejoras en la calidad y en la productividad. Por otra parte tanto (12) como (13) sostienen desde décadas atrás que los valores de Lean son aplicables y útiles en el desarrollo de software.

Existen varios autores que han escrito sobre Lean, cada uno enumerando una serie de prácticas y valores muchas veces intercambiables. (14) Busca los valores generales de Lean, comparando tanto bibliografía del área de la informática, como de la manufactura. Este análisis es bastante prometedor, pues permitiría unificar el paradigma Lean

2.6 Kanban

Kanban es un método de trabajo el cual, si bien comenzó siendo focalizado en software, sirve para lograr optimización de procesos causando bajo impacto para la empresa y brindando un ritmo sustentable para los trabajadores (15). Es un caso interesante pues es considerado una metodología *Lean* ya que también está inspirada en el *Lean* de Toyota. Como método gira en torno a la tabla Kanban, la cual contiene las tareas en espera, proceso y compleción de un determinado equipo de trabajo, y la búsqueda de la óptima cantidad de tareas siendo realizadas de modo que se maximice la velocidad y calidad del trabajo realizado.

Más específicamente, cada área de trabajo (programación, testing, integración) posee un número máximo de tareas que pueden ser realizadas de forma simultánea. Ya que este número no debe ser superado, se reduce la cantidad de trabajo concurrente y, más de forma más importante, se reduce la cantidad de trabajo parcial o en espera. Esto a su vez conlleva a una necesidad de priorizar y a un aumento de la velocidad a la que se realizan las tareas.

Por otra parte, mapear y transparentar el proceso de trabajo permite visualizar la “basura” que existe en el proceso y ofrece la posibilidad de modificarlo (o negociar una modificación) con argumentos sólidos y datos reales.

Como se verá más adelante, Kanban suele ser combinado con otras metodologías como herramienta de apoyo. Además se agrega que mediante simulaciones (16) demuestra que, al menos para mantención, Kanban trae resultados semejantes a Scrum.

2.7 Lean (Poppendieck)

Lean, como paradigma, se refiere al conjunto de metodologías de desarrollo de software que toman como inspiración la manufactura Lean de Toyota. Además, Lean es un paradigma en el sentido de que no está conformado por prácticas específicas sino por ideas, filosofía aplicada a un proceso de desarrollo con el objetivo de mejorarlo. Este paradigma y algunos elementos que resultan de su aplicación son descritos en el libro de Poppendieck. A gran escala, Lean se basa en la aplicación de los siguientes principios:

1. Eliminar la Basura: Eliminar todo aquello que no entregue valor al cliente incluyendo:
    a. Trabajo parcial
    b. Procesos innecesarios
    c. Features innecesarias
    d. Multitasking
    e. Esperas
    f. Movimiento innecesario
    g. Defectos
    h. Actividades de gestión
2. Amplificar el Conocimiento
3. Decidir lo más tarde posible: Tomar las decisiones cuando exista la mayor cantidad de información disponible
4. Desarrollar lo más rápido posible: Con el objetivo de entregar rápidamente valor al cliente y obtener su retroalimentación
5. Empoderar al equipo: Permitir que el equipo tome sus propias decisiones con las herramientas adecuadas
6. Incluir integridad: Generar un producto con una visión integral y unificada
7. Ver el todo

Estos principios se convierten en potenciales prácticas que pueden ser implementadas por el equipo de desarrollo conforme a sus necesidades. Por ejemplo, del principio de “Amplificar el Conocimiento” se puede deducir la necesidad de retroalimentación y, de esta última, se puede deducir la práctica de realizar iteraciones para presentar los avances y obtener la retroalimentación.

Dado que Lean es, entonces, un paradigma compuesto de un conjunto de ideas, la implementación de Lean radica en adaptar y mejorar el proceso actual de desarrollo, lo cual

significa una barrera de entrada más baja a diferencia de, por ejemplo, XP. Permite, además, adaptarse más fácilmente a la cultura de la empresa y delega al equipo de trabajo la dirección del proceso.

Estas implementaciones parecen ser exitosas, pues, como, (17) encuentra, sobre 30 papers, los resultados son prometedores, entregando menor tiempo de desarrollo, mayor productividad, menor costo y mayor satisfacción del cliente. Como requerimiento, sin embargo, encuentra que es necesaria la cooperación de las jefaturas, la voluntad de cambio y la incompatibilidad con la gestión tradicional de proyectos. Otras implementaciones incluyen (18), (19) y (20).

Por último, en el libro de Poppendieck se mencionan también múltiples prácticas que son deducciones de los principios mencionados anteriormente. Estas prácticas se utilizarán en el proceso sugerido teniendo en cuenta que describen una situación modelo: Un proceso de desarrollo de proyectos de software con un equipo de desarrollo abordando un único proyecto para una única organización. En caso contrario, el proceso sugerido puede no ser aplicable.

2.8 Híbridos

Las prácticas e ideas de las diversas metodologías y filosofías de desarrollo son, muchas veces, compatibles entre sí, permitiendo combinar y crear un método que cumpla las necesidades del equipo y de la organización. Si bien estas combinaciones se dan en la práctica, un ejemplo de una metodología híbrida publicada es Scrumban que combina prácticas de Scrum y Kanban (21). En esta metodología se plantea a Kanban como una base sobre la cual uno puede, o no, colocar prácticas de Scrum según considere necesario o útil.

En la práctica, existen muchas implementaciones híbridas de Agile (generalmente en la forma de Scrum) y Lean (tanto como set de principios Lean como prácticas en forma de Kanban). Como ejemplo, (22), (23), (24), (25) y (26) todas señalan el éxito que trajo seguir los principios Lean y Agile para el desarrollo de software. Esto indica que Scrum y Lean se complementan enormemente, con Scrum entregando herramientas y prácticas y Lean como filosofía que sirve de guía en los cambios futuros no contemplados por Scrum (por ejemplo, decidir qué nuevas prácticas o herramientas adoptar), lo cual es complementado por el trabajo de (14).

Interesantemente (27) habla sobre la radical diferencia de cambiar desde un paradigma considerado formal (CMMI) a uno ágil con Scrum y Lean, obteniendo enormes ganancias.

2.9 Métodos Tradicionales de Gestión de Proyectos:

Existen métodos de desarrollo de proyectos de software provenientes de un paradigma diferente al de los métodos ágiles, un paradigma proveniente de una ingeniería rigurosa, con gran énfasis en la planificación, la formalidad y la rigurosidad. Ejemplos de estos métodos son PMI, CMM, CMMi y

Prince2. Debido a su larga historia y peso en el mundo de desarrollo de proyectos, se utilizará PMI para validar el trabajo realizado.

2.10 PMI

PMI es el marco de referencia contenido en la "Guía de los Fundamentos para la Dirección de Proyectos" (en adelante "*PMBOK*", por sus siglas en inglés) es un libro publicado por el "Project Management Institute" que "proporciona pautas para dirección de proyectos individuales y [que] define conceptos relacionados con la dirección de proyectos" (28). También entrega detalles sobre el ciclo de vida de los proyectos y sus procesos.

Este libro nació a partir del establecimiento de buenas prácticas en la dirección de proyectos, el cual ha estado en constante estado de refinamiento desde que fue escrito por primera vez en el año 1996. Actualmente se encuentra en su quinta edición, con la sexta siendo publicado durante el primer trimestre del año 2017. Adicionalmente, esta quinta edición posee una extensión para el desarrollo de proyectos de software, reconociendo que existen particularidades específicas en esta área (29) y entregando soporte a métodos adaptativos, es decir, iterativos e incrementales.

PMBOK describe los procesos utilizados en el desarrollo de proyectos categorizándolos de dos grandes formas:

- Por grupo de procesos:
    - Procesos de Inicio
    - Procesos de Planificación
    - Procesos de Ejecución
    - Procesos de Monitoreo y Control
    - Procesos de Cierre
- Por área de conocimiento:
    - Gestión de Integración
    - Gestión de Alcance
    - Gestión de Tiempo
    - Gestión de Costos
    - Gestión de Calidad
    - Gestión de Recursos Humanos
    - Gestión de Recursos de Comunicación
    - Gestión de Riesgos
    - Gestión de Adquisiciones
    - Gestión de Interesados

3. Método desarrollado

Dado que el mayor factor para el fracaso en la implementación de una metodología ágil o Lean es la resistencia externa tanto por las jefaturas como por la cultura organizacional (30) (17), se optó por Lean, el cual posee la menor barrera de entrada (Scrum, por ejemplo, requiere de ScrumMasters y XP de prácticas que el equipo de desarrollo puede no preferir como programación a pares). A futuro, si el equipo de trabajo lo considera útil, es posible tomar prácticas o elementos de otras metodologías de desarrollo para complementar la propia.

Cabe repetir que el método desarrollado es para desarrollo de proyectos de software nuevos por parte de un equipo de desarrollo que esté completamente dedicado a esta labor. Esto es en contraposición a, por ejemplo, un equipo de mantención con varios clientes solicitantes, escenario en el cual el uso de iteraciones puede agravar la situación.

Entre Kanban y Lean, Kanban requiere la implementación de una tabla kanban y el establecimiento de trabajos máximos concurrentes por área. Esta última idea, si bien su aporte positivo es respaldado por la bibliografía, puede ser difícil de vender a la jefatura. La ausencia de iteraciones, por otra parte, requiere de un grado de madurez y coordinación al momento de desarrollar un proyecto. Lean por otro lado es fácil de entender para las jefaturas por su cercanía al Lean de Toyota mientras que las prácticas que recomienda son relativamente fáciles de implementar y ayudan al desarrollo del proyecto.

Otro elemento que provee Lean (y Kanban) es la posibilidad de una implementación progresiva (o una constante búsqueda por la perfección, Kaizen). Lean comienza con el proceso actual y lo guía para mejorar la calidad y velocidad de desarrollo, lo cual reduce el impacto que puede tener esta implementación en una empresa con reticencia al cambio.

Lean, al igual que las metodologías ágiles, empodera al equipo de desarrollo, aumentando su productividad y motivación, instándolos a tomar nuevas herramientas o prácticas que mejoren, continuamente, la calidad de su trabajo.

3.1 Sobre proyectos pequeños:

Como el Chaos Report indica, no todos los tipos de proyecto tienen la misma tasa de éxito. La tabla de tamaño-complejidad (Anexo 4) nos indica el riesgo en proyectos dependiendo de ambos factores, destacando que el riesgo aumenta de forma no lineal. De todos los proyectos, son los proyectos considerados pequeños los que tienen mayor tasa de éxito. Esto se puede deber a que son más fáciles de manejar, poseen límites y requerimientos más sencillos y claros, poseen menor interferencia en las jefaturas y, por último, en caso de que el proyecto esté mal definido desde su

concepción, permite un fracaso temprano y económico. Esto es clave si se considera que utilizar calendarizaciones y presupuesto poco realistas es el mayor riesgo durante el proyecto (31).

Debido a esto, se considera para el proceso de desarrollo sugerido que cada proyecto de tamaño mediano o grande sea llevado a cabo por medio de sucesivos proyectos de tamaño pequeño, cada uno con límites y alcance estrictamente claramente definidos.

3.2 Implementación de Lean

Como fue mencionado al inicio del informe, el proceso de desarrollo sugerido se generó para el caso de un equipo de que se encuentre desarrollando un sólo proyecto. En la práctica, existen otros casos, sobre todo para equipos de desarrollo pertenecientes a empresas cuyo rubro no es el desarrollo de software. En estos casos es posible encontrar equipos balanceando múltiples proyectos para múltiples clientes, tanto de desarrollo como de mantención de software. Si bien Lean es aplicable a estos casos (Kanban podría ser utilizado como una interesante herramienta de apoyo), el método desarrollado no es necesariamente óptimo para esas situaciones.

El proceso se encuentra documentado en el Anexo 2, pero a grandes rasgos se basa en las prácticas sugeridas por Poppendieck. Muchas de estas prácticas son similares a las de Scrum, por lo que existe cierto grado de similitud con esta metodología. A grandes rasgos, el proceso desarrollado es un proceso iterativo, incremental, con énfasis en la comunicación cara a cara y retroalimentación rápida. A continuación se muestra un esquema del procedimiento y una breve descripción de los procesos individuales:

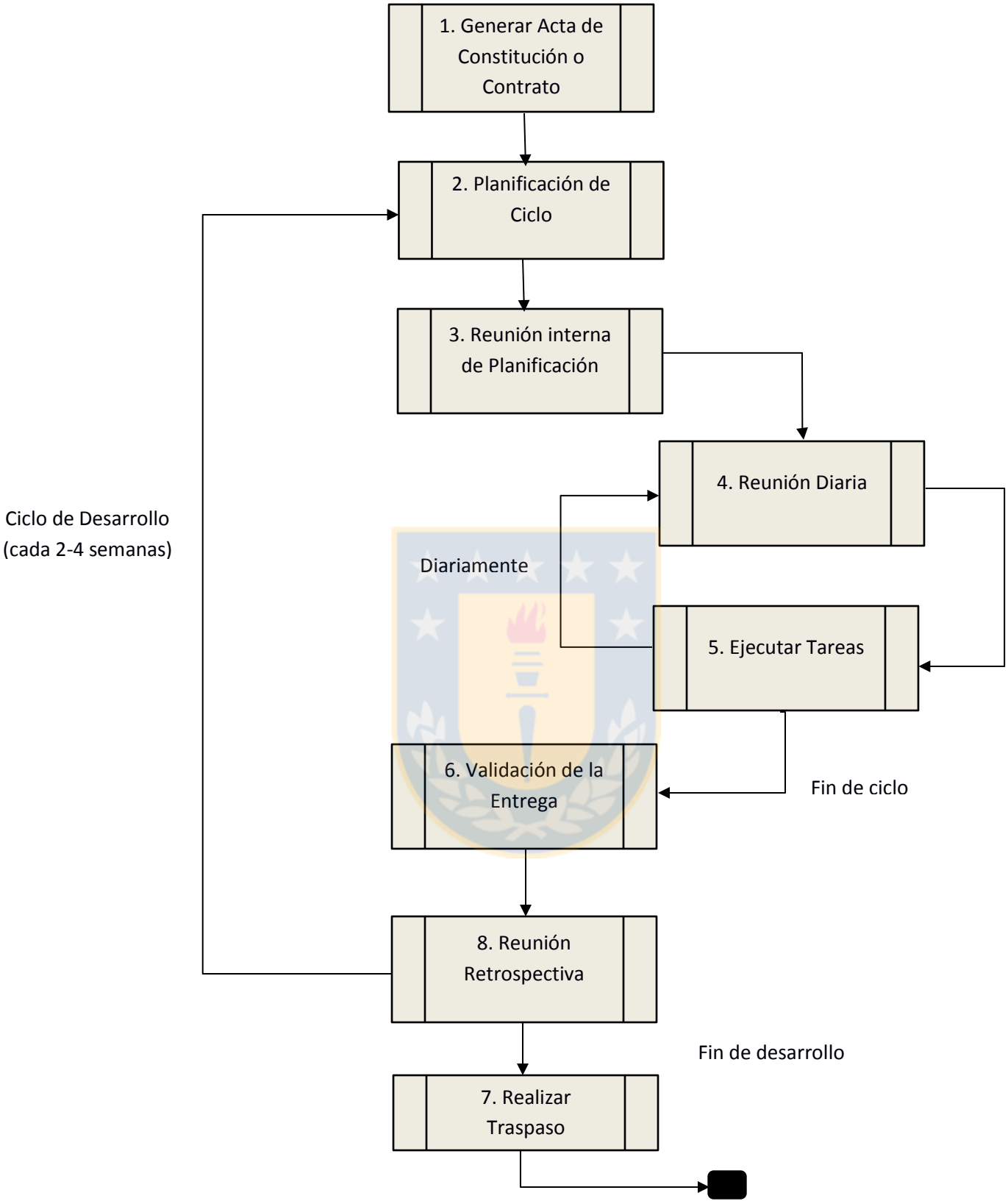


Figura 1: Diagrama del Proceso Sugerido

1. Generar acta de constitución

Se genera el acta o el contrato que da inicio al proyecto, delimitando una versión inicial del alcance, estándares iniciales (tales como procesos adicionales o seguridad) y asegurándose que el cliente, usuario y dueños del proceso accedan trabajar bajo el método delineado. Para esto se utilizan los procesos activos Factores Ambientales de la empresa y los Activos de los Procesos, generando un acta de constitución y listas de procesos adicionales, restricciones, interesados y reuniones programadas.

2. Planificación de Ciclo

Reunión en la que se negocian las tareas que se realizarán durante la iteración, reuniéndose con el cliente y el dueño del proceso e, idealmente, el o los usuarios. Basándose en los documentos del proceso anterior, apoyándose en entrevistas, cuestionarios, prototipos y desarrolladores expertos, se genera una lista priorizada de requerimientos del cliente.

3. Reunión Interna de Planificación

Se toman los documentos iniciales y la lista de requerimientos del proceso anterior y, junto con el equipo de desarrollo, se genera una traducción de estos a la forma acordada por el equipo de desarrollo, sean "minimum marketable features", user stories, cartas kanban, casos de uso u otros. Para esto se pueden apoyar de diversas herramientas tanto ágiles como tradicionales, generando una lista de tareas a realizar durante la iteración, además de una lista de recursos necesarios y pruebas de sistema.

4. Reunión Diaria

Diariamente se realiza una reunión entre el jefe de proyecto y el equipo de desarrollo para revisar avances. De manera similar a Scrum, se menciona el trabajo realizado en el día anterior, el trabajo que se realizará al día actual y los posibles obstáculos que se encuentren. También se revisa el correcto estado de la tabla de tareas (en la forma que el equipo de desarrollo haya acordado). Como resultado de esta reunión se obtiene una tabla que refleja el estado real del trabajo, así como una lista de recursos o tareas que debe realizar el jefe del proyecto para remover los obstáculos del equipo.

5. Ejecutar Tareas

Las tareas son ejecutadas por medio de “pulling”, es decir, los miembros del equipo de desarrollo se adjudican las tareas, en lugar de éstas ser asignadas por una jefatura. Esto evita la sobrecarga de trabajo y el multitasking ya que asegura que las tareas son asignadas cuando existe capacidad para realizarlas en primer lugar. Se recopilan también estadísticas sobre el trabajo para realizar ajustes futuros. Por último, Lean coloca un gran énfasis en generar un buen ambiente de

desarrollo, por lo tanto es necesario implementar herramientas o prácticas que resulten en una mayor calidad de software desarrollado (tales como testing automatizado, control de versiones, estándares de código, desarrollo orientado a testing, etc.). Estas modificaciones deben ser discutidas durante la reunión retrospectiva e implementadas a la brevedad.

Como resultado de este proceso se espera avances y entregas de la iteración, así como una tabla de tareas transparente y una serie de estadísticas que ayuden a tomar decisiones sobre iteraciones futuras

6. Validación de la Entrega

Se reúne el jefe de proyecto y el equipo de desarrollo con el cliente y el dueño del proceso para mostrar el trabajo realizado durante la iteración y reunir retroalimentación. Como entrada al proceso se entrega también la tabla de tareas, las minutas de la reunión de planificación de ciclo y las estadísticas. Como resultado se espera una validación de la entrega (o de ciertos elementos de la entrega), así como retroalimentación para ser utilizada durante la próxima iteración, tanto de los clientes como de los usuarios (éstos últimos, de ser necesario, por medio de encuestas).

