UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

Interpretación de ISO/IEC 29110 Perfil Básico con principios y prácticas de Lean Software Development

TESIS

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

PRESENTA:

PEREZ HERNANDEZ ALEJANDRO PARMENO

DRA. HANNA J. OKTABA

FACULTAD DE CIENCIAS

PROGRAMA DE POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MÉXICO, D.F. AGOSTO 2013

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

AGRADECIMIENTOS

A mis padres por su apoyo, cariño y comprensión.

Contenido

Capítulo 1. Introducción. 4

1. Motivación 4
2. Objetivo. 5
3. Metodología. 5
4. Metas. 6

Capítulo 2. Marco teórico. 7

1 ¿Por qué Lean? 7

2 Lean Software Development. 9

2.1 Desperdicios. 10

2.1.1 En la manufactura 11

2.1.2 En la industria de Software 12

2.2 Principios. 14

2.2.1 Pri1. Eliminar los desperdicios 14

2.2.2 Pri2.Calidad Integrada 15

2.2.3 Pri3.Crear Conocimiento 17

2.2.4 Pri4.Diferir el compromiso 18

2.2.5 Pri5.Reaccionar tan rápido como sea posible 18

2.2.6 Pri6.Respetar a las personas 19

2.2.7 Pri7.Optimizar el todo 20

2.3 Prácticas. 21

2.3.1 Pra1. Value Stream Mapping (mapa de flujo de valor o mapa de cadena de valor) 21

2.3.2 Pra2. Retroalimentación. 29

2.3.3 Pra3. Iteraciones 29

2.3.4 Pra4. Sincronización. 30

2.3.5 Pra5. Desarrollo basado en conjuntos 30

2.3.6 Pra6. Pensando en opciones 30

2.3.7 Pra7. Responder hasta el último momento 30

2.3.8 Pra8. Tomar decisiones 31

2.3.9 Pra9. Sistemas Pull 31

2.3.10 Pra10. Teoría de filas. 32

2.3.11 Pra11. El costo de atrasar 32

2.3.12 Pra12. Autodeterminación (Work-Out) 32
2.3.13 Pra13. Motivación 33
2.3.14 Pra14. Liderazgo 33
2.3.15 Pra15. Experiencia 34
2.3.16 Pra16. Refactorización 34
2.3.17 Pra17. Pruebas. 34
2.3.18 Pra18. Métricas 34
2.3.19 Pra19. Contratos 35
2.4 Lean y el manifiesto ágil 35
3 ISO/IEC 29110 Perfil Básico. 36
4 KUALI-BEH. 38
Capítulo 3. Procesos 40
1 Proceso de Administración del Proyecto (AP) 40
1.1 Listas de Productos del proceso de administración 41
1.1.1 AP productos de entrada 41
1.1.2 AP productos de salida 41
1.1.3 AP productos internos 41
1.2 Lista de tareas del proceso de administración con recomendaciones LSD 42
2 Proceso de Implementación de Software (IS) 48
2.1 Listas de productos del proceso de implementación 48
2.1.1 IS productos de entrada 48
2.1.2 IS productos de salida 48
2.1.3 IS productos internos 49
2.2 Lista de tareas del proceso de implementación con las recomendaciones LDS 49
2.3 Roles 64
3 Descripción de Productos 65
Capítulo 4. Validación 79
Capítulo 5. Conclusiones 82
Anexo 1. 85
Anexo 2. 85
Bibliografía 87

Capítulo 1. Introducción.

1. Motivación

La industria de Software ha permeando muchos aspectos de la vida cotidiana así también en la industria, desde el entretenimiento hasta los sistemas de producción y distribución de materias primas. Todos usamos algún producto de software, ya sea especializado o de uso más general, que junto al incremento en capacidad y cobertura de las redes de datos permite que siga con paso firme el crecimiento en la demanda de productos de software. La industria debe responder ante estos retos, las organizaciones tendrán que demostrar su valía en una competencia feroz. Los mercados actuales tienen gran demanda pero también mucha oferta. Las organizaciones deben buscar establecer sus ventajas competitivas y una cultura que las beneficie para así satisfacer a sus clientes.

Una de las formas en que las compañías de desarrollo de software pueden demostrar la calidad de sus productos es por medio de certificaciones. Una certificación dice que la organización fue evaluada y que ciertos procesos dentro de ella ya pasaron una revisión y evaluación, asegurándonos que es capaz de realizar esa parte de su operación según un estándar reconocido.

Dentro de este marco, se encuentran las certificaciones ISO/IEC, encargadas de promover normas internacionales de fabricación (tanto de productos como de servicios), comercio y comunicación para todas las ramas industriales. Sin quedar exenta la industria de software, existen varias normas las cuales pueden ser aplicadas. Dentro de estas normas se encuentra la norma ISO/IEC 29110-5-1-2:2011, la cual está enfocada a pequeñas organizaciones, que busca evaluar sus procesos de desarrollo y administración para la creación de productos de software.