7. Realizar Traspaso

Se traspasan los elementos validados a producción, entregando completa trazabilidad, por medio de la tabla de tareas, de los elementos desde su captura como requerimiento hasta su implementación. Es crucial realizar testing en esta etapa para capturar errores finales o errores de integración con los demás sistemas. Como resultado del testing se deben entregar los tests utilizados durante el proyecto, así como la documentación acordada entre el equipo de desarrollo y el equipo de mantención, tales como Casos de Uso, Manuales de Usuario, Descripción de la Arquitectura, etc.

8. Reunión Retrospectiva

Al final de cada iteración se realiza una reunión entre el jefe del proyecto y el equipo de desarrollo donde se examina el actuar durante la iteración, promoviendo aquellas prácticas o decisiones que aumentaron el éxito de la iteración y eliminando las prácticas nocivas. También se buscan aquellas prácticas o subprocesos que no otorgan valor al cliente (en otras palabras, basura) para ser eliminadas. Además se discute la implementación de nuevas prácticas o herramientas de desarrollo.

La modificación del proceso de desarrollo se debe realizar con el consentimiento del equipo de desarrollo e idealmente apoyándose en las estadísticas reunidas durante la iteración.

3.3 Equivalencia con PMI

Como fue mencionado anteriormente, con el propósito de verificar la validez de Lean se propuso realizar una equivalencia entre éste y otra metodología o marco de referencia con mayor historia en el área de gestión de proyectos, en particular, con las prácticas del PMBOK. El desglose de la equivalencia se encuentra en el Anexo 6, pero a grandes rasgos se encuentra que cerca de todos los procesos que se describen en el PMBOK tienen un equivalente en la metodología propuesta. En particular, los procesos sin equivalentes están relacionados los procesos de adquisiciones (sobre el cual no hay detalles en el libro de Lean), los procesos de gestión de costos y ciertos elementos de la gestión de riesgos como es el proceso de cuantificación de los riesgos.

Cabe destacar que, sobre el proceso de gestión de costos, si bien es crucial para determinar el éxito del proyecto, éste está muy cercanamente relacionado con el proceso de gestión de requerimientos. Como se menciona tanto en el libro como en (2) la mayoría de features de un sistema no se suelen utilizar, lo cual representa un costo inmenso en el sistema final.

Por otra parte, la elección correcta del contrato también incide en el costo final. Por ejemplo, un contrato en el cual se paga por el sistema a costo fijo puede resultar en un costo extremadamente alto si no hay claridad en los requerimientos y se maneja un costo por cambio de éstos. Es común encontrar casos en los cuales un desarrollo es vendido a un precio bajo pero con alto costo de cambio para sobrecompensar.

La gestión de riesgos, por otra parte, se realiza con una diferente perspectiva. En lugar de analizar y esperar la llegada de riesgos, éstos se abordan en el momento correcto, lo cual Lean denomina “Responsablemente tarde”. Esto se debe a que, por una parte, una decisión atrasada conlleva un mayor costo (como puede ser un cambio tardío de la arquitectura). Por otra parte, una decisión temprana implica una decisión sin toda la información disponible la cual puede resultar errada después.

Resumiendo, se observa que, con excepción del área de gestión de adquisiciones y de costos, el proceso de desarrollo sugerido cumple las actividades y procesos sugeridos en el PMBOK

3.4 Chaos Report

El CHAOS Manifesto es un documento que es parte del CHAOS Chronicles, el cual contiene datos sobre más de 50.000 proyectos de desarrollo de software con el objetivo de obtener las mejores prácticas y elementos para tener éxito en proyectos de este tipo. La edición del 2013 contiene datos sobre proyectos pequeños la cual será utilizada como punto de referencia para el método a construir.

En el Anexo 5, se puede comprobar que el método sugerido cumple, a grandes rasgos, las recomendaciones del reporte en su versión del año 2013.

3.5 Cuestionarios

Ante la necesidad de poder evaluar la implementación de una metodología Lean, se presentan dos cuestionarios, uno práctico y uno teórico. El primero tiene como objetivo comprobar la cercanía del método practicado con el método sugerido en el Anexo 2. El teórico, por su parte, tiene como objetivo revisar la cercanía de la metodología practicada con las recomendaciones e ideas de Lean. La utilidad de este segundo cuestionario radica en encontrar implementaciones de Lean o Ágil que no sean la sugerida. Esto es necesario pues la implementación final debe ser acorde a las necesidades específicas del equipo de desarrollo y, ante estos casos, la encuesta teórica entregaría valores altos con respecto a la cercanía con Lean.

Con respecto a la relación entre Lean y Agile, en el área de manufactura, la presencia de Agile implica la existencia de Lean (pero la presencia de Lean no presupone la existencia de Agile) (32). Intuitivamente, esta relación puede existir debido a que el libro de Poppendieck contiene varias menciones positivas sobre los Principios de Ágil. Esta relación se examina en el Anexo 1 en el cual se muestra cierta equivalencia entre, por un lado, de los principios de Lean y los del Manifiesto Ágil y por otro, entre los principios de Lean y las prácticas específicas de Scrum, este último a modo de ejemplo.

Como consecuencia de lo anterior, se sostiene que, dada la fuerte relación entre Lean y Ágil, realizar el cuestionario sobre un equipo utilizando métodos ágiles puede ser de utilidad, entregando una evaluación del grado de “Agilidad” del proceso.

Para completar este cuestionario se requiere la ayuda de un jefe de proyecto o supervisor de desarrollo, ya que muchas preguntas tratan sobre el manejo de las expectativas del cliente y comportamiento del equipo. Además, se asume que, para cada apartado, un 75% de respuestas positivas implican una correcta aplicación de las prácticas o ideas Lean. El detalle particular de los cuestionarios, así como las respuestas esperadas se encuentra en el Anexo 3

4. Experimentos y Resultados

El cuestionario fue realizado en dos empresas que producen software, Canal 13 y Akzio, siendo contestadas por el jefe de área y un supervisor de desarrollo respectivamente. Los detalles del cuestionario están en el Anexo 3 y aquí se muestran los comentarios finales sobre el análisis de las respuestas.

4.1 Resumen de Respuestas

Cuestionario Teórico

	Canal 13	Akzio
Principio 1: Eliminar la Basura	11/16	11.5/16
Principio 2: Amplificar el conocimiento	7.5/7	7/7
Principio 3: Decidir lo más tarde posible	4.5/5	5.5/4
Principio 4: Entregar lo más rápido posible	1.5/4	4.5/5
Principio 5: Fortalecer al equipo	8/8	4/8
Principio 6: Incluir Integridad	3/4	4/4
Principio 7: Ver el todo	2.5/3	2.5/4

Cuestionario Práctico

	Canal 13	Akzio
Proceso 1: Generar Acta	4.8/6	7.8/8
Proceso 2: Planificar Ciclo	9/9	6/9
Proceso 3: Planificación Interna	4.5/5	5/5
Proceso 4: Reunión Diaria	5/8	5/8
Proceso 5: Ejecutar Tareas	3.5/6	3.5/6
Prácticas de Desarrollo	13.5/18	9.5/18
Proceso 6: Validar Entrega	3/3	3/3
Proceso 7: Realizar Traspaso	2/4	4.5/
Proceso 8: Reunión Retrospectiva	12.5/13	8/13

4.2 Caso Canal 13

Canal 13 pertenece al rubro de la televisión y produce software para manejar sus propias necesidades. Utilizan una tabla Kanban (la cual, desafortunadamente, no tiene límites de trabajo) y se encuentran bajo constantes auditorías, lo cual significa que tienen la necesidad de traer cierto grado de formalidad al proceso de desarrollo. El cuestionario y las conversaciones con el entrevistado muestran un proceso iterativo minimal, adolecido por el exceso de carga de trabajo (que se traduce en esperas y multitasking, ambas formas de basura) y debilidades en el área de testing. En el lado más positivo, se encuentra un sentido de cohesión en el equipo de desarrollo y existe un fuerte énfasis en generar y esparcir el conocimiento sobre el negocio entre los miembros del equipo.

Más específicamente, en el lado teórico, tienen bajos valores en lo que se refiere a “Entregar Rápidamente” y “Eliminar Basura”, ambos siendo problemas relacionados: La entrega lenta conlleva una necesidad de realiza multitasking lo cual ralentiza aún más la velocidad de desarrollo. Por otra parte, el equipo de trabajo tiene motivación y libertad para abordar los problemas a su parecer y existen procedimientos para difundir el conocimiento entre los miembros del equipo, facilitando la captura de requerimientos.

Con respecto al lado práctico, se destaca el proceso de toma de requerimientos, el cual incluye a todos los involucrados y coloca énfasis en generar una visión común, mientras que la ejecución de las tareas aparece como una complicación, pues la carga excesiva conlleva a la falta de testing, lo cual potencialmente implica un empeoramiento de la calidad.

Como profundización de Lean, se recomienda focalizarse en aquellas áreas con puntaje bajo, es decir, aumentar la velocidad de entrega y eliminar la basura, en particular, el nefasto multitasking. Esto se podría lograr con una combinación de aumento de personal como por una restricción de la demanda de desarrollo, logrando así una nivelación de la velocidad de desarrollo.

4.3 Caso Akzio

Akzio es un caso particularmente interesante. Realiza software para empresas externas, Entel, en el caso analizado. La entrevista revela que, si bien es el proceso de desarrollo parece ser Scrum, la realidad se acerca más a un proceso conocido informalmente como “WaterScrumFall”, es decir, una mezcla entre Scrum y Cascada. Esto se hace evidente en el cuestionario cuando se revela que la captura de requerimientos no es iterativa y se entrega en forma de documentación al inicio del proyecto, mientras que el desarrollo se realiza de forma iterativa utilizando prácticas y artefactos de Scrum.

Con respecto a los resultados, en el lado teórico la mayor debilidad es la poca libertad y motivación que poseen los integrantes del equipo de desarrollo, cuyo trabajo se reduce a implementar aquello que es documentado. Por otra parte, extrañamente, se reúnen estadísticas de forma individual, lo cual puede sugerir una jefatura preocupada por el "tiempo de uso del recurso" que por sobre el correcto funcionamiento del equipo de trabajo. La eliminación de la basura puede ser también un tema importante, pues el proceso de priorización es mínimo y depende de la capacidad de negociación de la jefatura, mientras que las esperas al inicio del proyecto deberían ser reducidas. Sin embargo se destaca la capacidad de generar conocimiento, debido a que la toma de requerimientos se realiza una sola vez sin cambios negativos durante el avance del proyecto. Esto sólo se puede lograr si existe un profundo conocimiento y visión común sobre el negocio del cliente por parte de todos los actores involucrados en la toma de requerimientos. Gracias a esto, a ejecución de las tareas se vuelve relativamente fácil, lo cual conlleva una alta velocidad de entrega.

En el lado práctico, el uso de WaterScrumFall es contrario a las prácticas recomendadas por Lean, notándose en el apartado de captura de requerimientos (proceso 2). También existe cierta falta de visibilidad para los clientes, ausencia de libertad para los desarrolladores y espacio para realizar mejoras en la ejecución de las tareas (en particular, el área de testing). Por otra parte, se destaca que existe una gran motivación por parte de los interesados (probablemente por la velocidad de desarrollo y buen cumplimiento de sus requerimientos y buen tratamiento de los sprints donde existe gran preocupación por el testing (si bien éste puede ser mejorado por medio de automatización)

En términos de profundización de la metodología Lean, se recomienda potenciar al equipo de trabajo, por medio de motivación y liderazgo. Por otra parte, si bien la captura temprana de requerimientos de forma definitiva entrega buenos resultados para este caso en particular, es importante notar que ante nuevas problemáticas o casos de negocio, éste puede ser un riesgo, por lo que se recomienda contar con un proceso distinto para enfrentar problemas nuevos.

5. Conclusiones

Por medio de presente informe se presentó a Lean como una alternativa para realizar implementaciones ágiles en organizaciones que posean adversidad al cambio. Lean presenta el beneficio particular de tener un bajo impacto pues el punto de inicio es siempre el proceso actual. Por otra parte, como se expuso, Lean es compatible con PMI como marco de referencia, lo cual provee un fuerte argumento para aquellas jefaturas que prefieran utilizar métodos de gestión de proyectos tradicionales.

El cuestionario, además, permite comprobar la cercanía con el proceso propuesto y puede servir como referencia para aquellos que deseen implementar Lean. El método propuesto, por su parte, entrega ejemplos de herramientas útiles para realizar la implementación, aunque, como todo marco de referencia, es importante elegir las herramientas que provean valor real al equipo de desarrollo. Además, la metodología fue desarrollada con un caso particular en mente, proyectos pequeños con equipos pequeños que manejen sólo un proyecto a la vez.

Idealmente, se espera que este documento pueda servir como herramienta para facilitar implementaciones ágiles y/o persuadir a las jefaturas a realizar estas implementaciones, entregando a las organizaciones los beneficios que la literatura al respecto promete.

Como trabajo futuro, sería interesante realizar implementaciones de Lean en ambientes considerados adversos o tradicionales, revisando si Lean es capaz de modificar la cultura organizacional que rodea al equipo de trabajo. De ser así, Lean podría servir como alternativa a los cambios bruscos sugeridos por otras metodologías de desarrollo y como ruta intermedia entre "lo tradicional" y lo ágil.

Bibliografía

1. **VersionOne.** *7th Annual State of Agile Development Survey.* 2012.

2. **The Standish Group.** *The CHAOS Manifesto 2013.* EE.UU. : The Standish Group International, 2013.

3. **Ambler, S. W.** Comparing Software Development Paradigms. [Online] Enero 24, 2014. [Cited: Enero 26, 2017.] http://scottambler.com/backup_muse/comparing-software-development-paradigms.html.

4. **Poppendieck, Mary and Poppendieck, Tom.** *Lean Software Development: An Agile Toolkit.* s.l. : Addison-Wesley Professional, 2003.

5. *Managing the development of large software systems.* **Royce, Winston W.** 1970. Proceedings, IEEE WESCON. pp. 1-9.

6. *Iterative and incremental developments. a brief history.* **Larman, C. and Basili, V.** 6, Junio 11, 2003, Computer, Vol. 36, pp. 47-56.

7. **Beck, Kent, et al., et al.** Manifesto for Agile Software Development. [Online] 2001. [Cited: Enero 26, 2017.] http://agilemanifesto.org/.

8. **Schawaber, Ken and Sutherland, Jeff.** The Scrum Guide. [Online] Julio 2016. [Cited: Enero 26, 2017.] http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf.

9. **Wells, Don.** Extreme Programming: A gentle introduction. [Online] Octubre 2013. [Cited: Enero 26, 2017.] http://www.extremeprogramming.org/.

10. **Larman, C. and Vodde, B.** Lean Primer. [Online] 2009. [Cited: Enero 26, 2017.] http://www.leanprimer.com/downloads/lean_primer.pdf.

11. *Lean Software Development: Two Case Studies.* **Middleton, Peter.** Diciembre 2001, Software Quality Journal, pp. 241-252.

12. **Hou, Alexander C.** *Toward Lean Hardware/Software System Development: Evaluation of Selected Complex Electronic System Development Methodologies.* 1995.

13. *Lean software development: is it feasible? .* **Raman, S.** 1998. Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE.

14. *Identifying lean software development values.* **Lane, Michael T., Fitzgerald, Brian and Ågerfalk, Pär J.** Barcelona : s.n., 2012. Proceedings of ECIS Conference.

15. **Anderson, David J.** *Kanban: Successful Evolutionary Change for Your Technology Business.* 2010.

16. **Anderson, David J., et al., et al.** *A comparative study of Scrum and Kanban approaches on a real case study using simulation.* Beijing : s.n.

17. *Lean Software Development: A Systematic Review.* **Jonsson, Henrik.** Västerås : s.n., 2012. Research Methodology Course Mini Conference.

18. **Staats, Bradley R., Brunne, David J. and Upton, David M.** *Lean Principles, Learning, and Knowledge Work: Evidence from a Software Services Provider.* 2010.

19. *Lean Software Management: BBC Worldwide Case Study.* **Middleton, Peter and Joyce, David.** 2010. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT.

20. *Providing value to customers in software development through lean principles.* **Mehta, M., Anderson, D. and Raffo, D.** 2008, Advances in Software Process Improvement , pp. 101-109.

21. **Skarin, Henrik and Kniberg, Mattias.** Kanban and Scrum - Making the Most of Both. [Online] Diciembre 21, 2009. [Cited: Enero 27, 2017.] https://www.infoq.com/minibooks/kanban-scrum-minibook.

22. *Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles: Software engineering and software development .* **Manjunath, K. N., Jagadeesh, J. and Yogeesh, M.** 2013. 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s) .

23. *Agile & Kanban In Coordination.* **Polk, Ryan.** 2011. Agile Conference.

24. *Enhanced Agile Software Development – Hybrid Paradigm with LEAN Practice.* **Perera, G.I.U.S. and Fernando, M.S.D.** Sri Lanka : s.n., 2007. econd International Conference on Industrial and Information Systems. pp. 239-243.

25. *Implementing the Lean Concepts of Continuous Improvement and Flow on an Agile Software Development Project: An Industrial Case Study .* **Swaminathan, Balachander and Jain, Karuna.** 2012. Agile India .

26. *The Combination of Agile and Lean in Software Development: An Experience Report Analysis .* **Wang, Xiaofeng.** 2011. Agile Conference (AGILE).

27. *From CMMI and Isolation to Scrum, Agile, Lean and Collaboration.* **Hansen, Mads Troels and Baggesen, Hans.** 2009. Agile Conference.

28. **Project Management Institute.** *PMBOK® Guide and Standards.* Pennsylvania : Project Management Institute, Inc., 2013.

29. —. *Softare Extension to the PMBOK Guide Fifth Edition.* Pennsylvania : Project Management Institute, Inc., 2013.

30. **VersionOne.** *10th annual State of Agile™ survey.* 2015.

31. *Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers.* **Addison, Tom and Vallabh, Seema.** 2002. Proceedings of SAICSIT 2002. pp. 128 – 140 .

32. *Disentangling leanness and agility: An empirical investigation.* **Narasimhana, Ram, Swinka, Morgan and Soo Wook Kimb.** 2006, Journal of Operations Management, pp. 440–457.

Anexo 1: Relaciones entre Agile/Scrum y Lean

Relación entre prácticas de Scrum con principios de Lean.

Fuente: Elaboración del autor con influencia de la Guía de Scrum (8) y el libro de Lean (4):

Práctica de Scrum	Teoría de Lean
Existencia de un Product Owner	• Eliminar Basura: Procesos Extra (lograr identificar al product owner y centrarse en él reduce procesos extra) • Eliminar Basura: Mapear procesos (mapear y limpiar el proceso recalca la importancia del product owner)
Equipo de desarrollo Independiente y libre	• Fortalecer al Equipo: Autodeterminación • Fortalecer al Equipo: Liderazgo
Equipo de desarrollo igualitario	• Fortalecer al Equipo: Autodeterminación
Equipo de tamaño pequeño	• Amplificar conocimiento (el conocimiento se genera y distribuye más rápidamente en grupos pequeños)
Existencia de un Scrum Master	• Fortalecer al Equipo: Autodeterminación • Fortalecer al Equipo: Liderazgo • Master Developers • Fortalecer al Equipo: Motivación • (el Scrum Master es una combinación de elementos de Lean)
• Sprints	• Amplificar Conocimiento: Iteraciones
• Reunión de Planificación de Sprint	• Amplificar Conocimiento: Iteraciones
• Backlog	• Amplificar Conocimiento: Iteraciones (Backlog aparece mencionado como método) • Fortalecer al equipo: Autodeterminación (la manera de manejar el backlog debe ser autodeterminada)
• Libertad del equipo de desarrollo para decidir qué hacer en el Sprint	• Fortalecer al Equipo: Autodeterminación
• Definición de “Hecho”	• Incluir Integridad
• Objetivo del Sprint	• Amplificar Conocimiento: Retroalimentación (obtener retroalimentación correctamente requiere una iteración definida

• Reunión de Scrum Diaria	• Entregar el trabajo lo más rápido posible: Pull Systems • Amplificar el Conocimiento
• Reunión de Revisión de Sprint	• Amplificar Conocimiento: Iteraciones
• Retrospectiva de Scrum	• Fortalecer al Equipo
• Monitoreo de Sprint	• Ver el todo: Mediciones
• Transparencia de artefactos	• Autodeterminación • Eliminar la Basura (maximizar entrega de valor) • Eliminar basura: Mapear proceso

Relación entre principios de Agile y principios de Lean

Fuente: Elaboración del autor con la lectura de los Doce Principios del Software Ágil[1] y lectura del libro de Lean (4)

Agile	Lean
"Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor."	• Eliminar la Basura • Entregar lo más rápido posible
"Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente."	• Decidir lo Más Tarde Posible (flexibilidad) • Eliminar la Basura (valor al cliente) • Entregar lo más rápido posible (velocidad de cambio)
"Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible."	• Amplificar el conocimiento: Iteraciones
"Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto."	• Eliminar la Basura • Amplificar el Conocimiento • Incluir Integración
"Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo."	• Fortalecer al equipo

[1] http://agilemanifesto.org/iso/es/principles.html

“El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.”	• Incluir Integridad
“El software funcionando es la medida principal de progreso.”	• Entregar lo más rápido posible
“Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.”	• Fortalecer al Equipo (autodeterminación)
“La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.”	• Fortalecer al Equipo (motivación y estándares)
“La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.”	• Eliminar la Basura
“Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.”	• Fortalecer al Equipo (autodeterminación)
“A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.”	• Fortalecer al Equipo (autodeterminación)

Anexo 2: Proceso Sugerido

Contexto

1. Sobre Lean

Es necesario aclarar de inicio que Lean no es estrictamente una metodología de desarrollo propiamente tal en el sentido que no prescribe procesos. El libro de Lean de Poppendieck se refiere a Lean como un "Toolkit" o caja de herramientas la cual es aplicada al proceso real de la organización. Más abstractamente, Lean puede ser visto como una "filosofía" que gira en torno a los principios lean.

Un concepto muy importante en la manufactura Lean de Toyota es la idea de Kaizen (mejora) pero que en la manufactura se refiere a la mejora constante de las practicas, herramientas y procesos del negocio. Lean como filosofía, entonces, puede ser visto como el vehículo que trae el cambio que produce las mejoras constantes. Por ejemplo, el principio de "Eliminar la basura" lleva a encontrar y eliminar las áreas o prácticas que no generan valor para el cliente, bajando los costos y aumentando la velocidad de entrega.

Recordemos el Manifiesto por el Desarrollo Ágil de Software (apoyado por los autores del libro):

"[…]. 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 […]"

Es importante notar la idea de responder (y por lo tanto aceptar) el cambio antes que luchar contra él o minimizarlo, pues de esta idea es fácil darse cuenta que, si cada proyecto de software es diferente, ¿Por qué se debería adoptar un método único de desarrollo? ¿No se debería adaptar el proceso de desarrollo al proyecto de software? Los principios Lean ayudan aquí a esclarecer qué prácticas o herramientas se pueden adoptar o rechazar.

De todo lo anterior, se concluye que Lean debe ser visto en el contexto de un proceso de desarrollo establecido, revisando si los diversos actores del proceso siguen los principios Lean y revisando si se aplican las (pocas) prácticas explicitas que se mencionan en el libro.

2. Sobre PMI

Los procesos de PMI se dividen a grandes rasgos en 5 tipos:

- Inicio
- Planificación
- Ejecución
- Monitoreo y Control
- Cierre

Lean no tiene mucho material con respecto a los procesos de Inicio y Cierre (excepto por un pequeño capitulo que discute las ventajas y desventajas de ciertos tipos de contrato), por lo que es conveniente seguir los procesos definidos en el PMBOK en esas áreas.

En contraposición, Lean tiene bastante material en lo que se refiere a Ejecución y Monitoreo. Por otra parte, PMBOK abunda en material con respecto a la planificación. Sin embargo, se podría discutir que es demasiado material con respecto a la planificación. Recordando de nuevo el Manifiesto Ágil, la documentación excesiva de PMI y su énfasis en tener un plan detallado puede llevar a seguir un proceso en cascada, con las consecuencias nefastas que esto implica. La documentación de extensión del PMBOK (29), por otro lado, detalla el lado, mas "ágil" de PMI, presentando varias nuevas herramientas y consideraciones para los desarrollos de este tipo. Es sobre esta base que se realizará el análisis de los procesos de desarrollo.

3. Sobre los métodos iterativos

Lean se apoya en el desarrollo iterativo para minimizar el riesgo, maximizar el conocimiento y entregar resultados rápidos de alta calidad. Estos métodos se basan en iteraciones de entre 1 y 4 semanas en los que se realizan reuniones con los interesados, se discuten y se negocian los objetivos para la iteración actual culminando en una entrega de los objetivos discutidos al final de la iteración. Ya que las iteraciones son, idealmente, autocontenidas, es decir, no se acarrean tareas de una iteración a otra, la planificación que se realiza en un proyecto iterativo es centrada en cada una de las iteraciones, en consecuencia, la planificación es mucho más liviana y con información más actualizada.

PMI contiene bastantes artefactos llenos de información que deben ser constantemente actualizados. Lean, por otra parte, sostiene que sólo se debe completar aquella documentación que entregue valor al cliente. Los artefactos de Lean pueden contener una gran cantidad de información en un formato relativamente fácil de actualizar (por ejemplo, los backlogs). Por lo tanto el análisis está enfocado en aquellas herramientas que permitan maximizar ese flujo de información con el menor costo por sobre la documentación excesiva que PMI puede, a veces, sugerir.

4. Sobre las implementaciones particulares.

Cabe resaltar, como se mencionó anteriormente, que no todos los proyectos de software tienen las mismas características, por lo que es conveniente (y en ocasiones necesario) alterar un proceso de desarrollo para acomodarlo a las nuevas exigencias. Ejemplos de consideraciones adicionales:

- Proyectos de muy alto riesgo incluyendo pérdida de vida humana: Es muy importante definir tempranamente y verificar constantemente los requerimientos de seguridad que requiere el software.
- Proyectos públicos donde se debe demostrar la utilización de fondos: Especificar tempranamente la clase de documentación necesaria y establecer un formato de fácil actualización. Negociar una entrega de documentación que detalle el trabajo realizado y en vez de utilizar la documentación como guía de trabajo.
- Proyectos donde todos los actores conocen el negocio y el cliente tenga muy claro los requerimientos: Es posible realizar una implementación de cascada.
- Proyectos de enorme envergadura: Discutir la posibilidad de realizar subproyectos, con la precaución de realizar una cuidadosa integración entre los subproyectos.

Cabe destacar que cada cliente o usuario puede tener consideraciones personales adicionales. Todo esto impacta en la forma de relacionarse con él. Teniendo en cuenta que la correcta gestión de los interesados dicta la mayor probabilidad de éxito en proyectos, es crucial, negociar, informar y dialogar una manera de relacionarse con los interesados acorde a las necesidades del proyecto.

Detalle del Proceso

Proceso general: Este proceso está orientado al desarrollo de software nuevo por parte de un equipo de trabajo que quiera implementar prácticas Lean. Una vez realizada el acta de constitución o contrato del proyecto, éste se rige por medio de iteraciones en las que el cliente determina aquello que quiere recibir al final de la iteración, siguiendo, a grandes rasgos, una fase de planificación de iteración, una reunión diaria de seguimiento, un trabajo diario en el proyecto por parte de los desarrolladores y los testers. Finalmente, el cliente prueba y verifica la iteración, entregando retroalimentación que es utilizada para la siguiente iteración

Inicio del proyecto

1. Generar Acta de Constitución o Contrato

Inicio de Ciclos de Desarrollo

Inicio de Ciclo

2. Planificar Ciclo

3. Reunión Interna de Planificación

Diariamente:

4. Reunión Diaria

5. Ejecutar Tareas

6. Validar Entrega

7. (Opcional) Realizar Traspaso

Fin de Ciclo

8. Reunión Retrospectiva

Fin de Ciclos de Desarrollo

7. Realizar Traspaso

Fin del Proyecto

1. Generar Acta de Constitución o Contrato

El proceso de "Generar Visión, Condiciones Iniciales, Aceptación Firmada o Contrato" es el proceso de desarrollar, en conjunto con los interesados, la visión inicial del proyecto a desarrollar, así como definir las restricciones o consideraciones particulares del proyecto (tales como seguridad) y generar un documento formal que autorice formalmente la existencia del proyecto.

Muchos procesos de planificación que se encuentran en el PMBOK son variables y genéricos ya que se pueden establecer procesos particulares para proyectos particulares de diferente índole. Por otra parte, este documento se centra solamente en proyectos de TI de desarrollo de software aplicando Lean, por lo que se asume una serie de procesos (desarrollo iterativo, reuniones cíclicas con los interesados, reuniones diarias, etc.). Esto permite acortar la gestión de los procesos para definir los casos excepcionales al inicio del proyecto (o, en la mayoría de los casos, para enunciar la forma de trabajo a los interesados).

Específicamente, en este proceso, el (futuro) jefe de proyecto se reúne con los interesados del proyecto con el objetivo de definir una visión del proyecto, así como las condiciones del proyecto. El objetivo de esta reunión es, por una parte, obtener la información necesaria para lograr decidir si el jefe de proyecto y su equipo pueden abordar exitosamente el proyecto, por la otra, informar a los interesados del modus operandi del proyecto, informando, por ejemplo, que se realizarán reuniones periódicas y que se espera un interés constante de los interesados en revisar los avances entregados.

Con respecto a los interesados, se agrega, además que aquellos que estén presentes en esta reunión no son necesariamente los únicos interesados. Si bien el interesado que conoce el negocio está generalmente presente, el usuario que maneja este negocio puede estar ausente. Es importante que este usuario sea involucrado en el proceso de desarrollo de forma temprana para que pueda probar y aprender a usar el sistema que se quiere desarrollar.

Se recuerda que cuando se realice la negociación sobre las reuniones, al calendarizar o estandarizar las reuniones de validación y planificación, éstas deben tener, como máximo, un mes de diferencia.

1.1 Entradas

1.1.1 Factores Ambientales de la Empresa

Se refiere a condiciones fuera del control del equipo del proyecto, las cuales influencian en el resultado del proyecto. Entre ellas se encuentran:

- Cultura y estructura de la empresa
- Distribución geográfica del equipo, instalaciones y los recursos necesarios

- Estándares industriales y gubernamentales
- Recursos humanos disponibles
- Legislación laboral
- Estado del Mercado

1.1.2 Activos de los Procesos de la Organización

Se refiere a planes, procesos, políticas, información histórica y bases de conocimiento pertenecientes a la organización y que pueden utilizarse al momento de realizar el proyecto. Estos documentos pueden modificarse con los resultados e información producida durante el transcurso del proyecto. Estos pueden incluir:

- Procedimientos y estándares de la organización
- Guías de procesos
- Plantillas
- Software de control de proyectos (financiero, aprobación, riesgos, etc.)
- Bases de conocimiento
- Información histórica

1.2 Herramientas y Técnicas

1.2.1 Juicio de Expertos

Un experto es aquel con experiencia y conocimiento en los temas técnicos o de gestión del proyecto o la necesidad de los interesados. La organización de los interesados puede poseer expertos en otras áreas de su organización, mientras que el jefe de proyecto puede tener desarrolladores con experiencia en temas similares (posiblemente Master Developers)

1.2.2 Negociación de Contrato

El tipo de contrato elegido puede cambiar el desenlace del proyecto, por lo que es importante negociar un tipo de contrato que pueda satisfacer las necesidades, expectativas y aprehensiones de las partes negociantes.

1.3 Salidas

1.3.1 Acta de Constitución del Proyecto.

El Acta de Constitución del proyecto es el documento interno que afirma la existencia del proyecto y su inicio. Este documento posee una descripción de alto nivel del proyecto y de la necesidad de negocio, los criterios de éxito, las restricciones, condiciones, supuestos, los responsables del proyecto y los hitos.

1.3.2 Lista y descripción de procesos excepcionales

Es un documento que detalla procesos excepcionales que se deben llevar a cabo durante la ejecución del proyecto y que representan una desviación de la vía estándar de desarrollo.

1.3.3 Lista de Interesados y sus formas de comunicación

Es una lista de los interesados del proyecto (tanto poseedores del proceso de negocio como usuarios del sistema), junto con su cargo, rol en el proyecto y su forma de contacto.

Esta lista debe ser exhaustiva, pues la ausencia de algún interesado puede significar el fracaso del proyecto. Entre posibles interesados se incluyen:

- El cliente que solicitó el proyecto
- El dueño del proceso de negocio
- Jefaturas que controlen los recursos del proyecto
- Usuarios y/o Público
- Auditores
- Todos aquellos que sean afectados por la ejecución y/o entrega del proyecto

1.3.4 Lista de reuniones programadas o frecuencia de reuniones de entrega

Se refiere a la lista de reuniones programadas con su correspondiente entrega (en caso de seguir un modelo de entrega por hitos) y la frecuencia de reuniones de tipo planificación y validación junto con los asistentes preliminares a cada una de esas reuniones.

1.3.5 Lista inicial de Restricciones, Condiciones y Supuestos del proyecto

Es un documento que contiene las restricciones, condiciones y supuestos del proyecto. Este documento es actualizado continuamente durante la ejecución del proyecto conforme se disemina la información y la visión sobre éste.

1.3.6 Contrato Firmado

En caso de que el proyecto sea con una empresa externa, se firma el contrato que autoriza el inicio del proyecto

1.4 Consideraciones adicionales

1.4.1 Tipo de Contrato

Según Lean, los tipos de contrato y sus observaciones son:

1.4.1.1 Contrato de precio Fijo:

Tiene el riesgo de que la empresa contratada ofrezca un precio bajo pero capitalice por medio de costos de cambio con respecto a la especificación inicial (que generalmente es cambiada). Si el proceso de adjudicación del proyecto es favoreciendo a la alternativa de menor costo, entonces, si el cliente no entiende a cabalidad el problema, existe el riesgo de que el problema sea entregado al proveedor más optimista o desesperado. Finalmente posee el riesgo de que la empresa contratista niegue la compleción del trabajo al final del proyecto.

La mayor parte del riesgo y el costo van hacia el proveedor

1.4.1.2 Contrato por Tiempo y Materiales

El proveedor puede alargar innecesariamente el proyecto con el objetivo de maximizar sus ganancias a costa de la empresa contratista

Vuelca el costo hacia la empresa contratista.

Son una buena elección para este método de desarrollo, beneficiando a ambas partes, si se negocia un pago posterior a cada ciclo de desarrollo, entregando la opción a la empresa contratista de poner fin al proyecto después de cada entrega.

1.4.1.3 Contrato “Multi-etapa”:

Si existe una etapa previa de investigación del problema del proyecto, existe el riesgo de que se congele la especificación. Si es “por hitos” existe el riesgo de que con cada iteración o entrega se renegocie el contrato con la nueva información encontrada conforme avanza el proyecto.

1.4.1.4 Contrato Costo Objetivo:

Los de tipo Contrato de Costo más Tasa Fija (“Cost plus fixed fee”) dan incentivo a ambas partes a cooperar para maximizar ganancias. Los de tipo “Profit not to exceed” no entregan incentivo al proveedor para trabajar a bajo costo excepto si hay un bonus por entrega temprana

1.4.1.5 Contrato Fecha Objetivo:

Este tipo de contrato es utilizado en caso de que sea necesario llegar a una fecha particular por sobre el costo. Ejemplos de fechas importantes para un proyecto pueden ser fechas de arreglo de errores, SLAs, fechas de presentaciones o ferias o imposiciones legales. En estos casos es crucial trabajar con el cliente para manejar de forma estricta el backlog del proyecto.

2. Planificar Ciclo

Este proceso se realiza al principio de cada ciclo donde los interesados deciden qué quieren ver implementado al final de la iteración. El jefe de proyecto y/o el equipo de trabajo, por su parte, negocian lo que pueden entregar apropiadamente al final del ciclo. Es importante negociar correctamente el alcance de la iteración para no sobrecomprometerse y romper las expectativas de los interesados. Dependiendo de las herramientas utilizadas, puede ser posible realizar bosquejos del programa, por ejemplo, construyendo la interfaz durante la reunión, lo cual puede ayudar a todas las partes involucradas a entender y definir correctamente la solución a las necesidades de los interesados.

Los interesados, por lo general, solicitan aquello que necesitan de forma urgente primero. Esto, a su vez, reduce el riesgo general del proyecto, ya que se implementa y se prueba aquello que es crucial primero.

Dependiendo de la complejidad de las tareas y de la negociación del ciclo, es posible que se realice la descomposición de las tareas y su traspaso al backlog total o parcialmente durante la reunión.

Por último, cabe agregar que los requisitos pueden, incluir elementos como: modos de validación particulares, interesado que validador particular, condiciones no funcionales (fiabilidad, seguridad, accesibilidad, etc.), criterios de calidad, etc.

2.1 Entradas

2.1.10 Factores Ambientales de la empresa

Ver 1.1.1

2.1.2 Activos de los procesos de la Organización

Ver 1.1.2

2.1.3 Acta de Constitución del Proyecto o Contrato Firmado

Ver 1.3.1

2.1.4 Lista de Interesados y sus formas de comunicación

Ver 1.3.3

2.1.5 Lista de reuniones programadas o frecuencia de reuniones de entrega

Ver 1.3.4

2.1.6 Lista inicial de Restricciones, Condiciones y Supuestos del proyecto.

Ver 1.3.5

2.1.7 Datos estadísticos del proyecto o de la organización.

Ver 5.3.1

2.1.8 Backlog/Kanban

Ver 3.3.1

2.1.9 Calendario de Recursos

Ver 3.1.4

2.1.10 Lista de Cambios de las iteraciones anteriores

Ver 6.3.2

2.2 Herramientas y Técnicas

Con respecto a la captura de requerimientos

2.2.1 Entrevistas

Como es posible que la reunión no cuente con la presencia de todos los afectados con el proyecto o todos los conocedores del proceso de negocio, es recomendable realizar entrevistas a actores tales como otros usuarios o jefes de otras áreas. Si la ayuda de estas personas es recurrente es provechoso intentar negociar su ingreso al proyecto.

2.2.2 Técnicas Grupales de Creatividad:

Son herramientas que pueden ayudar a mejorar la calidad del proyecto. Entre ellas se encuentran:

- Tormenta de Ideas
- Técnicas de Grupo nominal: Se agrega votación a la tormenta de ideas para obtener una priorización.
- Mapa conceptual/mental: Se consolida en un solo esquema los puntos en común surgidos durante la tormenta de ideas con el punto de obtener los puntos en común y discutir las diferencias

2.2.3 Cuestionarios y encuestas.

Por ejemplo, en caso de necesitar la retroalimentación de varios usuarios, es posible realizar encuestas sobre un gran número de afectados con respecto a, por ejemplo, dos posibles prototipos. También son útiles en caso de que existan diferencias grandes entre usuarios y/o dispersión geográfica.