Actualmente hay una tendencia de la industria que apunta a los desarrollos de software agiles siendo estos más populares en las pequeñas organizaciones de software dada su sencillez y adaptabilidad. Se propone una fusión que apoyará a desarrollar una cultura dentro de la organización así como promover el valor de la misma mediante una certificación con una Norma ISO/IEC. Así nace la idea de adaptar principios y prácticas de

una metodología ágil siendo usados para este trabajo principios y prácticas de Lean Software Development (LSD). Aplicados dentro de los procesos descritos en la norma ISO/IEC 29110 Perfil Básico.

LSD fue elegida como la metodología ágil a usar en este trabajo dadas las historias de éxito Lean encontradas en otras ramas en la industria, por ejemplo la manufactura y la administración.

2. Objetivo.

Desarrollar una propuesta de interpretación de los principios y prácticas de Lean Software Development que se puedan aplicar a los dos procesos de ISO/IEC 29110 Perfil Básico (29110-5-1-2:2011).

3. Metodología.

Se propone una metodología de investigación teórica para realizar un análisis Histórico - Lógico que nos adentrara en los fundamentos del pensamiento *Lean* y nos permitirá hacer un acercamiento sencillo y coherente a las técnicas, practicas y metodologías que construyen el tema de investigación.

Así dividimos la investigación en 2 partes para realizar el análisis. Se hará uso del método histórico que estudia la trayectoria real de los fenómenos y acontecimientos en el transcurrir de su historia. Y del método lógico que investiga las leyes generales de funcionamiento y desarrollo de los fenómenos.

Estos métodos en conjunto reflejan el objeto en sus conexiones más esenciales, ofrecen la posibilidad de comprender su historia que junto con los métodos lógicos que expresan, en forma teórica, la esencia del objeto, la necesidad y la regularidad de este, buscando reproducir el objeto en su forma superior madura.

Estos métodos permiten unir el estudio de la estructura del objeto de investigación y la concepción de su historia. (Hernández Meléndrez, 2006)

En este trabajo se aplicará metodología antes descrita para comprender las practicas *Lean* desde sus inicios hasta llegar a su aplicación para el desarrollo de software. Con esto se

podrán analizar y vincular los elementos del ISO/IEC 29110 Perfil Básico y los principios y las practicas de *Lean*.

4. Metas.

- Crear una interpretación de ISO/IEC 29110 Perfil Básico usando Lean Software Development.
- Obtener una recopilación de prácticas y principios como resultado de la revisión bibliográfica histórica.
- Clasificar y analizar los principios y prácticas identificados aplicables a los procesos de ISO/IEC 29110 Perfil Básico.
- Desarrollar los elementos que constituyan una interpretación de ISO/IEC 29110 Perfil Básico con principios y prácticas de LSD.
- Obtener una validación de la interpretación por un experto en prácticas Lean.

A continuación en los siguientes capítulos de hace una descripción del marco teórico usado en este trabajo de tesis. El marco teórico incluye como temas principales Lean Software Development (LSD), el ISO/IEC 29110 Perfil Básico y KUALI-BEH. De LSD hace una descripción de la filosofía e historia agregando una recopilación de principios y practicas. Del ISO/IEC 29110 Perfil Básico se escribe un breve resumen el cual busca introducir los conceptos y dar contexto al siguiente capitulo donde se usaran las definiciones de tareas y productos propias del ISO/IEC 29110 a las que se agregaron recomendaciones basadas en LSD. De KUALI-BEH se agrega un breve resumen ya que de ese trabajo fue usada la plantilla de definición de prácticas (Anexo 1.)

Posteriormente se incluye en este trabajo la validación de lo entes expuesto por un experto en el área de ingeniería de software con amplia experiencia en las metodologías agiles. Finalmente se agregan las conclusiones resultantes de la validación junto a la experiencia recopilada durante la elaboración de este trabajo.

Capítulo 2. Marco teórico.

1 ¿Lean?

El *Lean Software Development* (LSD- Desarrollo de Software Lean) tiene sus inicios en el *Toyota Production System* (TPS- Sistema de Producción de Toyota). Su propósito es ayudar a las organizaciones de software a optimizar sus procesos y sus métodos de producción de manera en que pueda entregar sus productos al mercado de la forma más rápida y con mejor calidad. El movimiento *Lean* puede considerarse como un nuevo método de desarrollo que intenta identificar y erradicar los problemas y desventajas de metodologías tradicionales, como por ejemplo el método de Cascada.

El modelo de Cascada es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso en cuanto a costos, tiempo y calidad.

El proceso de creación del software puede tomar mucho tiempo, ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.

Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. (Pressman, 2005)

Lean hace énfasis en las personas y la comunicación. Propone, que si se respeta a las personas que producen software y estos se pueden comunicar de forma eficiente y es más probable que logren entregar un buen producto y satisfacer al consumidor final.

Lean se originó en Japón dentro del área de manufactura y gradualmente se expandió a otras áreas tales como la gestión, el manejo de oficinas, y finalmente al desarrollo de software.

El TPS surgió con base en ideas e innovaciones de Kiichiro Toyoda, Taiichi Ohno y otros responsables de Toyota en los años 30, que implementaron una serie de innovaciones en sus líneas de producción de tal modo que facilitaran tanto la continuidad en el flujo de material como la flexibilidad en el momento de fabricar distintos productos. Esto se hizo aún más necesario a finales de la 2ª Guerra Mundial, cuando surgió la necesidad de fabricar pequeños lotes de una gran variedad de productos.