2.2.4 Prototipos

Consiste en producir uno o varios prototipos detallando posibles ramas de desarrollo del sistema. Este método requiere rapidez de entrega, pero permite que los interesados exploren varias perspectivas, reglas de sus necesidades y opciones de presentación. Según Lean es uno de los métodos más importantes y se deben invertir recursos en mejorar la velocidad y calidad del prototipado. Idealmente, debería ser posible definir, al menos, la interfaz del producto durante la reunión de planificación.

2.2.5 Análisis de Documentos

Algunos proyectos pueden poseer grandes cantidades de documentos con información relevante para la ejecución del proyecto tales como literatura del negocio, documentación de proceso, etc.

2.2.6 Juicio de Expertos

Ver 1.2.1

Descomposición de tareas

Ver 3.2.1

2.3 Salidas

2.3.1 Lista de Requisitos con priorización

Esta es la lista de requisitos capturados durante la reunión en la cual cada requisito debe tener asignada una prioridad con respecto a las demás, así como sus consideraciones particulares (por ej: seguridad, forma de validar, interesado validador, secuencialidad).

2.3.2 Minuta de Reuniones

Dependiendo del grado de confianza, es posible tener que realizar minutas de reuniones detallando los acuerdos logrados durante la reunión. Es caso de tener que realizar estas

minutas, éstas deberían ser firmadas por el jefe de proyecto y un representante de los interesados para dejar constancia de su validez.

2.3.3 Actualización a los documentos o artefactos del proyecto

Los documentos o artefactos del proyecto pueden sufrir modificaciones como consecuencia de esta reunión. Entre estos cambios se incluyen

- Actualizaciones a Backlog/Kanban
- Actualizaciones a la lista de restricciones, condiciones y supuestos
- Actualizaciones a la lista de reuniones
- Actualización a la lista de interesados

2.4 Consideraciones adicionales

Es importante notar que, según el Chaos Report, el 50% de las features de un programa no se utilizan, mientras que un 30% se utiliza sólo a veces. Es decir que, potencialmente, un 80% de un programa puede ser innecesario. Según Lean, esto se debe a que el usuario, al no estar seguro de lo que necesita (o temiendo que no sea consultado durante el desarrollo del sistema), tiende a pedir más de lo necesario “en caso de que lo llegue a necesitar”. Lean elimina este problema entregándole una voz a los interesados después de cada iteración y dando prioridad a aquellas features que los interesados necesitan primero.

3. Reunión interna de Planificación

En esta reunión, con ayuda del equipo de desarrollo, se realiza el traspaso de los requerimientos a la tabla de tareas con la priorización adjuntada en la reunión con los interesados. También se buscan los requerimientos adicionales necesarios para cumplir la lista de requisitos comprometida.

Con respecto a la gestión de riesgos: El uso de iteraciones, decisiones tardías con máxima información, retroalimentación constante y la entrega temprana de valor al cliente reducen considerablemente el riesgo lo cual significa que el análisis formal de los riesgos no debería ser necesario en la gran mayoría de los casos (pues, además, genera documentación innecesaria y basura general). De todas formas, es útil en esta reunión identificar los posibles riesgos de los requerimientos.

En esta reunión también se discuten las posibles pruebas que se pueden aplicar al final de la iteración para verificar el correcto funcionamiento del sistema.

En el caso de utilizar Kanban, es posible también ofrecer clases de servicio para distintos tipos de tarea. Esto significa una segunda priorización antes de traspasar las tareas a Kanban.

Finalmente se determinan los recursos necesarios para la compleción de la iteración y se los calendariza.

3.1 Entradas:

3.1.1 Lista de Requisitos con priorización y consideraciones

Ver 2.3.1

3.1.2 Backlog/Kanban

Tanto el Backlog como el Kanban son artefactos de monitoreo del proyecto, pudiendo ser físicos o electrónicos. El backlog es una tabla con tres columnas, una para la lista de tareas que deben ser completadas, otra para las tareas actualmente en desarrollo y una última para las tareas finalizadas. Cada tarea es movida entre las tres columnas para reflejar el estado actual del ciclo. La lista de tareas finalizadas puede ser borrada al final de la iteración o al final del proyecto, dependiendo de los acuerdos del equipo de desarrollo.

Por otra parte, Kanban es una tabla (física y/o virtual) que contiene las tareas del equipo de desarrollo en su estado actual, es decir, clasificadas en columnas que corresponden al área de desarrollo de la tarea (por ejemplo, “desarrollo”, “testing”, “traspaso”). La mayor peculiaridad de esta tabla es que cada columna tiene definido un máximo número de tareas concurrentes, lo cual apunta a reducir el multitasking, normalizar la carga de trabajo y ayudar a detectar cuellos de botella durante el proceso de desarrollo.

El uso de Kanban también se une directamente al concepto de pulling la asignación de una tarea se realiza por parte del propio equipo y sólo cuando existen los recursos para resolverla. Permite, además, la captura de estadísticas adicionales y el hallazgo de cuellos de botella en el proceso de desarrollo.

3.1.3 Minutas de Reuniones con los interesados

Ver 2.3.2

Lista inicial de Restricciones, Condiciones y Supuestos del proyecto.

Ver 1.3.5

3.1.4 Calendario de Recursos

Durante la ejecución del proyecto, es posible que se requieran de recursos adicionales para completar el proyecto. Un calendario de recursos identifica los tiempos en que uno de estos recursos adicionales está disponible. Estos recursos pueden ser de diversos tipos como, por ejemplo, disponibilidad de un usuario experto, disponibilidad de un servidor o fecha de obtención de licencias.

3.1.5 Lista de Procesos Excepcionales

Ver 1.3.2

3.2 Herramientas y Técnicas

3.2.1 Juicio de Expertos

Ver 1.2.1

3.2.2 Elicitación de Requerimientos

Consiste en buscar requerimientos adicionales a los ya estipulados para lograr los primeros. Por ejemplo: ¿qué tareas necesitan ser completadas previamente a cumplir un requisito? ¿Qué arquitectura es necesaria?

3.2.3 Estimación de peso

En caso de que la traducción de requerimiento a tarea requiera una estimación del esfuerzo, existen mecanismos que pueden facilitar esa estimación:

- Planning Poker de Scrum busca realizar estimaciones sin “anclaje” (causar que la primera estimación sirva de base para las siguientes).
- Preguntar a expertos.
- Estimación de tres puntos (estimar mejor, peor y más probable caso y generar un estimado con la probabilidad de cada caso)
- Estimar apoyándose en tareas completadas similares o tareas más pequeñas que se puedan escalar hasta el tamaño actual

3.2.4 Análisis de Alternativas

Revisar las posibles formas de resolver un requisito del usuario. Estas opciones pueden conducir a prototipos independientes.

3.2.5 Descomposición:

Consiste en dividir las tareas en tareas más pequeñas. El tamaño final de la tarea y su descomposición particular depende del consenso del equipo. Por ejemplo, se pueden dividir las tareas en “Minimum Marketable Features”, es decir, “la menor pieza de funcionalidad que, al ser desarrollada, entrega valor tanto a la organización que la requiere como a la gente que la utiliza”. O se pueden clasificar las tareas por “Peso”, es decir, por el tiempo que le tomaría a un desarrollador resolverla, definiendo un peso máximo por tarea pasado el cual se realizaría otra descomposición. También se puede dividir por categoría, en caso de que la tarea involucre conocimientos específicos. Finalmente, el formato puede ser kanban, caso de uso o un user story. El método y formato particular elegido depende completamente del equipo de desarrollo.

3.2.6 Análisis de Riesgo

Ante un riesgo, existen dos elementos a balancear: “Es más barato abordar un riesgo de forma temprana” y “abordar un riesgo con la mayor información posible aunque implique aplazar el riesgo”. No sólo este balanceo es complejo, sino que, por otra parte, es preferible mantener el análisis de riesgo liviano porque puede generar basura. Un método liviano para analizar y decidir si conviene abordar el riesgo es con una matriz de probabilidad e impacto en donde se cruza un estimado de la probabilidad de que el riesgo se vuelva realidad con el impacto que tendría sobre el proyecto. Estos estimados pueden ser discretos (“Alto”, “Medio”, “Bajo”) y permitirían rápidamente realizar una priorización de las tareas. Se puede recurrir al juicio de expertos o a una votación en el equipo de desarrollo para obtener mayor información con respecto al riesgo.

3.2.7 Análisis de los 5 “Por Qué”

Este análisis busca encontrar la causa raíz de un problema. Se realiza preguntando “¿Por qué?” a un problema encontrado y luego preguntando “¿Por qué?” de nuevo a la respuesta entregada 4 veces. Los resultados de este análisis pueden servir también para la reunión retrospectiva.

3.3 Salidas:

3.3.1 Backlog/Kanban Actualizado

Ver 3.1.1. El Backlog o Kanban debe estar priorizado y con sus tareas en el formato acordado (Minimum Marketable Features, Peso, Categoría, Secuencia, etc.)

3.3.2 Lista de Recursos necesarios

Durante la reunión se discuten los recursos necesarios para completar exitosamente el proyecto y se realiza una lista de ellos con el objetivo de obtenerlos inmediatamente o siguiendo un calendario (Ver 3.1.4)

3.3.3 Lista de pruebas para el ciclo

Con el objetivo de obtener una alta calidad de desarrollo, se genera una lista de pruebas generales para validar la compleción del ciclo. Esta lista proviene de los requisitos que los interesados entregaron e, idealmente, conllevan a una prueba global de todas las piezas de software desarrolladas.

3.4 Consideraciones Adicionales

4. Reunión Diaria

Al igual que Scrum, se realiza una reunión diaria de menos de 15 minutos con todos los miembros del equipo, en donde cada uno resume el trabajo que realizó en día anterior, lo que planean realizar hoy y los posibles obstáculos que tengan. También se debe corroborar en esta reunión que la tabla esté en correcto estado.

Es muy útil revisar las estadísticas del proyecto (Burnup/Burndown) con el propósito de detectar atrasos y de tomar las medidas adecuadas frente a ello.

4.1 Entradas

4.1.1 Tabla de tareas/Kanban

Ver 3.1.1.

4.1.2 Radiador de información

Un radiador de información es un artefacto que entrega información sobre el estado actual del ciclo, indicando si una tarea se encuentra atascada en desarrollo o si existen demasiadas tareas por realizar para el resto del ciclo, instando una renegociación del ciclo o un aumento del esfuerzo.

4.1.3 Estadísticas del Proyecto

Ver 5.3.1.

4.1.4 Calendario de Recursos

Ver 3.1.4

4.2 Herramientas y Técnicas:

4.2.1 Juicio de Expertos

Ver 1.2.1. Un experto es consultado para determinar cómo superar los obstáculos del equipo de desarrollo.

4.2.2 Revisión del Desempeño

Con el objetivo de tener una visión realista sobre el estado actual del ciclo se realiza un análisis de las estadísticas del proyecto para detectar atrasos. Las estadísticas que más claramente

indican un posible atraso en el ciclo son el diagrama de Burnup y Burndown, pero otras estadísticas pueden también ayudar.

4.2.3 Compresión del Cronograma

Por medio de negociación o imposición, se buscan maneras de reducir el retraso ante el final del ciclo, sea por un aumento de esfuerzo (horas extra) o por una renegociación del alcance de la iteración.

4.3 Salidas:

4.3.1 Tabla de tareas/Kanban/Radiador de Información.

Ver 3.3.1 y 4.1.2

4.3.2 Lista de obstáculos a resolver.

Es una lista informal obtenida por el jefe de proyecto al preguntar a los miembros del equipo de desarrollo sobre los posibles obstáculos que pueden tener que les impidan realizar las tareas asignadas.

4.3.3 Lista de recursos necesarios.

Complementario a la lista del ítem anterior, es posible tener que modificar la lista de recursos necesarios entregada en 3.3.2

4.3.4 Actualización a las Estadísticas del Proyecto

Ver 5.3.1

4.4 Consideraciones adicionales.

Dado que el importante que la reunión sea corta y resumida, ésta, en lo posible, debe ser realizada de pie. Por otra parte, es posible incluir algún otro tópico a la reunión si así es convenido siempre y cuando se tenga en cuenta que la reunión debe ser corta.

En caso de utilizar una tabla Kanban es útil revisar las cargas y los atascos en el proceso de desarrollo y priorizar aquellas tareas que se encuentran atascadas.

5. Ejecutar Tareas

La ejecución de tareas se realiza con un sistema de pull en lugar de un push. Es decir, en vez de que el jefe de proyecto asigne tareas a los miembros del equipo de desarrollo, cada miembro se asigna una tarea en la medida de que tenga la capacidad de resolverla. El multitasking, el intentar realizar varias tareas al mismo tiempo, genera grandes retrasos en el desarrollo de software, mientras que el pulling garantiza que las tareas sean realizadas de forma secuencial, permitiendo al desarrollador concentrarse por completo en la resolución de la tarea.

En Kanban, una de las metodologías consideradas como Lean, cada área de trabajo posee un "Límite de trabajo concurrente", es decir, un máximo de tareas que puede ser realizado por área, lo cual energiza la capacidad de pulling y ayuda a detectar los cuellos de botella en el proceso de desarrollo.

Durante el desarrollo es también importante dejar registro de las tareas realizadas tanto en el ciclo actual como en los ciclos anteriores con el propósito de entregar trazabilidad así como permitir el cálculo de datos estadísticos sobre el desarrollo de software. En "Salidas" se mostrarán algunas de las estadísticas que se pueden utilizar.

En esta etapa, el jefe de proyecto tiene la responsabilidad de resolver los obstáculos detectados por el equipo de desarrollo durante la reunión diaria y obtener (o calendarizar) los recursos solicitados por el equipo de desarrollo.

Finalmente, también se implementan los cambios conversados durante la reunión retrospectiva.

5.1 Entradas:

5.1.1 Backlog/Kanban

Ver 3.1.1

5.1.2 Lista de obstáculos a resolver

Ver 4.3.2

5.1.3 Lista de Recursos necesarios

Ver 3.3.2

5.1.4 Calendario de Recursos

Ver 3.1.4

5.1.5 Lista de interesados

Ver 1.3.5

5.1.6 Lista de cambios al proceso de desarrollo

Ver 8.3.1

5.2 Herramientas y Técnicas:

5.2.1 Prototipos

Ante diferencias en la visión final del programa, es posible realizar más de un prototipo para que sea presentado durante la verificación de la entrega. Ver 2.2.4

5.2.2 Análisis de Documentos

Principalmente de las minutas de las reuniones. Ver 2.2.5

5.2.3 Contacto con los interesados/usuarios

Idealmente, al igual que Extreme Programming, se negocia la presencia de un usuario o cliente in situ. Dado que es difícil llegar a esa situación, ante dudas sobre la visión del usuario o los interesados, es recomendable contactarse con ellos. Se recomienda un contacto en persona o, en su ausencia, una comunicación con conexión remota para que el usuario pueda ver con facilidad el problema del desarrollador

5.2.4 Control de Calidad

La calidad es un tema clave en Lean. De nada sirve desarrollar software rápido si es de mala calidad. A continuación se explican elementos que ayudan a mejorar la calidad del software:

5.2.4.1 Generación constante de retroalimentación

La retroalimentación es un concepto que está presente en todas las etapas de Lean. Permite validar el trabajo realizado, desde la captura de requerimientos hasta el traspaso a producción. Es más importante aumentar la retroalimentación que adquirir aprobaciones formales por parte de los distintos actores en el desarrollo del proyecto. Ejemplos prácticas de retroalimentación:

- Testear el código tan pronto sea escrito (código defectuoso es más difícil de arreglar a futuro)
- Probar las ideas por medio de código en vez de documentación

- Desarrollar prototipos en lugar de documentar requerimientos
- Actualizar interfaces de sistemas antiguos para probar ideas en vez de construir un sistema completamente nuevo.

5.2.4.2 Calidad del equipo de trabajo:

El equipo de trabajo:

- Debe ser pequeño y con la expertise necesaria para abordar el proyecto.
- Debe poseer información suficiente sobre las tareas a realizar en cada iteración
- Debe tener los recursos necesarios para el desarrollo del proyecto garantizados y entregados a tiempo
- Debe tener libertad, soporte y habilidad para poder cumplir con los compromisos
- Debe poseer o crear un buen ambiente de desarrollo con herramientas tales como:
    - Automated Build Process
    - Testeo Automatizado
    - Estándares de código
    - Herramientas de control de versiones
    - Etc.

5.2.4.3 Calidad de Testing

Los tests realizados deben simular las condiciones en las cuales el sistema funcionará, por lo que también se debe comprobar continuamente la integración con el resto del sistema, en tandas pequeñas para facilitar la detección de errores y preferiblemente de forma automatizada. Estos tests se pueden definir de manera anticipada y es muy conveniente guardarlos y entregarlos al final del proyecto para permitir hacer pruebas sobre nuevos desarrollos realizados durante el periodo de mantención del sistema.

Es importante probar el desarrollo tanto desde el punto de vista técnico (“¿Funciona correctamente?”) como desde el punto de vista del negocio (“¿Es lo que el cliente pidió?”)

Es útil también desarrollar una aplicación de Testing que realice una prueba transversal del sistema, es decir, que, en cadena, revise varias o todas las funcionalidades del programa y compruebe la consistencia de su funcionamiento general

5.2.1.4 Calidad de Programación

Según Lean, el código implementado debe ser lo suficientemente flexible como para permitir modificaciones futuras, en donde habrá más información disponible sobre las necesidades de los interesados. Esto significa adoptar una serie de prácticas al programar:

- Colaboración directa entre miembros del equipo de desarrollo y traspaso continuo de información de diseño parcialmente completa.
- Uso de módulos para ocultar el comportamiento de una parte del sistema, cada uno con una responsabilidad única y definida.
- Uso de interfaces y separación entre interfaz e implementación.
- Uso de parámetros y abstracciones.
- Evitar repetición en el código.
- Evitar herramientas personalizadas (implican tomar decisiones de diseño muy tempranamente).
- Encapsular la variación, mantener las interfaces constantes y estables.
- Evitar implementar features adicionales "en caso de que se necesiten". Toda implementación adicional implica pruebas adicionales, mantención y riesgo.
- Desarrollar un entendimiento profundo del dominio.
- Desarrollar una capacidad de respuesta rápida.
- Preocuparse de refactorizar, arreglando el código actual por sobre agregar nuevas features al programa.

5.2.1.5 Integridad

5.2.1.5.1 Integridad Conceptual

Un sistema con integridad conceptual es un sistema que funciona como un todo fluido y cohesivo. Obtener esto implica un flujo constante, temprano y bidireccional de información en pequeñas porciones de manera informal y presencial. También implica una correcta división de las distintas capas de software, prototipado rápido y constante y utilización de estándares.

5.2.1.5.2 Integridad Percibida

Un sistema con integridad percibida es un sistema con un balance de función, usabilidad, dependencia y economía. Para lograr esto es necesario mantener una continua comunicación entre los distintos actores del proyecto para refrescar continuamente la visión que engloba todo el sistema. Para facilitar esta comunicación se deben realizar iteraciones cortas con tests de cliente (comprobar que lo implementado corresponde a lo que el cliente pide). Es conveniente desarrollar un lenguaje y un modelo mental común nacido de los clientes para entender el negocio.

Documentos y elementos tales como: Modelos de dominio, Glosarios y Casos de Uso pueden ayudar a crear esta visión global. Sin embargo, estos documentos deben ser constantemente actualizados, aunque, por otra parte, una ausencia de voluntad por actualizar un documento puede indicar que ese documento no entregaría valor a pesar de su actualización.

Durante las reuniones es útil comenzar a definir los tests que se utilizarán para probar la iteración, permitiendo, a futuro, recordar los detalles pactados.

5.1.2.6 Master Developers

El liderazgo en Lean gira en torno al "Master Developer", un developer apasionado por el proyecto que asume la responsabilidad del éxito del proyecto. Este rol no surge por imposición, sino que se aparece de forma orgánica cuando se le permite a los desarrolladores definir una solución propia y personal (es decir, "auto-organizada") al problema de los interesados. Estos Master Developers también suelen ser diseñadores con abundante experiencia y conocimiento del dominio (así como voluntad por aprender más).

5.3 Salidas

5.3.1 Estadísticas del proyecto: Tales como:

- Promedio de tareas completadas por ciclo.
- Tiempo promedio de compleción de una tarea.
- Tiempo promedio de espera de una tarea.
- Tasa de errores.
- Tiempo promedio por área de desarrollo: En caso de utilizar kanban, muestra la cantidad de tiempo por cada tipo de actividad
- Tasa de Tareas entregadas vs Tareas Comprometidas.
- Velocidad/Throughput: Cantidad de tareas realizadas por ciclo o día
- Diagrama de Flujo Acumulado: En caso de utilizar Kanban, muestra la evolución en la cantidad de tareas por columna (tipo de actividad) versus la fecha
- Diagrama de Burndown/Burnup : Muestra la evolución de la cantidad total de tareas pendientes o realizadas versus la cantidad total de días disponibles para terminarlas dentro del ciclo

5.3.2 Entrega de Ciclo (Verificados)

Es la entrega pactada al inicio del ciclo, la cual puede incluir software, hardware, implementaciones de hardware, configuraciones y documentación entre otros. Esta entrega

debe estar verificada, es decir, gracias a las pruebas realizadas se puede garantizar que el software funciona correctamente.

5.3.3 Tabla de Tareas/Kanban Actualizada

Ver 3.1.1.

5.3.4 Calendario de recursos Actualizado

En el caso de que se entreguen recursos solicitados. Ver 3.1.4.

5.3.5 Lista de Recursos Actualizada

En el caso de que se elimine la necesidad de monitorear cierto recurso o que el recurso sea disponible a libre disposición por el equipo de desarrollo. Ver 3.3.2.

6. Validación de la Entrega

Al final de cada ciclo, el equipo presenta a los interesados el trabajo realizado durante el ciclo. En esta presentación se muestra el trabajo desarrollado en funcionamiento, con el objetivo de validar la entrega y que los interesados puedan explorar la solución desarrollada.

Es aconsejable que esta reunión sea informal, es decir, que no se utilice una presentación pre-construida, sino que se explore activamente la entrega de la iteración.

Finalmente, en este proceso o durante el traspaso es importante realizar la documentación necesaria para una fácil mantención. Realizar esta documentación de forma tardía permite que ésta contenga los datos reales de la implementación y facilita su generación

6.1 Entradas

6.1.1 Kanban/Backlog completado

Con el propósito de entregar trazabilidad a los entregables, se puede mostrar en la reunión la tabla de tareas acordada en la reunión de planificación de ciclo. Mostrar el Kanban o el Backlog permite que los interesados vean de modo transparente el estado actual del ciclo. Ver 3.1.1

6.1.2 Entrega de ciclo (Verificados)

Ver 5.3.2.

6.1.3 Estadísticas de proyecto:

Las estadísticas que se pueden entregar en esta reunión pueden ayudar a definir con más exactitud la carga óptima de trabajo del equipo de desarrollo y dar un contexto tangible a la entrega del ciclo.

6.1.4 Activos de los Procesos de la Organización

Ver 1.1.2

6.1.5 Minuta de Reuniones

Ver 2.3.2. Es importante notar que las minutas deben ser utilizadas como apoyo a la conversación, no como herramienta para encontrar “culpabilidad”.

6.2 Herramientas y Técnicas:

6.2.1 Inspección de la entrega

La entrega debe ser revisada por los interesados y, en caso de ser necesario, por los demás usuarios del sistema. Se debe proporcionar acceso a los usuarios para realizar esta tarea y su retroalimentación puede ser obtenida por encuestas (2.2.3)

6.2.2 Técnicas Grupales de Creatividad:

Ver 2.2.2

6.2.3 Encuestas: En caso de tener que entregar el programa a una gran cantidad de usuarios, es recomendable entregar encuestas o recopilar su retroalimentación por medio del envío de reseñas. Ver 2.2.3.

6.3 Salidas

6.3.1 Entrega de ciclo validada

La validación de la entrega quiere decir que la entrega funciona correctamente y su funcionamiento cumple las necesidades de los interesados

6.3.2 Lista de Cambios a entregar en la próxima reunión de planificación

En caso de tener que realizar modificaciones a la entrega del ciclo, éstas son registradas y presentadas en la próxima planificación de ciclo

6.3.3 Actualización a los documentos del proyecto

En particular se actualiza la lista de reuniones con los hitos entregados para reflejar el estado actual

6.3.4 Documentos de Mantención

Ver 7.3.3

7. Realizar Traspaso

Dependiendo de las características del proyecto, este proceso se puede realizar después de cada iteración o al final del proyecto. Consiste en traspasar la versión desarrollada y validada del sistema a producción para habilitar su uso por los usuarios. Comúnmente, este proceso se realiza al final del proyecto, sin embargo, es posible que el cliente quiera aprovechar una oportunidad de negocio o cumplir expectativas externas (por ejemplo, legales). Existe un riesgo al realizar un traspaso después de cada iteración aunque se podría contraargumentar que el traspaso tardío también trae un riesgo pues esconde posibles problemas de compatibilidad hasta el final del proyecto, momento en el cual el costo de realizar cambios es muy elevado. Por lo tanto, este elemento y el tiempo de su realización quedan definidos por la naturaleza del proyecto.

El traspaso de cada elemento puede ser contemplado dentro de la tabla de tareas para proveer completa trazabilidad y cadena de responsabilidad desde su captura hasta su implementación final en producción

Durante este proceso (o el anterior, dependiendo del proyecto) se deben producir los documentos para la mantención. Éstos deben ser acordados por el equipo de desarrollo y el equipo de mantención y pueden incluir modelos (de arquitectura, clases, etc.), casos de uso, descripciones de alto nivel, manuales de usuario, etc. Es crucial, sin embargo, entregar todos los tests realizados durante la iteración

7.1 Entradas

7.1.1 Elementos de la entrega validados / Sistema desarrollado

Ver 6.3.1. Como este proceso puede realizarse por ciclo o por proyecto, la cantidad de elementos de este ítem es variable

7.1.2 Lista de Pruebas producida durante el desarrollo del proyecto

Ver 3.3.3 Es importante en esta instancia realizar pruebas transversales como se menciona en el ítem 5.4.2.3

7.1.3 Activos de Procesos de la Organización

Ver 1.1.2

7.1.4 Tabla de Tareas/Kanban

Ver 3.1.1

7.2 Herramientas y Técnicas:

7.2.1 Testing

Es indispensable probar el correcto funcionamiento del sistema en producción y, para lograr eso, es posible utilizar los tests producidos durante el desarrollo (7.1.2).

7.2.2 Marcha Blanca

Una manera rápida de comprobar el correcto y apropiado funcionamiento del sistema desarrollado es realizan una marcha blanca, es decir, ejecutar el sistema con usuarios reales pero sin guardar de forma definitiva los datos o resultados del ejercicio. En caso de que el programa sea una modernización, es útil pedir a los usuarios que utilicen ambos para luego verificar las diferencias entre ambos sistemas.

7.3 Salidas

7.3.1 Sistema en producción

El sistema entregado y funcionando en producción.

7.3.2 Documentos de cierre del proyecto

Se refiere a la documentación que transfiera oficialmente el sistema al cliente y gatille el cierre legal del proyecto.

7.3.3 Documentación para mantención

Son los documentos que facilitarán la mantención del sistema, cuya implementación específica debe ser acordada entre los miembros del equipo de desarrollo y el equipo de mantención. Estos pueden incluir Casos de Uso, modelos de diversa índole, descripciones de alto nivel, cartas kanban, manuales de usuario, descripciones de la arquitectura, etc.

8. Reunión Retrospectiva

En esta reunión se discuten técnicas o prácticas del proceso de desarrollo con el objetivo de mejorar el proceso. Es crucial mejorar continuamente en Lean, agregando o quitando reglas, procedimientos o herramientas al proceso de desarrollo.

La reunión se realiza al terminar una iteración del proyecto o cada cierta cantidad de tiempo previamente acordada (en caso de grupos de mantención o multi-proyecto). Participan el Jefe de Proyecto y todo el equipo de desarrollo. En Herramientas y Tecnicas se conversará de algunos de los ítemes que pueden tener la reunión.

8.1 Entradas

8.1.1 Estadísticas del Proyecto o del Periodo

Ver 5.3.1

8.1.2 Activos de los Procesos de la Organización

Ver 5.3.2

8.1.3 Entregas del Ciclo Validados

Ver 6.3.1

8.1.4 Diagrama del proceso de desarrollo

Es el diagrama que detalla el proceso de desarrollo específico de cada organización

8.2 Herramientas y Técnicas

8.2.1 Value Stream Mapping:

Consiste en generar y revisar el diagrama con el proceso de desarrollo para encontrar los tiempos y puntos que entregan valor al cliente. Muchas actividades en el proceso de desarrollo pueden ser innecesarias o fáciles de sustituir por otras que entreguen mayor valor. Por ejemplo, es útil ver la cantidad de aprobaciones necesarias para realizar un cambio a un sistema o el proceso por el cual se acepta un desarrollo en caso de ser un departamento de una empresa.

8.2.2 Eliminación de Basura:

Una filosofía importante en Lean es la de eliminación de basura, donde basura es definido como todo aquello que no aporte valor al cliente. El libro de Lean habla de “Los 7 Desechos del Desarrollo de Software” los cuales son:

- Trabajo parcialmente completado: El software sólo entrega valor directamente cuando está en producción, aumentando el costo y riesgo si se entrega tarde
- Procesos Adicionales: Eliminar procesos burocráticos y documentación innecesaria, buscar establecer documentación estándar, artefactos livianos y planillas.
- Features Adicionales: Cada feature adicional no solicitada representa un costo y un riesgo
- Multitasking: Genera retrasos y menor calidad
- Esperas: Buscar reducir todo tiempo de espera de cada actor del proceso
- Movimiento: Buscar la co-locación del equipo de software y reducir el tiempo que toma alcanzar al cliente
- Defectos: Buscar arduamente los defectos tempranamente para evitar pagar un costo de arreglo más alto después.

8.2.3 Amplificación de conocimiento y retroalimentación

Si el software producido es de baja calidad técnica, buscar prácticas para verificar el software (Por ejemplo: Programación a pares, “Test Oriented Development”, Integración Continua, Estándares de código). Si el software es rechazado por el cliente (error conceptual) buscar agregar prácticas donde el cliente verifique más seguido y donde el equipo pueda generar más información sobre el negocio (cliente/usuario in-situ, mayor frecuencia de reuniones, mayor presencia del equipo en reuniones)

8.2.4 Decisión Responsablemente Tardía

Conforme se va avanzando el proyecto, el equipo de desarrollo encuentra más información sobre el negocio del cliente lo que puede significar retractarse sobre decisiones tomadas previamente durante el proceso de desarrollo. Decidir tarde responsablemente implica tomar decisiones de forma tardía, cuando exista la máxima cantidad de información disponible. Para lograr eso, se debe, también generar un diseño y sistema flexible donde decisiones tempranas no impacten sobre cambios futuros. Algunas de estas técnicas se encuentran en la sección 4.2.1.4.

8.2.5 Búsqueda de cuellos de botella y Análisis de Estadísticas

Cada proceso tiene al menos un cuello de botella. Si se utiliza una tabla Kanban se facilita la búsqueda de ese cuello de botella, permitiendo aumentar la velocidad de desarrollo.

Las estadísticas reunidas pueden indicar la dirección de ciertos problemas en el desarrollo. Una alta tasa de error indica que existen deficiencias técnicas, mientras que alta tasa de rechazo muestran problemas conceptuales. Tiempos de espera indican cuellos de botella, multitasking o baja cantidad de elementos entregados al final de la iteración indica sobre-compromiso. Es importante estudiar las estadísticas y encontrar formas de mejorarlas para la próxima iteración.

8.2.6 Fortalecimiento del Equipo

Algunos de los elementos que Lean sugiere para fortalecer al equipo son:

- Autodeterminación: Permitir que el equipo encuentre la mejor forma de realizar su trabajo. Cambiar el rol de las jefaturas de ordenar y dirigir directamente el proyecto a evitar interferencias externas, entregar recursos y protección.
- Propósito: Comenzar con un propósito claro y lograble. Permitir que el cliente tenga fácil acceso al cliente
- Pertenencia: Fomentar el respeto y la honestidad en el equipo. Evitar dividir en “ganadores” y “perdedores”.
- Seguridad: Evitar la mentalidad “cero defectos” y permitir que los desarrolladores se equivoquen para fomentar la experimentación
- Progreso: Utilizar herramientas o prácticas que den la sensación de que existe un progreso constante. Celebrar hitos importantes.
- Liderazgo: Fomentar la formación de Master Developers
- Expertise: Fomentar la presencia de comunidades de expertise, gente gurú con profundo conocimiento sobre temas útiles en el proceso de desarrollo.

También se puede recurrir a la educación para aumentar la competencia del equipo. Si bien los cursos pueden traer beneficios hay que tener en cuenta los siguientes elementos:

- Los cursos sobre el uso de herramientas nuevas deben servir para el desarrollo de proyectos. Esto significa que se debe verificar la utilidad de las herramientas probándolas internamente primero.
- Los cursos sobre prácticas y manejo de proyectos deben estar alineados al estilo de desarrollo del equipo. Es decir, si el equipo utiliza métodos ágiles, los cursos de métodos pesados y burocráticos como CMM pueden ser nocivos.

8.2.7 Análisis de los 5 “Por qué”.

Ver 4.2.2.

8.3 Salidas:

8.3.1 Diagrama del proceso de desarrollo actualizado

Ver 8.1.4

8.3.2 Lista de cambios al proceso de desarrollo

Es una lista de cambios al proceso de desarrollo, los cuales serán implementados con ayuda del jefe de proyecto durante la próxima iteración

8.3.3 Actualización a los activos de los procesos de la organización

Ver 1.1.2

Anexo 3: Cuestionarios

Para cada pregunta del cuestionario, ciertas respuestas suman puntos para evaluar el total de Agilidad o grado Lean del proceso. Se asume que un proceso es Lean si tiene un puntaje mayor a 75%.

Cuestionario Práctico

Proceso 1: Generar Acta (5/5)

1. ¿Es el proyecto “pequeño”? (Sí: +1. No: +0)
2. Si es externo: ¿Qué tipo de contrato se utiliza? (+1 Si el contrato es Tiempo-Materiales o muestra consideración por una negociación de alcance. De lo contrario +0)
3. ¿Están los interesados comprometidos con el éxito del trabajo? (Sí: +1. No: +0)
4. ¿Es el dueño del proceso fácilmente accesible? (Sí: +1. No: +0)
5. ¿Es el usuario del sistema fácilmente accesible? (Sí: +1. No: +0)

Proceso 2 Planificar Ciclo (8/8):

6. ¿Es un método iterativo? ¿Se realiza varias veces la captura de requerimientos? (Sí: +1. No: +0)
7. ¿Hay un proceso de negociación, conversación o descubrimiento del alcance? (Sí: +1. No: +0)
8. ¿Es el alcance fijo? ¿Puede cambiarse o refinarse conforme avanza el proyecto? (Sí: +1. No: +0)
9. ¿Existe un proceso de priorización? (Sí: +1. No: +0)
10. ¿Hay un énfasis en el prototipado para mostrar ideas? (Sí: +1. No: +0)
11. ¿Se reúnen los requisitos de los usuarios? (Sí: +1. No: +0)
12. ¿Se reúnen las condiciones de test de los interesados? (Sí: +1. No: +0)
13. ¿Se prefiere la documentación a la negociación cara a cara? (Sí: +0. No: +1)

Proceso 3 Reunión Diaria de Planificación (3/3):

14. ¿Existe un proceso para abordar el riesgo?
    a. ¿Es el proceso de medir el riesgo un aporte o una interferencia para el proyecto? (Si es una interferencia:+0. Si no hay un proceso: +1. Si hay un proceso y es ágil o ligero: +2)
15. ¿Existe un proceso sobre la descomposición de requisitos? (Sí: +1. No: +0)

Proceso 4 Reunión Diaria (5/5):

16. ¿Se realiza una reunión diaria? (Sí: +1. No: +0)
17. ¿Existe algún tipo de radiador de información que muestre de forma transparente el estado del proyecto? (Sí: +1. No: +0)
18. ¿Está informado el jefe de proyecto sobre los obstáculos que tiene el equipo de desarrollo? (Sí: +1. No: +0)
    a. ¿Realiza el jefe de proyecto el trabajo de remover los obstáculos del equipo de desarrollo? (Sí: +1. No: +0)
19. ¿Es el estado actual del proyecto, con el desglose de qué persona trabaja en qué tarea visible? (Sí: +1. No: +0)

Proceso 5 Ejecutar Tareas (6/6):

20. ¿Es la ejecución de tareas asignada por el jefe de proyecto o por el propio equipo de desarrollo? (Jefe de Proyecto: +0. Equipo de Desarrollo +1)
21. ¿Existe énfasis en reducir el multitasking? (Sí: +1. No: +0)
22. ¿Se recopilan estadísticas sobre el desarrollo del proyecto? (Sí: +1. No: +0)
    a. ¿Miden las estadísticas el desempeño del equipo de trabajo o el desempeño individual? (Miden desempeño del equipo: +1. Miden desempeño individual: +0)
23. ¿Existe una política de control de calidad? (Sí: +1. No: +0)

Sobre las prácticas (19/19):

24. ¿Se prueba el código tempranamente? (Sí: +1. No: +0)
25. ¿Existe una preferencia por los prototipos por sobre la documentación? (Sí: +1. No: +0)
26. ¿Es el equipo “pequeño” (menos de 10 personas)? (Sí: +1. No: +0)
27. ¿Tiene libertad el equipo de desarrollo para elegir la forma en la cual cumplir con el requerimiento de los interesados? (Sí: +1. No: +0)
28. ¿Existe un buen ambiente de desarrollo?
    a. ¿Automated Build Process? (Sí: +1. No: +0)
    b. ¿Testing automatizado? (Sí: +1. No: +0)
    c. ¿Estándares de código? (Sí: +1. No: +0)
    d. ¿Control de versiones? (Sí: +1. No: +0)
    e. ¿Testing Transversal? (Sí: +1. No: +0)
    f. Otros (+0.5 por c/u)
29. ¿Se utilizan buenas prácticas para permitir un código y diseño flexibles?
    a. ¿Módulos? (Sí: +1. No: +0)
    b. ¿Interfaces que oculten la implementación? (Sí: +1. No: +0)
    c. ¿Abstracciones? (Sí: +1. No: +0)
    d. ¿Encapsulación? (Sí: +1. No: +0)

    e. ¿Código minimal (No implementar más allá de lo que sea solicitado)? (Sí: +1. No: +0)
    f. ¿Refactorización? (Sí: +1. No: +0)
30. ¿Existe una continua comunicación entre los distintos participantes del proyecto para definir una visión común? (Sí: +1. No: +0)
31. ¿Hay un lenguaje común entre los distintos participantes del proyecto? (Sí: +1. No: +0)
32. ¿Existe un Master Developer (un desarrollador que voluntariamente asume responsabilidad del proyecto con conocimiento del dominio y experiencia)? (Sí: +1. No: +0)