El TPS se fundamenta en la optimización de los procesos productivos mediante la identificación y eliminación de desperdicios (MUDA en japonés, o WASTE en inglés) y el análisis de la cadena de valor, para finalmente conseguir un flujo de material estable y constante, en la cantidad adecuada, con la calidad asegurada y en el momento en que sea necesario. Es decir, tener la flexibilidad y fiabilidad necesarias para fabricar en cada momento lo que pide el cliente. Ni más, ni menos.

Toyota ha observado que adaptando los equipos de fabricación a sus capacidades reales de producción y con la introducción de sistemas de calidad integrados en los procesos (poka-yokes) mejoró la calidad de los productos.

Se notó que al tener equipos de producción, que seguían la secuencia de fabricación completa, lograban fabricar muchos lotes pequeños de distintas piezas mejorando la capacidad de producción y su flexibilidad.

También se notó que haciendo que cada máquina avisara a la máquina anterior cuando necesitaba material (sistema pull) harían posible el fabricar con bajos costos, una amplia variedad de productos manteniendo una alta calidad consiguiendo que los tiempos de proceso (lead times) fueran muy rápidos.

También observaron que con la introducción de un sistema de gestión de la información facilitaban la producción y hacían que esta fuera más precisa.

Con la suma de las adecuaciones y cambios supieron que podrían responder de manera efectiva y eficaz a las variaciones en las demandas de los clientes.

Los principios del TPS trasladados al LSD permiten a los equipos de trabajo enfocar sus esfuerzos al desarrollo de software y provocan un cambio en el ambiente de trabajo.

Este cambio busca crear un lugar donde haya estabilidad y control (en lo se puede controlar) en el mejor interés y habilidades del equipo para así poder crear software que pueda satisfacer en la mayor medida posible al cliente.

Por ejemplo el análisis de la cadena de valor, aplicada al desarrollo de software, busca eliminar tareas y artefactos que no contribuyen a la satisfacción del cliente y que puede ser visto como un desperdicio bajo los principios propuestos por el LSD.

Las ideas *Lean* fueron trasladadas al desarrollo de software por Mary y Tom Poppendieck y publicadas en el libro *Lean Software Development: An Agile Toolkit* (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003).

Estas ideas fueron extendidas y desglosadas para su aplicación en un ambiente de desarrollo de software como una guía de aplicación en la publicación de los mismos autores *Implementing Lean Software Development - from Concept to Cash* (Poppendieck & Poppendieck, Implementing Lean Software Development - from Concept to Cash, 2006).

2 Lean Software Development.

Lean Software Development muestra a los profesionales del desarrollo de software como alcanzar avances en la calidad, costos, velocidad de desarrollo y valor del negocio al adaptar los 7 principios Lean, que han provocado cambios revolucionarios en las áreas de manufactura y desarrollo e investigación(R&D).

Mary y Tom Poppendieck tomaron las practicas industriales Lean y nos muestran cómo aplicarlas directamente al desarrollo de software. Sabemos que es muy diferente leer sobre estos principios y prácticas para la manufactura en una planta industrial que ver como estas ideas pueden ser aplicadas en procesos de desarrollo de software. La

experiencia de Mary en la manufactura y el desarrollo industrial de productos le proporciona un punto de vista privilegiado en el momento de pensar cómo aplicar estas prácticas y principios que junto a Tom, que posee experiencia en tecnologías de la información, les brindó una magnífica posición para saber cómo aplicar los principios y practicas al desarrollo de software. El LSD de clasifica por sus principios y prácticas como una metodología ágil.

El desarrollo de software ágil tiene sus principios hace ya casi 20 años (y como movimiento desde el 2003) aun que en la manufactura y desarrollo de productos industriales muchos de estos principios (Lean y agiles) se remontan a hasta la década de los 30s lo cual le proporciona credibilidad adicional a los principios y prácticas del desarrollo de software ágil, pero más importante les provee una gran cantidad de ideas y experiencia que pueden fortalecer a las practicas agiles.

Existen un conjunto de herramientas LSD propuestas para administradores de proyectos, líderes de equipo y administradores de tecnología que quieren agregar valor a sus proyectos en vez de convertirse en obstáculos para sus equipos de trabajo y en general para sus proyectos.

El LSD es una metodología que busca la eliminación de los 7 desperdicios de los productos manufacturados (sobreproducción, tiempo de espera, transportes innecesarios, exceso de procesado, inventarios, movimientos innecesarios y productos defectuosos o re trabajos).

2.1 Desperdicios.

Desperdicio es considerado todo aquello que no agrega valor al producto, ya sea un producto manufacturado (TPS) o un producto de software (LSD). Se considera valor de un producto a una cualidad que es descrita en la medida que el producto satisface las necesidades del cliente.

Aquí las definiciones de los 7 desperdicios para la manufactura según Lean y los 7 desperdicios en la industria de software según Lean Software Development.