Proceso 6 (3/3):

33. ¿Existe una validación de los ciclos por parte de los interesados? (Sí: +1. No: +0)
34. ¿Prueban los usuarios la entrega? (Sí: +1. No: +0)
35. ¿Es transparente el avance del proyecto para los interesados? (Sí: +1. No: +0)

Proceso 7 (4/4):

36. ¿Se realiza un traspaso constante del desarrollo a producción? (Práctica recomendada pero no obligatoria) (Sí: +1. No: +0)
37. ¿Se entregan las pruebas realizadas durante el desarrollo del proyecto/iteración? (Sí: +1. No: +0)
38. ¿Qué documentos se entregan al final del proyecto para realizar las mantenciones? (+0.5 por c/u)
39. ¿Existe trazabilidad entre los requerimientos del usuario y la pieza de software implementada en producción? (Sí: +1. No: +0)
40. ¿Son estos documentos solicitados explícitamente por el equipo de mantención? (Sí: +1. No: +0)

Proceso 8 (14/14):

41. ¿Existe una instancia de retrospección sobre el proceso de desarrollo? (Sí: +1. No: +0)
    a. ¿Incluye a los desarrolladores? (Sí: +1. No: +0)
    b. ¿Se llevan a cambio sus sugerencias? (Sí: +1. No: +0)
42. ¿Está el proceso de desarrollo mapeado? (Sí: +1. No: +0)
43. ¿Hay una preocupación por entregar el máximo valor al cliente rápidamente? (Sí: +1. No: +0)
    a. ¿Hay una preocupación por eliminar aquellas prácticas, procesos o herramientas que no entregan valor al cliente? (Sí: +1. No: +0)
44. ¿Se buscan prácticas para reducir la cantidad de defectos? (Sí: +1. No: +0)
45. ¿Hay un proceso definido de toma de decisiones? (Sí: +1. No: +0)

    a. ¿Se toman las decisiones en el momento correcto? (En el sentido Lean, es decir, en el último momento responsable y con la máxima información disponible)
46. ¿Se analizan las estadísticas de los proyectos con el propósito de encontrar mejoras? (Sí: +1. No: +0)
47. ¿Existe una mentalidad de cero defectos? (Sí: +0. No: +1)
48. ¿Hay una sensación de avance constante? (Sí: +1. No: +0)
49. ¿Existen comunidades de expertise dentro de la organización? (Sí: +1. No: +0)
50. ¿Los cambios a futuros engloban mayor agilidad o mayor formalidad? (Agilidad: +1. Formalidad: +0)

Cuestionario Teórico

Principio 1: Eliminar basura (16/16)

- ¿Hay preocupación por reducir la cantidad de trabajo parcialmente completado? Es decir ¿Se busca reducir la cantidad de tiempo entre que una tarea es completada y es pasada a producción? (Sí: +1. No: +0)
- ¿Qué medidas se toman para lograr esto? (+0.5 por c/u)
- ¿Podría usted aseverar que todas las actividades en el proceso de desarrollo entregan valor al cliente? (Sí: +1. No: +0)
- ¿Es la documentación que se produce necesaria para el cliente? (Sí: +1. No: +0)
- ¿Son todos los procesos realizados necesarios? (Ejemplo: ¿hay un priorización útil? ¿Hay una estimación útil? Etc.) (Sí: +1. No: +0)
- ¿Se producen solamente las features que el cliente solicita cuando el cliente solicita? ¿O se producen features por adelantado? (Sólo solicitadas: +1. Features por adelantado: +0)
- ¿Existe un énfasis en reducir el multitasking de los desarrolladores? (Sí: +1. No: +0)
- ¿Existen procedimientos o actividades que buscan reducir la cantidad de interferencia a los desarrolladores? (Sí: +1. No: +0)
- ¿Existen esperas entre que el usuario realiza una solicitud y su comienzo de trabajo? (Sí: +1. No: +0)
    - (si las hay) ¿A qué se deben estas esperas?
- ¿Es el cliente fácilmente accesible? (Sí: +1. No: +0)

- ¿Es el usuario fácilmente accesible? (Sí: +1. No: +0)
- ¿Es el dueño del proceso fácilmente accesible? (Sí: +1. No: +0)
- ¿Cuáles son las políticas o prácticas de testing que se utilizan? (ejemplos: testing automatizado, testing por el cliente, tester dedicado, unit testing, client unit testing, spanning applications, ambiente de test, revision de pares, etc.) (+0.5 por herramienta)
- ¿Existen actividades de gestión de proyectos (Por ejemplo: monitoreo y priorización)?
    - ¿Son éstas un aporte al portafolio del proyecto? (No hay: +1. Hay y son un aporte: +1. Hay y no son un aporte: +0)
- ¿Se encuentra el proceso de desarrollo mapeado? (Sí: +1. No: +0)

Principio 2: Amplificar Conocimiento (7/7)

- ¿Cuáles son las herramientas o procesos utilizados para reunir, explicitar, refinar y elicitar los requerimientos del usuario?
    - (ejemplos: documentación, acuerdos escritos, cambios controlados, documentos de trazabilidad, iteraciones, obtención de pruebas de cliente, prototipado, separación de capas, etc.) (+0.5 por prácticas ágiles)
- ¿Es el equipo de trabajo pequeño, cohesivo y con la expertise necesaria? (Sí: +1. No: +0)
- ¿Tiene éste la libertad de decidir cómo cumplir los requerimientos? (Sí: +1. No: +0)
- A grandes rasgos, ¿Qué herramientas o prácticas de apoyo del desarrollo se encuentran en uso?
    - (Ejemplos: Testing automatizado, estándares de código, programación a pares, control de versiones, ambientes de desarrollo, etc.) (+0.5 por herramienta)
- ¿Cómo se reúne la retroalimentación del cliente? (reuniones, tests pre-armados, tests del cliente) (+0.5 por práctica)
- ¿Cómo se reúne la retroalimentación del usuario? (Si hay método: +1. Si no hay método: +0)

Principio 3: Decidir lo más tarde posible (5/5)

- ¿Consideraría usted que los programas o sistemas desarrollados son flexibles (de fácil modificación durante el desarrollo y mantención)? (Sí: +1. No: +0)
- ¿Existen prácticas para generar esta flexibilidad? (Sí: +1. No: +0)
- ¿Es común que usted se encuentre en la situación de tener que realizar cambios profundos a los sistemas desarrollados en un proyecto avanzado? (Sí: +0. No: +1)
- ¿Es común que usted tome decisiones tempranas sobre el proyecto sin preguntar al cliente? (Sí: +0. No: +1)
- ¿Comparte usted información con el equipo de desarrollo parcialmente completa? (¿espera usted a tener la documentación o imagen clara o prefiere experimentar tempranamente?) (Sí: +1. No: +0)

Principio 4: Entregar lo más rápido posible (4/4)

- ¿Encuentra usted que los clientes están satisfechos con su velocidad de desarrollo? (Sí: +1. No: +0)
    - En caso que sí, ¿A qué cree que se debe esto?
- ¿Empuja usted el trabajo hacia el equipo de desarrollo o son estos los que tiran el trabajo? (Empujar: +0. Tirar: +1)
- ¿Qué herramientas o prácticas existen para monitorear el trabajo diario? (ejemplos: radiadores de información, tablas kanban, reuniones diarias) (Si hay herramientas: +1. Si no: +0)
- En orden de prioridad, si hay tiempo libre por algunos miembros del equipo de desarrollo usted cree que éstos deberían: (Si tomar nuevos proyectos es prioridad: +0. Si Apoyar el término o tomarse el tiempo libre es prioridad: +1.)

    1. Apoyar el término de proyectos
    2. Tomar nuevos proyectos
    3. Tomarse el tiempo libre

Principio 5: Fortalecer al Equipo (8/8)

- ¿Encuentra usted que el equipo de desarrollo está motivado? (Sí: +1. No: +0)

- ¿Encuentra usted que el equipo de desarrollo tiene la libertad necesaria? (Sí: +1. No: +0)
- ¿Encuentra usted que existe un sentimiento de pertenencia al equipo de desarrollo? (Sí: +1. No: +0)
- ¿Siente usted un sentimiento de progreso con respecto al trabajo del equipo? (Sí: +1. No: +0)
- ¿Es común que un desarrollador en sus proyectos tome el liderazgo y se haga responsable por el éxito del proyecto? (es decir, se siente responsable y comprometido personalmente en el éxito, en contraposición a simplemente obedecer las órdenes) (Sí: +1. No: +0)
- ¿Cómo es la comunicación dentro del equipo? (Cara a cara e informal: +1. Por medio de documentos: +0. Correo o teléfono como medio principal: +0.5)
    - ¿Y entre clientes y equipo? (Cara a cara e informal: +1. Por medio de documentos: +0. Correo o teléfono como medio principal: +0.5)
    - ¿Y entre usuarios y equipo? (Cara a cara e informal: +1. Por medio de documentos: +0. Correo o teléfono como medio principal: +0.5)

Principio 6: Incluir Integridad (4/4)

- ¿Cómo se suele propagar la información dentro del proyecto? (Elaborar. Ejemplos: por documentos, videoconferencia, mail, teléfono, cara a cara) (Cara a cara: +1. Documentos: +0. Otros: +0.5)
- ¿Cree usted que existe una visión común entre los diversos actores del proyecto? (¿Comparten los desarrolladores, el cliente y el usuario la visión y el lenguaje del sistema?) (Sí: +1. No: +0)
- ¿Existen formas de saldar la deuda técnica? (¿Cómo sabe cuándo es necesario rehacer parte del sistema?) (Sí: +1. No: +0)
- ¿La implementación empieza antes o después de que los requerimientos estén fijados? (Antes: +1. Después: +0)

Principio 7: Ver el todo (4/4)

- ¿Existe una práctica de análisis de causales? (ejemplo: 5 porqués, Diagrama de Pareto) (Sí: +1. No: +0)

- ¿Qué herramientas de mediciones se encuentran en uso? (+0.5 por herramienta)
- ¿Miden estas performance grupal o individual? (Grupal: +1. Individual: +0)
- ¿Cuál tipo de contrato se suele utilizar? (precio fijo, tiempo y materiales, multi-etapa, costo objetivo, fecha objetivo, beneficios compartidos) (descartar si el equipo de desarrollo es interno) (Tiempo y Materiales, Beneficios Compatidos o Multi-etapa con alcance negociable: +1. Otros: +0)
- ¿Considera usted que es el tipo de contrato adecuado? (Sí: +1. No: +0)

Anexo 4: Matriz Complejidad Tamaño

Se recomienda que el tamaño del proyecto sea pequeño. El chaos report captura este concepto en el "Size and Complexity Chart" y los vincula al riesgo del proyecto (2). En el documento original, el apartado "costo" del proyecto corresponde al costo de labor de trabajo estadounidense[2]. Dado que este costo es significativamente mayor al costo chileno[3], se ajustó este valor al sueldo de desarrolladores de software chilenos. El tamaño total es la sumatoria de tamaños redondeado hacia arriba.

Tamaño: Costo	
Debajo de 223 Millones de Pesos	Tamaño +0.5
Entre 223 y 668 Millones de Pesos	Tamaño +1
Entre 668 y 1337 Millones de Pesos	Tamaño +1.5
Entre 1337 y 2230 Millones de Pesos	Tamaño +2
Más de 2230 Millones de Pesos	Tamaño +2.5

Tamaño: Equipo	
Menos de 6 personas/meses	Tamaño +0.5
Entre 6 a 12 personas/meses	Tamaño +1
Entre 12 y 24 personas/meses	Tamaño +1.5
Entre 24 y 50 personas/meses	Tamaño +2
Más de 50 personas	Tamaño +2.5

[2] http://money.usnews.com/careers/best-jobs/software-developer/salary contiene el sueldo promedio del año 2015

[3] IT-Hunter, VI Estudio Público de Sueldos TIC en Chile (2016)

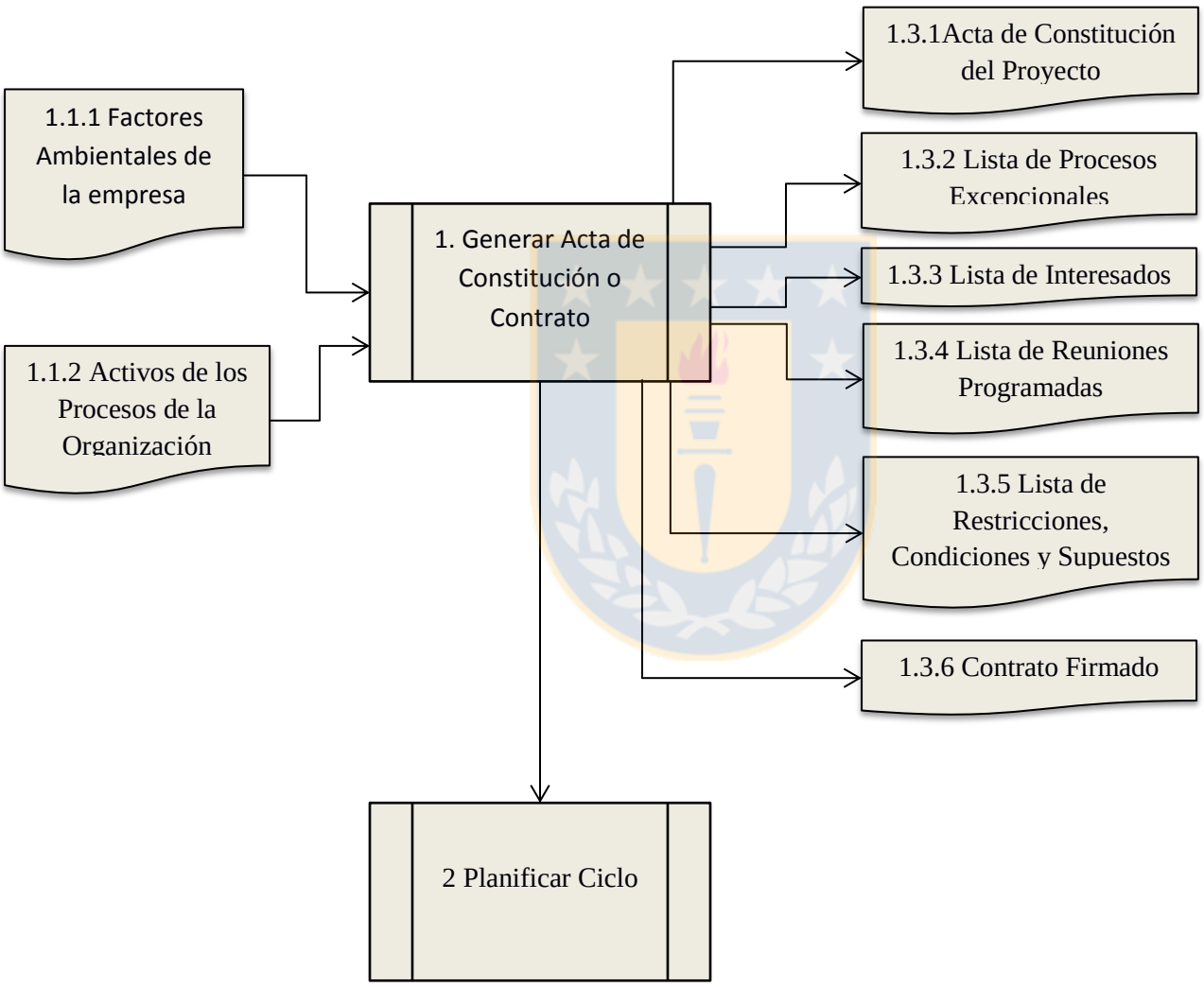
Complejidad	
Base	Complejidad = 1
Nuevo terreno	Complejidad + 1
Requerimientos confusos	Complejidad + 1
Equipo distribuido en varios lugares	Complejidad + 1
Interesados distribuidos en varios lugares	Complejidad + 1
Diversos objetivos de usuario	Complejidad + 1
Personal no cooperativo	Complejidad + 2
Interesados no cooperativos	Complejidad + 3

		Complejidad				
		C1	C2	C3	C4	C5
Tamaño	T1	100	250	400	550	700
	T2	175	325	475	625	775
	T3	250	400	550	700	850
	T4	325	475	625	775	925
	T5	400	550	700	850	1000

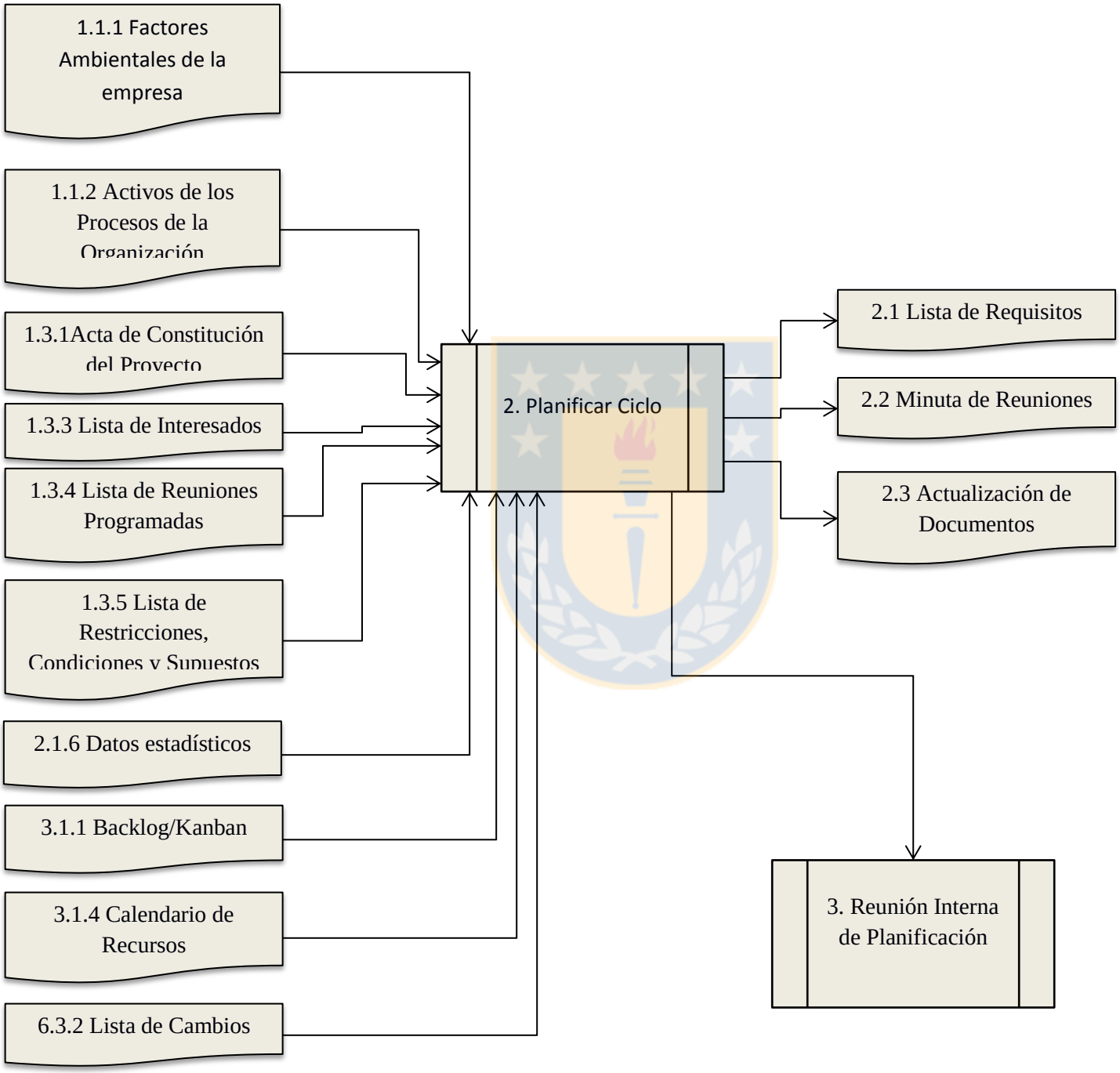
El área definida en gris corresponde a los proyectos pequeños y es el área recomendada tanto para la aplicación de Lean como para maximizar el éxito de un proyecto según el Reporte del Caos.