En la manufactura	En la Industria de software
Sobreproducción	*Trabajo parcialmente hecho*
Tiempo de Espera	*Reaprendizaje*
Transportes innecesarios	*Transferencias*
Sobre procesamiento o procesamiento incorrecto	*Características Extra*
Inventarios	*Atrasos*
Movimientos innecesarios	*Cambio de tareas*
Productos defectuosos o re trabajos	Defectos

2.1.1 En la manufactura

En la manufactura (viéndolo desde el Toyota Production System) los desperdicios se reflejan negativamente y de manera directa en la calidad o costo del producto. Bien del lado de los clientes es visto como un precio alto o un mal producto que dadas sus cualidades. La relación costo-beneficio es negativa. De esta manera todo aquello que resta valor desde el punto de vista del cliente es un desperdicio. El desperdicio puede ser generado de manera directa sobre el producto o de manera indirecta, estando así latente en cualquier fase de la producción.

Desperdicio	Definición
Sobreproducción	Si existe sobreproducción es necesario incrementar el inventario de materiales y sus respectivos costos.
Tiempo de Espera	Los operadores esperan observando a las máquinas o esperan la entrega de recursos como herramientas, materiales o partes. Es aceptable que en ocasiones la máquina espere al trabajador pero no a la inversa.
Transportes innecesarios	Todos los recorridos innecesarios durante el proceso de producción. Tiempo desperdiciado dado que genera costos y también genera riesgos.
Sobre procesamiento o procesamiento incorrecto	Son responsables de los incrementos en los costos de producción.
Inventarios	El exceso de inventario tanto de materia prima, de productos en proceso y de producto terminado causa largos tiempos

	de entrega, alto riesgo de obsolescencia de los productos, deterioro de los artículos, elevados costos de transporte, almacenamiento y retrasos. El inventario permite que se oculten problemas como producción desnivelada, entregas a destiempo por parte de los proveedores, defectos, tiempos ociosos de los equipos y largos tiempos de preparación. Aunado a lo anterior debemos contemplar que para manejar el inventario se requiere personal para cuidarlo, controlarlo. Todo esto no agrega valor al producto y solo genera costos y riesgos innecesarios.
Movimientos innecesarios	Cualquiera que sea el movimiento efectuado por el personal durante sus actividades como observar, buscar, acumular partes, herramientas siempre que no tenga nada que ver con la actividad productiva.
Productos defectuosos o re trabajos	La producción de partes defectuosas, las reparaciones o re-procesos, los remplazos en la producción e inspección demandan dedicación de tiempo y esfuerzo que se pueden utilizar para realizar labores que agregan valor al producto.

2.1.2 En la industria de Software

En la industria de software (desde el punto de vista de Lean Software Development) los desperdicios son también impactos negativos en el valor del producto de software desde el punto de vista del cliente, indicadores muy claros son el costo del proyecto y las cualidades del producto de software que dejan desbalanceada la relación costo-beneficio. El desperdicio puede ser generado de manera directa o de manera indirecta sobre el producto de software, estando así latente en cualquier fase del proyecto.

Desperdicio	**Definición**
Trabajo parcialmente hecho	El trabajo parcialmente hecho es un desperdicio porque las ideas no concretadas o el código no probado no ayudan al desarrollo y no agregan valor al producto. Por consiguiente se convierten en desperdicios.
Reaprendizaje	Papeleo, aprobaciones innecesarias son desperdicio, procurar que todo lo que se guarde sea necesario y de valor como aprendizaje para la organización.
Características Extra	Todo código extra trae más trabajo y más riesgos y agrega complejidad y no necesariamente agrega valor al producto.
Transferencias	Cuando se deja una tarea a otro equipo o persona, hay que explicar en lugar en donde uno se encontraba para que esta persona pueda retomar el trabajo, aun así siempre existe una curva de aprendizaje la cual representa un desperdicio.
Atrasos	Esperar a que las cosas pasen. Lo cual solo es un desperdicio de tiempo y no agrega valor al producto.
Cambio de tareas	Cada vez que hay un cambio a una nueva tarea hay un desperdicio de tiempo ya que hay que retomar la nueva tarea y de nuevo al regresar a la tarea anterior.
Defectos	Probar lo más pronto posible para encontrarlos lo más pronto posible y así repararlos lo más pronto posible. Los defectos en si mismos son un desperdicio por que nos indican que algo que no fue hecho correctamente y nos lanzan a una etapa de corrección.

El lema de esta filosofía es "obtener las cosas correctas, en el momento correcto, en el lugar correcto y en la cantidad correcta, minimizando el desperdicio y estando abiertos al cambio".

2.2 Principios.

En esta parte profundizaremos en los 7 Principios de Lean y los relacionamos con mitos comunes conocidos dentro de la industria de Software.

Principio	**Mito**
Eliminar el desperdicio	Especificaciones tempranas reducen el desperdicio
Calidad integrada	El trabajo de las pruebas es encontrar defectos
Crear conocimiento	Las predicciones crean predictibilidad
Diferir el compromiso	Planear es un compromiso
Entregar rápido	Apresurarse crea desperdicio
Respetar a las personas	Solo hay una mejor forma
Optimizar el todo	Optimizar por descomposición

2.2.1 Pri1. Eliminar los desperdicios

Todo lo que no añade valor al cliente se considera un desperdicio.