Anexo 5: Diagramas de procesos

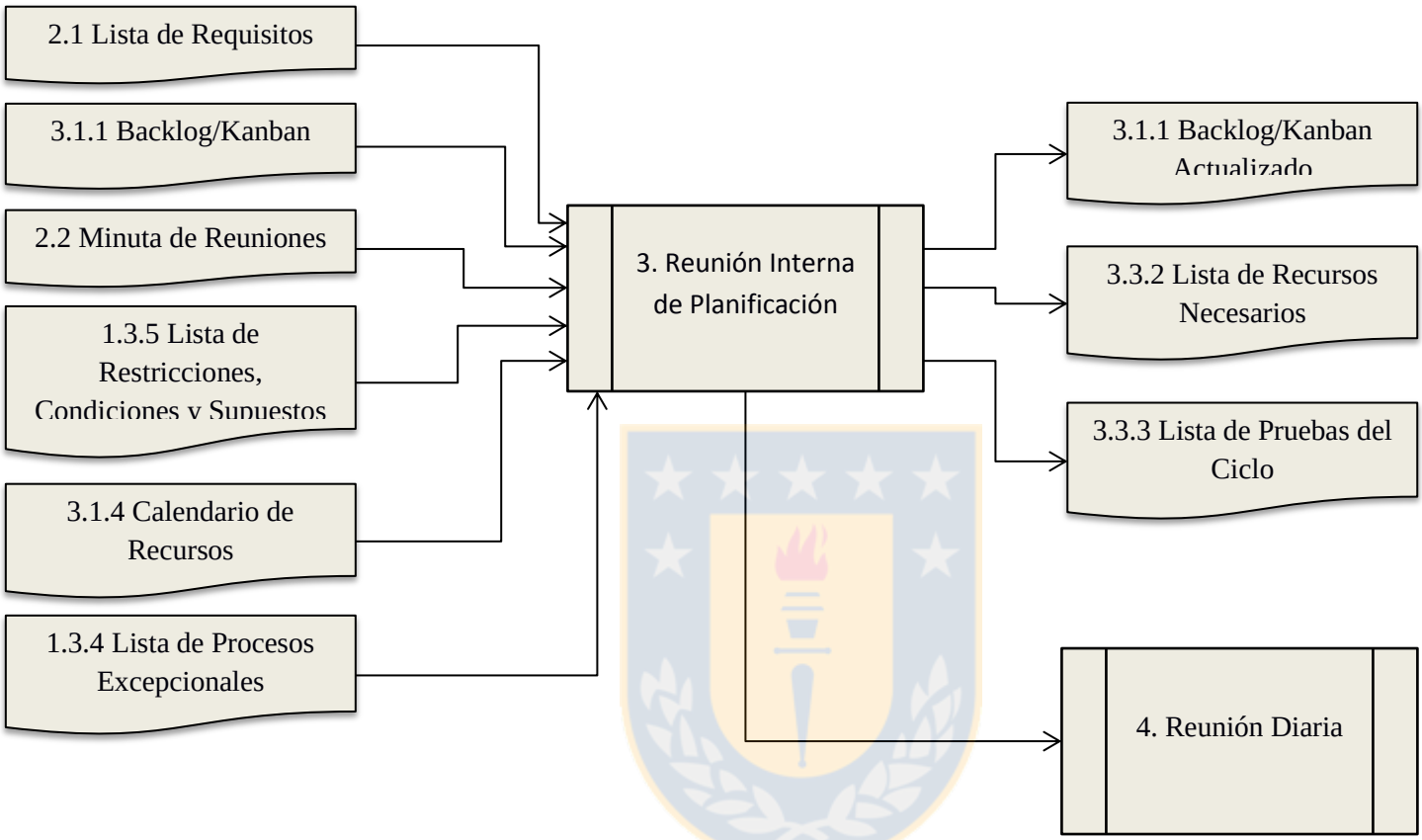
Proceso 1: Generar Acta de Constitución o Contrato



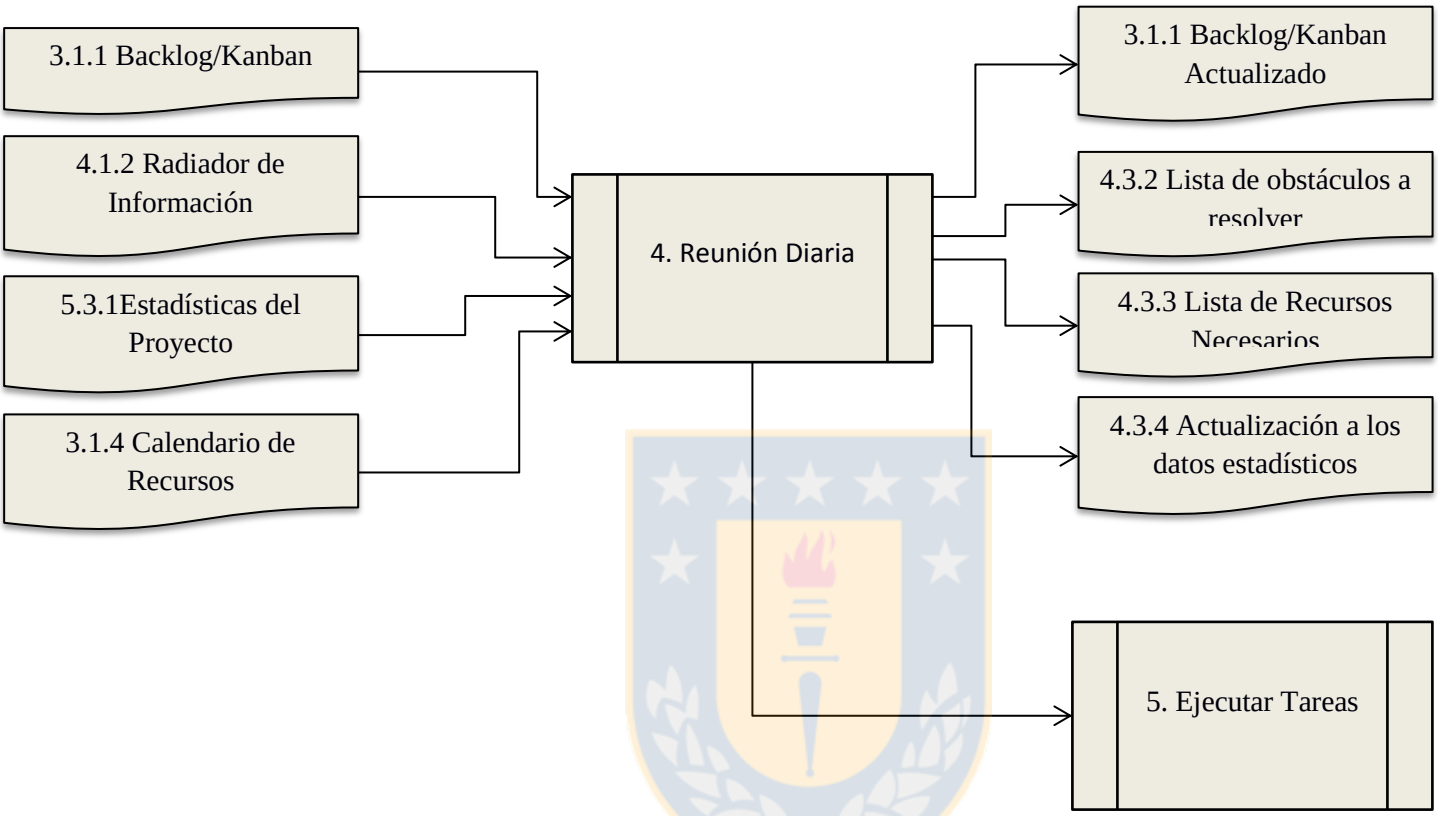
Proceso 2: Planificar Ciclo



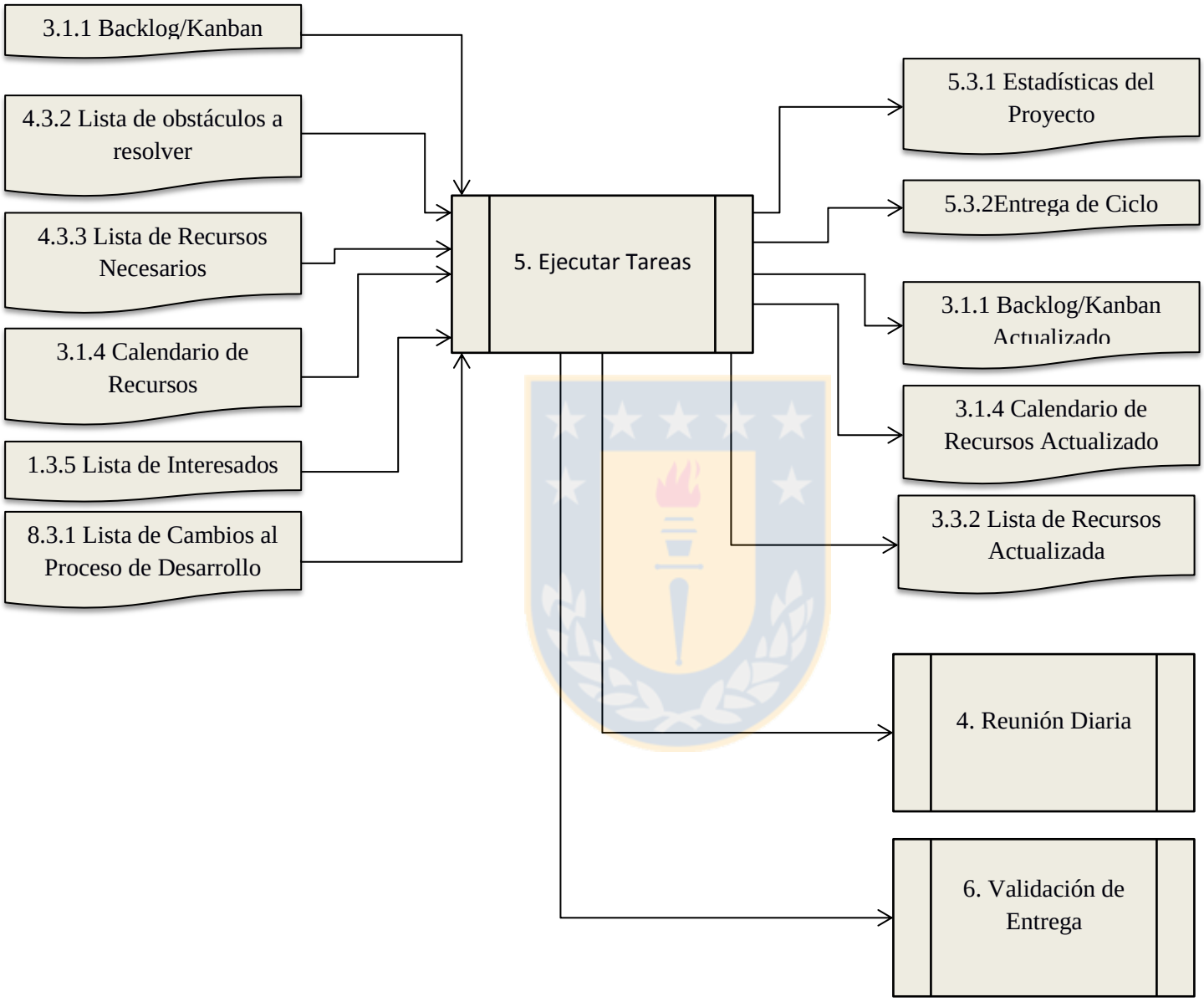
Proceso 3: Reunión Interna de Planificación