Con el fin de poder eliminar los desperdicios primero se debe reconocer el desperdicio y encontrar su causa raíz. Toda actividad que pueda ser excluida dado que aun sin ella podemos llegar al resultado, es desperdicio. Todo proceso o funcionalidad extra que no sea usada por el cliente es desperdicio. Son desperdicios también las esperas causadas por actividades externas, por otros equipos o procesos. Los defectos y la baja calidad son de los peores desperdicios. También así los gastos de gestión que no producen o agregan valor real al producto final son considerados como desperdicio.

Mito 1. Especificaciones tempranas reducen el desperdicio

El sentido común dice que hay que pensar antes de actuar, tratar de planear todo, y por eso esperamos que el cliente desde un inicio nos proporcione una lista de todos los requerimientos que él quiere en su producto, pero es muy común que después se necesite plantear un cambio a los requerimientos iniciales. Ya solicitado el cambio se necesita que estos cambios pasen por una serie de personas o departamentos para que estos sean aprobados y puedan ser considerados en el desarrollo y el problema no solo radica en el

flujo de la información y el tiempo que este tome; sino realmente en los costos agregados que producen los cambios,

Por lo que en general y en consecuencia de lo anterior el cliente genera una lista de requerimientos muy extensa pensando en que si pone todo lo que se le ocurre después no habrá necesidad de hacer cambios a los requerimientos iniciales. Con esta práctica se sofoca al proyecto dado que los tiempos y los costos son limitados y se incrementa el riesgo de que ideas no concretadas sean llevadas a los desarrolladores que finalmente tendrán la tarea de encontrar sentido a todo lo requerido por el cliente.

2.2.2 Pri2.Calidad Integrada

La calidad en la filosofía Lean es una cualidad esperada, inherente al sistema y que mejora mientras más ágil sea el proceso de desarrollo. La calidad de los sistemas es pensada y tomada en cuenta desde un inicio. Crece según crece nuestro sistema y que debe mantenerse en mente durante todo el desarrollo.

Hay muchas formas de mantener la calidad de un producto, una de ellas es el uso de estándares, pero los estándares no deben aplicarse solo en el papel, en un ambiente de desarrollo Lean un estándar es la mejor forma que tiene la organización de realizar alguna tarea.

Como parte del ambiente de desarrollo Lean, se anima a los participantes para proponer nuevas y mejores formas de hacer las cosas, probar y si los resultados demuestran alguna ventaja sobre el estándar anterior, esta forma nueva se convertirá en el estándar en uso en la organización.

La calidad también se refiere a la noción de satisfacción del cliente con el uso del producto, a su experiencia como usuario, además de que el producto cumpla con lo que se requiere. Conceptos importantes dentro de la definición de calidad son los conceptos de integridad perceptual e integridad conceptual.

El concepto de integridad perceptual que se refiere a la experiencia del usuario al usar nuestro producto, que percepción proyecta el producto. ¿Cómo es publicitado, entregado,

implementado y accedido? ¿Qué tan intuitivo es su uso? ¿Qué tan bien resuelve los problemas? Un gran ejemplo de la integridad perceptual. Google y los productos Apple.

La integridad conceptual, nos habla de componentes del sistema, se debe pensar en que separados deben funcionar bien y así mismo deben funcionar correctamente juntos, integrados, como en un todo, logrando equilibrio entre la flexibilidad, mantenibilidad, eficiencia y capacidad de respuesta. En resumen las partes del sistema deben ser altamente cohesivas y con bajo nivel de acoplamiento.

Esto puede lograrse mediante la comprensión del dominio del problema y buscando una solución general. La información necesaria para la solución es intercambiada entre el cliente y el equipo de desarrollo, debe ser recibida en pequeños lotes, de preferencia con comunicación cara a cara, sin ninguna documentación por escrito. El flujo de información debe ser constante en ambas direcciones, del cliente a los desarrolladores y viceversa, evitando así periodos de desarrollo aislados del cliente, como lo haría el desarrollo en cascada.

Mito 2. El trabajo de las pruebas es encontrar defectos

El trabajo de las pruebas y de las personas responsables por las pruebas es prevenir los defectos, no encontrarlos. En una organización que busca calidad en los productos de software, uno de los proceso más importantes es integrar la calidad en el código desde un inicio en vez de buscarla probarla ya muy adelante en el proceso de desarrollo. Con esto no se dice que las verificaciones y validaciones sean innecesarias, un verificación final o prueba final siempre es una buena idea, es solo que el encontrar un defecto cuándo se realizan las pruebas no debe ser común o la regla, sino debería ser poco común o la excepción.

Si las pruebas generan un ciclo prueba, corrección y prueba, corrección entonces debemos considerar que el proceso de desarrollo es defectuoso. El sentido común nos dice que alrededor del 80 por ciento de los defectos en una planta de producción son causados por las personas que la operan, cuando en realidad son causados por un sistema que permite

que los errores sucedan, por eso, la mayoría de los defectos son en verdad problemas de administración.