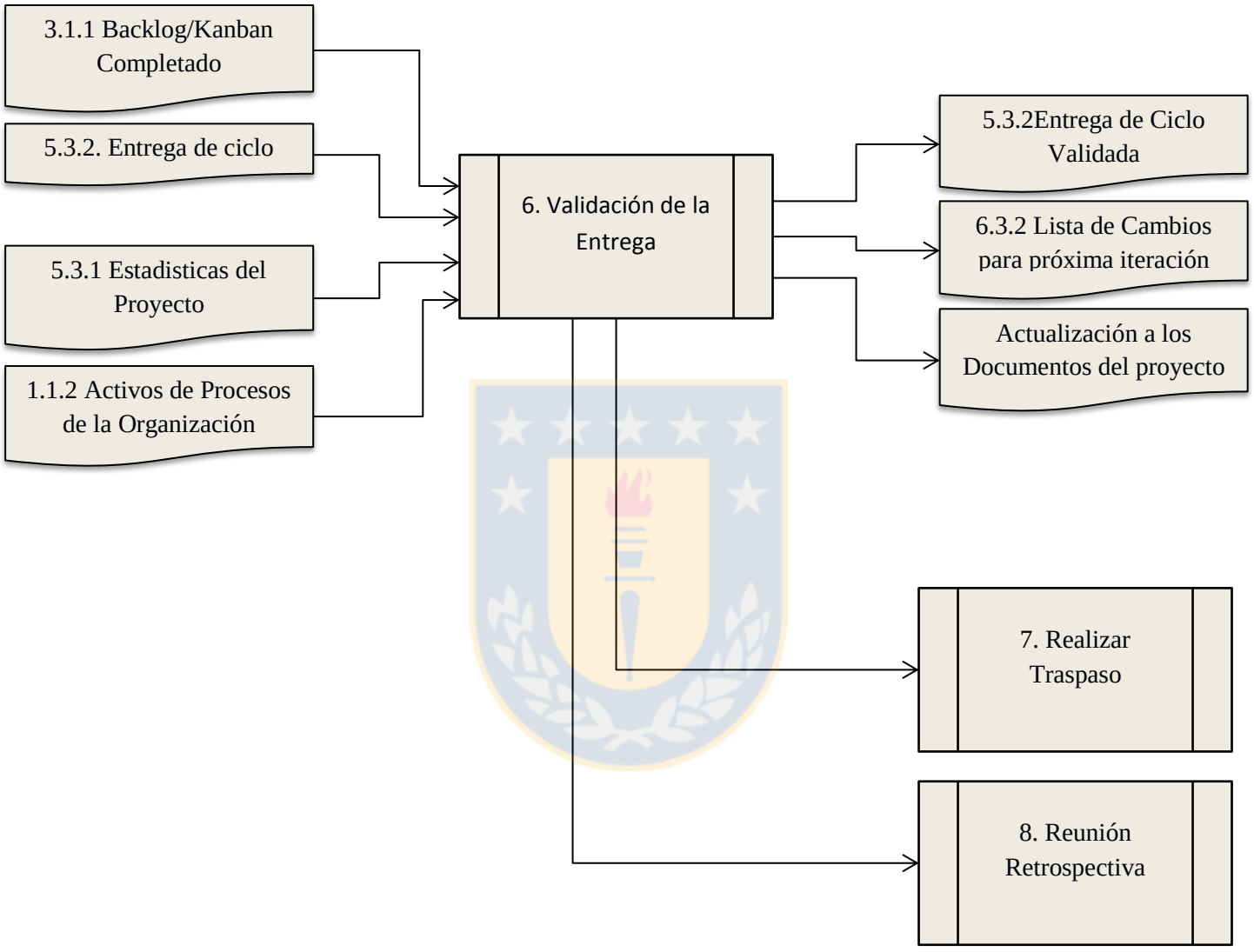
Proceso 4: Reunión Diaria



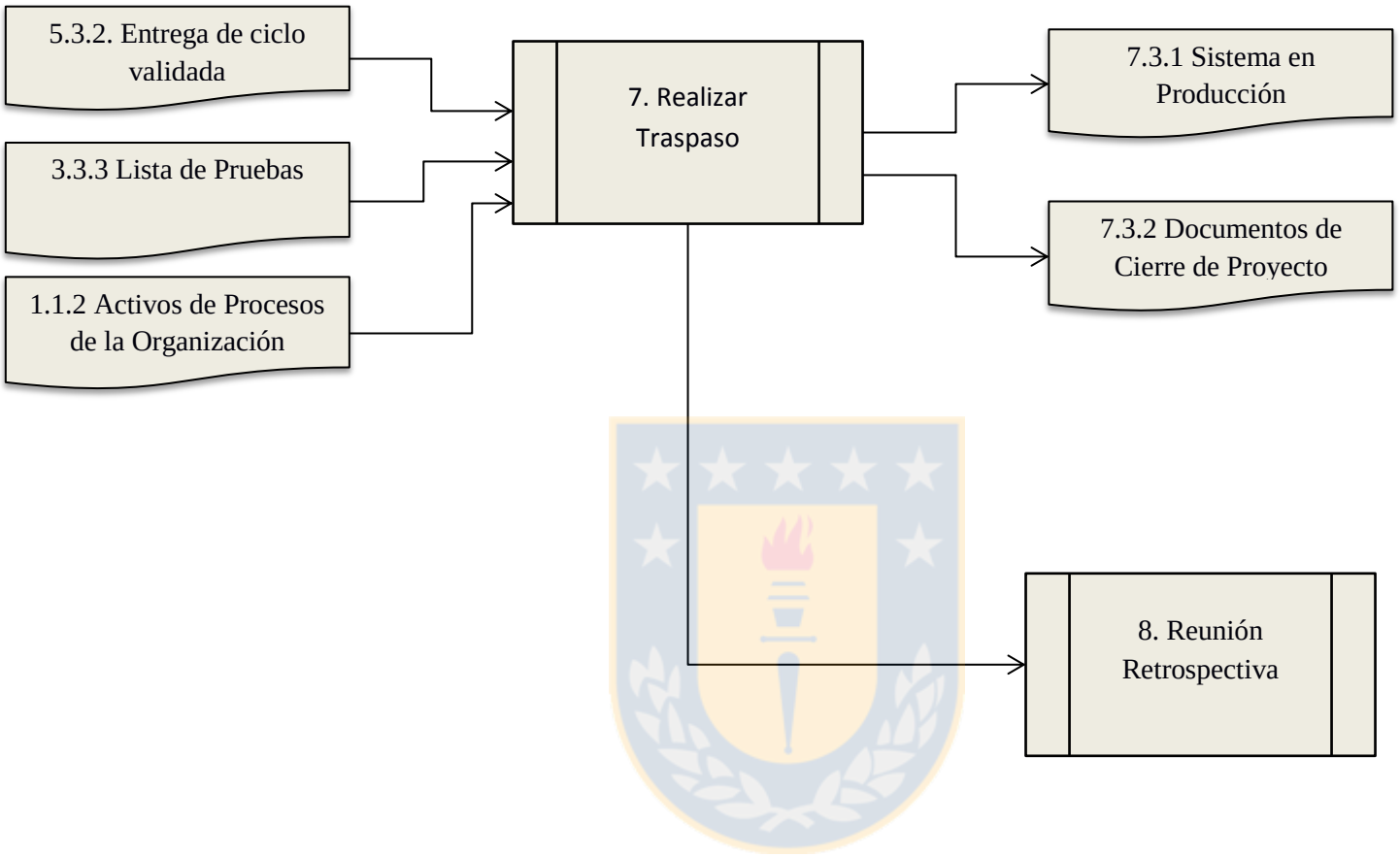
Proceso 5: Ejecutar Tareas



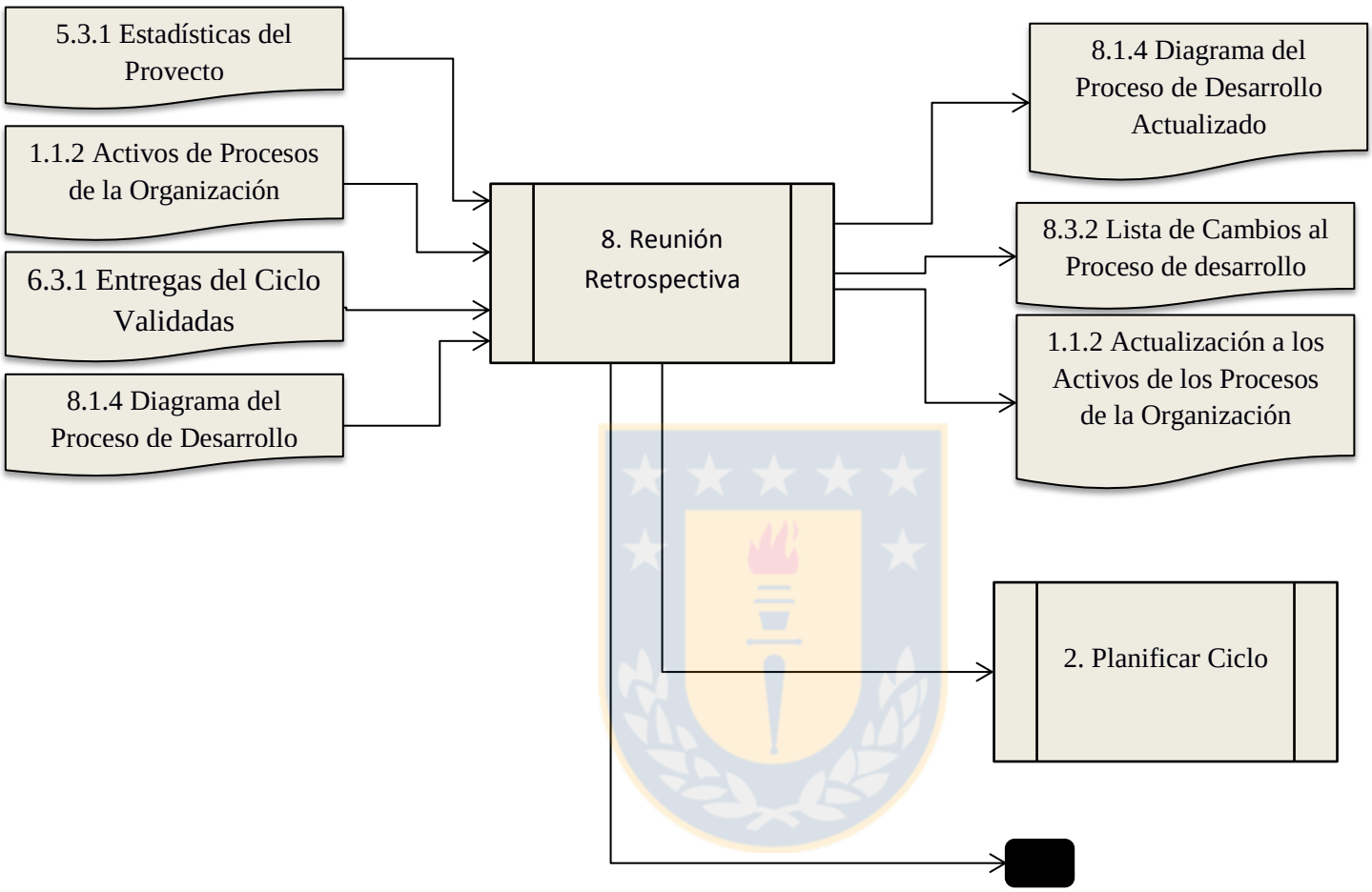
Proceso 6: Validación de la Entrega



Proceso 7: Realizar Traspaso



Proceso 8: Reunión Retrospectiva



Anexo 6: Verificación con Chaos Report

El Chaos Report de 2013, se centra en proyectos pequeños (que también son los que tienen mayor tasa de éxito). Cada apartado tiene un número de puntos que corresponde a su importancia en el éxito del proyecto. A continuación se realizará una comparación entre el método descrito anteriormente y el Chaos Report.

Executive Management Support (20/100)

Es el mayor factor en el éxito de un proyecto. Este ítem se refiere a la presencia de un Sponsor Ejecutivo, una persona que se haga responsable del resultado del proyecto y que esté comprometida con su éxito. Esta persona suele ser, también, la persona que posee el último decir en la aprobación del costo del proyecto.

En Lean esta persona pertenece al grupo de los interesados, los cuales aparecen constantemente en el desarrollo, modificando y ajustando el proyecto para hacer realidad su visión. Chaos Report también aconseja, para proyectos pequeños concentrarse en "hacer" el proyecto por sobre "negociarlo". Lean, con su política de prototipado rápido, comunicación cara a cara y retroalimentación continua mantiene constantemente a los interesados comprometidos con el proyecto mientras reduce el riesgo y el tiempo de decisión del proyecto.

Adicionalmente, al generar Master Developers, se genera un interesado más que también se hace responsable del proyecto, sobretodo del lado técnico del proyecto. Esto permite realizar negociaciones e intercambios de información de manera más rápida y exacta.

User Involvement (15/100)

El usuario es la persona que tiene la última palabra en el proyecto. Un sistema que cumple a la perfección los objetivos del dueño del proceso y clientes no tiene ningún valor si el usuario es incapaz de utilizarlo. En consecuencia es necesario incluir tempranamente al usuario para recibir la retroalimentación correspondiente.

El método entregado asume que el usuario estará en la lista de los interesados. También recomienda tener un acceso rápido al usuario para poder verificar constantemente temas de interfaz y/o accesibilidad al sistema.

Optimization (15/100)

Se refiere a optimizar los recursos del proyecto, entregando un producto rápido, barato y de alta calidad. Es importante tener en cuenta que la gran mayoría de las features de un sistema se usan muy poco o nunca (20% son usadas frecuentemente, 50% muy poco o nunca y 30% a veces). Esto

indica que existe una gran cantidad de esfuerzo realizado en un proyecto que no entrega ningún valor a los interesados. Por lo tanto, un proyecto optimizado, buscará cuales sean esas features cruciales y muy usadas se enfocará en ellas para realizar un sistema sólido con entrega rápida.

Lean, por otra parte, se centra enormemente en esta observación, utilizando iteraciones, retroalimentación continua de los interesados y reglas de desarrollo, para evitar realizar trabajo innecesario. El concepto de "Reducir la basura" apoya esta práctica.

Skilled Resources (13/100)

Se refiere a tener a la gente correcta realizando las labores correctas en el momento correcto. También a que la gente posea las habilidades adecuadas para realizar las labores necesarias a tiempo.

Lean, por otra parte, es parte de las metodologías ágiles que se centran en el lado humano del desarrollo de software, permitiendo que el equipo de trabajo se autodetermine, se automotive y se automejore, entregándoles las herramientas para realizar su labor de la forma que consideren más adecuada. A cambio, se exige mayor compromiso con el proyecto.

Con respecto a la motivación, es bueno agregar que los proyectos pequeños con hitos (o entregas) frecuentes motivan a los equipos de trabajo porque entregan una rápida y frecuente sensación de logro, mientras que los equipos pequeños entregan una sensación de cooperación y compañerismo.

Project Management Expertise (12/100)

El jefe de proyecto debe tener experiencia manejando proyectos, con buen juicio y capacidad de negociación y de toma de decisiones. Esta capacidad se puede lograr sólo por medio de la educación y experiencia. Es aconsejable, también, que conozca a profundidad el proceso de desarrollo y que lo salvaguarde. También debe tener tacto, diplomacia y conocimientos formales de Administración de Proyectos para poder tener una buena relación con los interesados y el sponsor ejecutivo.

En Lean, parte de las labores de supervisión se vuelven visibles por medio de las continuas retroalimentaciones y los artefactos. La presencia de los Master Developers también reduce la dependencia de los Jefes de Proyecto. Esto no quiere decir que los Jefes de Proyecto sean innecesarios, sino que su rol se concentra en guiar y proteger al equipo y el proceso, así como reducir las interferencias externas. La experiencia es crucial en el jefe de proyecto, mientras que la educación debe ser la correcta; certificaciones para métodos pesados y lentos (CMM) juegan en contra de una implementación ágil

Agile Process (10/100)

Lean es, por defecto, un método ágil.

Clear Business Objetives (6/100)

Para proyectos pequeños, la alineación con la estrategia de negocios de la empresa no es tan crucial porque los proyectos pequeños tienen menos costo y, por lo tanto, se pueden dar el lujo de ser más experimentales. Sin embargo, la visión del proyecto debe ser, de todas formas, clara. Con el objetivo de aumentar la velocidad de diseminación de información se recomienda priorizar métodos informales tales como: reuniones diarias de pie, tablas de tareas y diagramas de burndown. El ideal es mantener aquellos artefactos de más alto valor de forma ligera, promover la innovación individual y mantener un bajo nivel de carga para la administración del proyecto.

Lean se concentra naturalmente en lograr estos objetivos promoviendo varios artefactos ligeros y comunicación informal y cara a cara, así como alta velocidad y colaboración

Emotional Maturity (5/100)

La madurez emocional se refiere a las habilidades para auto-organizarse y auto-motivarse, así como cooperar y motivar a otras personas. Estas habilidades son mucho más importantes en grupos pequeños, pero, a la vez, son más difíciles de medir al inicio del proyecto. Se recomienda revisar el historial de los miembros del equipo de desarrollo y los interesados y monitorear la existencia de comportamientos peligrosos como el sobrecomprometerse en las entregas, la arrogancia, la indiferencia o falta de interés en el proyecto y el falsificar datos o mentir sobre el estado del proyecto.

La metodología propuesta posee mecanismos para fortalecer al equipo, sin embargo, se asume que un equipo auto-motivado y con ansias de lograr importantes metas es capaz de superar los obstáculos de madurez emocional por su cuenta.

Execution (3/100)

Es cumplir el proyecto basado en un plan. Ya que el costo del proyecto es menor, es posible experimentar con diferentes enfoques, modificando las herramientas o las metodologías. Sin embargo, de todas formas, es necesario controlar y enfocar al equipo, midiendo las estadísticas de desarrollo, probando las entregas y reajustando la dirección del proyecto en caso de ser necesario.

Lean (y más explícitamente, kanban) se basan en revisar y mejorar constantemente el proceso de desarrollo, estableciendo y deshaciendo reglas conforme sea necesario y adaptándose continuamente al cambio. Herramientas como las reuniones retrospectivas, decisiones tardías y estándares de programación que otorguen flexibilidad son algunas de las prácticas que permiten mejorar la ejecución del proyecto

Tools and Infraestructure (1/100)

Son el ítem menos importante pues apoyarse demasiado en las herramientas conlleva dependencia y aumento de costos. El uso de herramientas debe ajustarse al proyecto en cuestión. Si bien es útil, por un lado, disponer de varias opciones para llevar a cabo distintos tipos de proyecto, por otro, es un costo mantener varias opciones con superposición cuando existe la posibilidad de estandarizar. En caso de los proyectos pequeños existe la posibilidad de probar nuevas herramientas con un menor riesgo, expandiendo las capacidades del equipo de desarrollo.

Al igual que en el punto anterior, Lean recomienda probar nuevas herramientas continuamente, siempre y cuando el dominio de las nuevas herramientas traiga consigo un menor tiempo o una mayor calidad de desarrollo

Anexo 7: Equivalencia entre PMI y Lean

Fuente: Elaboración del autor tras lectura del PMBOK (28), la Extensión de Software del PMBOK (29) y el libro de Poppendieck (4)

4. Gestión de la Integración del Proyecto		
Proceso en PMBOK	Equivalente Lean	Equivalente según proceso propuesto
4.1 Desarrollar el Acta de Constitución del Proyecto	No tiene, se entregan herramientas para ver viabilidad del proyecto aunque es implícito con el contrato. Se pone énfasis en construirlo con la ayuda de los Master Developer.	Generar Contrato. Iteración inicial
4.2 Desarrollar el Plan para la Dirección del Proyecto	Implícito al utilizar Lean/Discutir con los Master Developer/Variable durante la ejecución del Proyecto/Comunicación constante	Implícito
4.3 Dirigir y Gestionar el Trabajo del Proyecto	Implícito en Lean.	Implícito
4.4 Monitorear y Controlar el Trabajo del Proyecto	Monitoreo del Proyecto en iteraciones, Reuniones de avance, estadísticas.	Testing y Validación
4.5 Realizar el Control Integrado de Cambios	Daily Build, Testing continuo, Reuniones con cliente	Planificación del Ciclo y Reunión interna
4.6 Cerrar Proyecto o Fase	Reunión de Avance (Final), Contrato	Traspaso y cierre
5. Gestión del Alcance del Proyecto		
5.1 Planificar la Gestión del Alcance	Contrato/Implícito en Lean (Reuniones/iteraciones)	Implícito
5.2 Recopilar Requisitos	Reuniones/Iteraciones	Planificar Ciclo
5.3 Definir el Alcance	Priorización de items del Cliente	Planificar Ciclo
5.4 Crear la EDT/WBS	Contrato/Implícito en Lean (Reuniones/iteraciones)	Reunión Interna
5.5 Validar el Alcance	Verificación: Testing/Daily build. Validación: Reuniones con el cliente/usuario	Validación de Entrega
5.6 Controlar el Alcance	Reuniones/Iteraciones	Validación de Entrega, Reunión

		Interna
6. Gestión del Tiempo del Proyecto		
6.1 Planificar la Gestión del Cronograma	Contrato/Iterativo con backlog/Velocidad/Pulling	Implícito
6.2 Definir las Actividades	Implícito/Minimum marketable features/Descomposición de items	Reunión Interna
6.3 Secuenciar las Actividades	Definición en Reuniones con énfasis en decidir tarde y generar un diseño flexible/Pulling	Reunión Interna. Reunión Diaria
6.4 Estimar los Recursos de las Actividades	Reunión Diaria	Reunión Interna. Reunión Diaria
6.5 Estimar la Duración de las Actividades	Velocidad e iteraciones	Reunión Interna.
6.6 Desarrollar el Cronograma	Priorización de Backlog e iteraciones	Reunión Interna
6.7 Controlar el Cronograma	Priorización de Backlog e iteraciones/Velocidad/Burndown	Reunión Diaria
7. Gestión de los Costos del Proyecto		
7.1 Planificar la Gestión de los Costos	Contrato + Iteraciones	N/A
7.2 Estimar los Costos	Contrato + Iteraciones + Análisis de Costo + Master Developers	N/A
7.3 Determinar el Presupuesto	Contrato	N/A
7.4 Controlar los Costos	Reuniones y Negociación	N/A
8. Gestión de la Calidad del Proyecto		
8.1 Planificar la Gestión de la Calidad	Reuniones y planes para establecer estándares de código, arquitectura para daily builds, herramientas de testing, revisiones de código y negociaciones para determinar factores de éxito y riesgo en el proyecto (IE. Necesidades de seguridad), client testing	Implícito
8.2 Realizar el Aseguramiento de la Calidad	Mejora Constante	Prácticas de programación de Lean

8.3 Controlar la Calidad	Practica del 8.1	Prácticas de programación de Lean
9. Gestión de los Recursos Humanos del Proyecto		
9.1 Planificar la Gestión de los Recursos Humanos	Rol de facilitador. Master Developers. Auto-organizada. Implícito en Lean.	Implícito
9.2 Adquirir el Equipo del Proyecto	N/A	N/A
9.3 Desarrollar el Equipo del Proyecto	Auto-organizada, Test Driven Programming, co-localización,	Reunión Retrospectiva
9.4 Dirigir el Equipo del Proyecto	Performance del equipo, Feedback	Reunión Diaria
10. Gestión de las Comunicaciones del Proyecto		
10.1 Planificar la Gestión de las Comunicaciones	Reuniones, Gran énfasis en comunicación verbal, evitar utilizar documentos como método de comunicación	Implícito
10.2 Gestionar las Comunicaciones	Reuniones + Radiadores de Información	Negociación con interesados, lista de artefactos (Backlog/Kanban/Radiador)
10.3 Controlar las Comunicaciones	Reuniones + Backlog priorizados/Velocidad actualizada	Backlog/Kanban/Radiador/Reuniones
11. Gestión de los Riesgos del Proyecto		
11.1 Planificar la Gestión de los Riesgos	Implícito con Lean (Priorización de backlog por riesgos), reducción de riesgo con decisiones tardías e informadas, arquitectura flexible)	Implícito
11.2 Identificar los Riesgos	Reunión Diaria y con Cliente	Reunión interna, practicas (prototipos, decisión tardía, entrega rápida, retroalimentación)
11.3 Realizar el Análisis Cualitativo de los	Velocidad/Master Developers/5 Why	Reunión Diaria, Practicas

Riesgos		
11.4 Realizar el Análisis Cuantitativo de los Riesgos	Priorización de Backlog, Reuniones con cliente	N/A
11.5 Planificar la Respuesta de los Riesgos	Discusión con cliente	Reunión Diaria, practicas, reuniones con cliente
11.6 Controlar los Riesgos	11.1 En la práctica + Testing	Practicas
12. Gestión de las Adquisiciones del Proyecto		
12.1 Planificar la Gestión de las Adquisiciones	Lean no tiene mucha información al respecto excepto la sección sobre contratos	N/A
12.2 Efectuar las Adquisiciones		N/A
12.3 Controlar las Adquisiciones		N/A
12.4 Cerrar las Adquisiciones		N/A
13. Gestión de los Interesados del Proyecto		
13.1 Identificar a los Interesados	Cliente usuario involucrados en las iteraciones/Master Developers	Reuniones
13.2 Planificar la Gestión de los Interesados	Implícito en Lean	Implícito y/o negociado
13.3 Gestionar la Participación de los Interesados	Reuniones	Reuniones
13.4 Controlar la Participaciones de los Interesados	Reuniones	Reuniones