2.2.3 Pri3.Crear Conocimiento

En el pasado un recurso esencial era poseer tierras, aquellos que poseían las tierras tenían el control sobre todo. En algún momento esto cambio y lo importante ahora eran las habilidades, por ello los gremios de artesanos y los comerciantes fueron poco a poco siendo más acaudalados que los terratenientes. Después, en la revolución industrial, lo importante fue el capital, lo que provoco que el poder más importante fuera el poder económico. Ahora la constante es el conocimiento, conocimiento técnico, conocimiento administrativo, conocimiento del proceso y conocimiento del mercado. Mucho de todo este conocimiento es expresado en forma de un producto de software. Muchos de los paradigmas actuales siguen basados en las finanzas y formas de pensar industriales, pero aquellas organizaciones que dominaran este siglo serán aquellas que cambien y se enfoquen en el conocimiento.

Debemos a toda costa guardar entonces el conocimiento que adquirimos y dejarlo disponible a todos los que lo necesiten en nuestra organización, por ello es importante documentar, pero solo escribir algo sobre un problema no es necesariamente documentar, dado que la información relevante también se perderá en un mar de palabras; de igual manera se perderá por la falta de documentación. Para documentar hay que condensar el conocimiento para que el conocimiento crítico y necesario sea escrito y de fácil acceso a quien lo necesite.

Mito 3. Las predicciones crean predictibilidad

En una economía de mercado a menudo se usan predicciones para especular sobre los comportamientos del mercado y con esto poder tomar mejores decisiones orientadas hacia el futuro. Esto no funciona si estas predicciones están basadas en hechos muy complejos o el ambiente sobre el que se pretende predecir contiene altos niveles de incertidumbre y este es el caso del desarrollo de software. La industria de desarrollo de software ha demostrado ser altamente impredecible. Por lo cual el hecho de tomar

decisiones basadas en las predicciones que buscan obtener resultados predecibles demuestra ser un enfoque poco acertado, es mejor ser altamente adaptable y rápido en la respuesta a los eventos para que estos sean manejados así como se presentan creando una mejor noción de predicción ya que en un sistema en donde hay retroalimentación es posible una mejor adaptación al entorno.

2.2.4 Pri4.Diferir el compromiso

Otra idea clave del LDS se establece en el diseño. Si un nuevo sistema de frenos es necesario para un coche, por ejemplo, tres equipos pueden diseñar soluciones al mismo problema. Cada equipo aprende sobre el ambiente del problema y cada equipo diseña una posible solución. Cuando una solución se considera poco racional o poco factible, se desecha. Al pasar de un tiempo, los diseños sobrevivientes se comparan y se elige uno, quizá con algunas modificaciones basadas en el aprendizaje de los demás, este es un gran ejemplo de compromiso aplazado hasta el último momento posible. Las decisiones en el software también podrían beneficiarse de esta práctica para minimizar el riesgo provocado por un solo gran diseño realizado por adelantado.

Mito 4. Planear es un compromiso

Para hablar de esta parte es mejor verlo desde el punto de una frase –Quieres hacer reír a Dios, cuéntale tus planes- en el mundo de los negocios se tienen que hacer compromisos lo cual no implica que tenemos que seguir los planes al pie de la letra, ya que el mundo es altamente flexible y esto provoca que los planes tengan que ser ajustados a las situaciones cambiantes a los que estos se enfrentan.

2.2.5 Pri5.Reaccionar tan rápido como sea posible

En esta era de rápida evolución de tecnológica, no es el más grande quien sobrevive, sino el más rápido. Cuanto antes el producto final se entrega sin defectos considerables más pronto se pueden recibir comentarios y se incorporan en la siguiente iteración. Cuanto más cortas sean las iteraciones, mejor es el aprendizaje y la comunicación dentro del equipo. Sin velocidad, las decisiones no pueden ser postergadas. La velocidad asegura el cumplimiento de las necesidades actuales del cliente y no lo que éste requería para ayer.

Esto da la oportunidad a los clientes de tomar su tiempo y pensar en lo que realmente necesitan. Los clientes valoran la entrega rápida de un producto de calidad.

Si hemos eliminado el desperdicio, incrementaremos el porcentaje del tiempo que gastamos en agregar valor al producto en cada ciclo de desarrollo, como consecuencia entregaremos resultados cada vez más rápido. La rapidez es fundamental para que funcione la ecuación Lean.

En Lean todo debe pasar muy rápido, los ciclos son rápidos y la respuesta al cambio debe ser rápida, lo que significa que debemos acostumbrarnos a la velocidad, recordando que todo aquello que nos quita velocidad es un atraso y los atrasos no agregan valor sino al contrario lo quitan. Una de las cosas que podemos tomar en cuenta para saber si estamos haciendo las cosas bien en Lean es el tiempo.

Mito 5. Apresurarse crea desperdicio

En la búsqueda por disciplina, muchas organizaciones desarrollan documentación de procesos a un nivel de extremo detalle, estandarizan los trabajos de la producción, crean documentación para el flujo de trabajo y generan descripciones específicas de trabajo. A menudo hecho por el grupo de personas que entrena a los trabajadores en el proceso y que los monitorea. El objetivo es alcanzar un proceso respetable y estandarizado el cual entre otras cosas hace fácil mover gente entre proyectos. Pero un proceso diseñado para poder intercambiar gente no produce el tipo de gente que es esencial para crear un proceso de trabajo ágil y flexible.

2.2.6 Pri6.Respetar a las personas

Lean es un sistema de administración que promueve que la gente piense en cada nivel de la organización y de manera particular se enfoca a aquellos que están en la primera línea de desarrollo. Si se implementan todos los principios Lean excepto uno (respetar a las personas) se le quitaría a Lean una de las partes más importantes y obtendrías solo la sombra de las ventajas que Lean puede lograr. Pero si solo se implementa un solo principio (respetar a las personas) los demás principios Lean serian descubiertos por las personas en la organización e implementados de manera casi natural.

Unas de las innovaciones en la forma de pensar que es sugerida a los administradores es saber que cuando las cosas salen mal, usualmente podemos atribuírselas al sistema, por ello esto debe ser un problema de administración.

Los administradores que presionan más y más a las personas para que cumplan con las fechas de entrega o para que reduzcan los defectos están ignorando los problemas de su proceso de desarrollo y con esto usualmente provoca que las cosas salgan peor. Por el contrario los líderes deben tener visión y promover cambios fundamentales que se enfoquen en las causas verdaderas que generan los problemas actuales.

Mito 6. Solo hay una mejor forma

La tarea de encontrar e institucionalizar la mejor forma de hacer cualquier trabajo ha sido encargada a los ingenieros industriales en la mayoría de las industrias. Antecesores de la así llamada policías de procesos, los ingenieros industriales a menudo crean estándares sin realmente entender el trabajo y entonces imponen estos estándares sin darse cuenta de que puede haber una mejor forma de hacer el trabajo.

Los procesos deben ser mejorados por el equipo de trabajo, que es el que hace el trabajo. Ellos necesitan el tiempo, la personalidad y la guía para atacar sus problemas, un problema a la vez, iniciando por el problema más grande y así avanzar en la medida de lo posible hasta los problemas más pequeños. Este continuo proceso de mejora debe ser encontrado en toda organización que desarrolle software.

2.2.7 Pri7.Optimizar el todo

Los sistemas de software hoy en día no son simplemente la suma de sus partes, sino también el producto de sus interacciones. Hay que recordar que los defectos en el software tienden a acumularse durante el proceso de desarrollo por la descomposición de las grandes tareas en pequeñas tareas y por la normalización de las diferentes etapas de desarrollo. Las causas reales de los defectos deben ser encontradas y eliminadas. Cuanto más grande sea el sistema, más serán las organizaciones que participan en su desarrollo y más partes son las desarrolladas por diferentes equipos y mayor es la importancia de tener bien definidas las relaciones entre los diferentes proveedores con el fin de producir

una buena interacción entre los componentes del sistema. Optimizar el todo implica optimizar el producto de software a su interior como a su exterior, todo esto puede ser visto como la optimización de la interacción de los componentes.

Mito 7. Optimizar por descomposición

Una creencia errónea es que si cada parte del sistema es optimizada el sistema se encontrara optimizado. Si hay algo que no pueda ser medido por completo, no hay que medirlo del todo, porque una medición a medias nos dará una optimización a medias lo que es igual o peor que no haber optimizado. Por ejemplo usualmente medimos el desempeño de un proyecto basados en su costo, la agenda o el alcance pero estas métricas no toman en cuenta la calidad y la satisfacción del cliente.

2.3 Prácticas.

En esta parte se abordaran algunas prácticas usadas por el LSD. Prácticas que buscan aplicar los principios de una manera coherente y ordenada para poder así llegar a las metas que cumplan con un los preceptos de calidad, tiempo y presupuesto.

Una de las practicas más usadas y aceptadas dentro del LSD es value stream mapping (o mapa de flujo de valor) para la cual fue usada la plantilla de definición de prácticas de KUALI-BEH (Anexo 1.). Al usar la plantilla se agrega un gran valor a la definición de la práctica ya que dada su naturaleza sistemática se consideró que la mejor forma de generar una definición es mediante esa herramienta.

2.3.1 Pra1. Value Stream Mapping (mapa de flujo de valor o mapa de cadena de valor)

Se utiliza una técnica llamada value stream mapping (o mapa de flujo de valor) para distinguir y eliminar los desperdicios. Representación de la práctica a través de KUALI-BEH (KUALI-BEH, 2012).

VSM-PS	**Práctica**
Mapa de flujo de Valor de estado presente o Mapa de cadena de valor de estado Presente(Value Stream Map Present State)	
Objetivo	
Crear una representación grafica de un proceso en donde queden plasmadas todas las actividades de planeación y creación de un producto de software.	
Entrada	**Resultado**

Conocimiento del proceso de los miembros de la compañía.	Mapa de flujo de valor.
Guía	
Actividades	

Definiciones.

Valor. El valor de un producto de software es una cualidad que es descrita en la medida que el producto satisface las necesidades del cliente. Algunas cualidades de un producto de software son:

Correcto
Confiable
Robusto
Eficiente
Amigable
Verificable
Reusable
Portable
Interoperable
Productivo
A Tiempo
Visible
Coheso
Desacoplado
Comprensible
Mantenible

Cadena de valor O flujo de valor. Es un concepto teórico que permite describir el desarrollo de las actividades de una organización. Visualizándolo como una cadena de eslabones, que inicia con el cliente y termina con el cliente. Cada eslabón debe agregar valor al producto de software.

En la representación gráfica de un flujo de valor podemos hacer un análisis práctico de cómo se va agregando valor al producto mediante la realización de las tareas asociadas y de igual forma, ver en donde no se agrega valor, siendo estas partes consideradas desperdicio.

Tiempo total de trabajo. En un proceso o actividad, es la suma de los tiempos usados por todos los involucrados para completar las tareas propias del proceso o actividad.

Tiempo desperdiciado. En un proceso o actividad, es la suma de los tiempos laborales donde los involucrados no realizan tarea alguna.

Ejemplos:

Línea de espera para sacar copias
Tiempo de inicio del sistema o restablecimiento
Tiempos de compilación

Tiempo Transcurrido. En el proceso o actividad el tiempo total transcurrido, la suma del tiempo total de trabajo más el tiempo desperdiciado.

Eficiencia. Indicador en forma de porcentaje que muestra cuanto del tiempo transcurrido ha sido dedicado al trabajar.

Creación del Mapa de flujo de Valor de estado Presente

Pasos para crear un Mapa de flujo de Valor:

1. Identificar un proceso o una actividad que potencialmente sea generadora de desperdicio.
2. Dividimos el proceso o actividad en tareas. Esto puede ser tan granular como se considere correcto.
3. Identificamos las relaciones de dependencia entre las tareas.

4. Registramos el tiempo estimado para completar cada tarea.
5. Registramos el tiempo estimado de espera entre las tareas del proceso o actividad.
6. Si alguna tarea se repite registramos el promedio de repeticiones.
7. Calculamos el tiempo total de trabajo y el tiempo desperdiciado.
8. Creamos un VSM-PS con la representación grafica de lo que registramos recomendada. A continuación la simbología usada para la creación de la representación grafica.

Flujo de información

Tareas

Tarea

Línea de Tiempo. Las crestas indican tareas que agregan valor los valles indican tareas o acciones que no agregan valor.

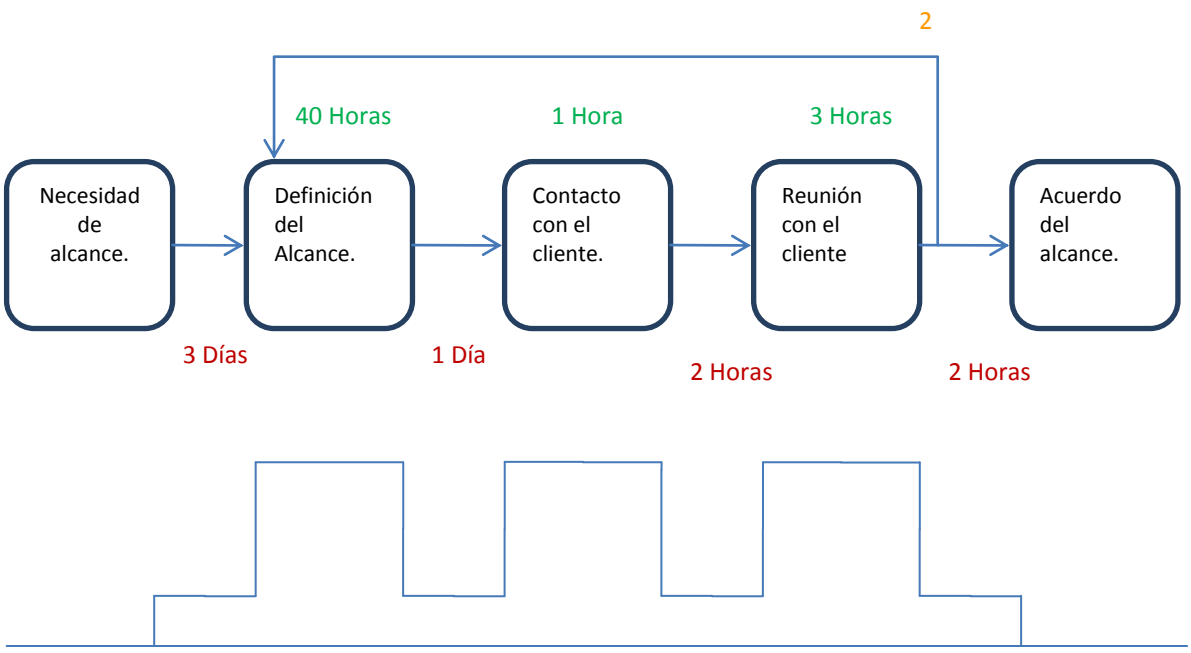
Tiempo de trabajo : X Hrs, X Días
Tiempo Desperdiciado: X Hrs, X Días
Indicaciones de Ciclos. X

Movimiento de producto

Movimiento de Producto Terminado

Posible Mejora

Para visualizar mejor los pasos a continuación un ejemplo de cómo se construye el VSM-PS.
Ejemplo. Un equipo de trabajo comenzará a trabajar en un proyecto nuevo, pero antes, necesita definir y obtener la aprobación de los alcances de un proyecto, para esto necesita realizar las siguientes tareas:
Necesidad definir alcance.
Definición del Alcance.
Contacto con el cliente.
Reunión con el cliente
Acuerdo del alcance.



Tiempo Total de Trabajo
(40+1+3)*2= 88

Tiempo Transcurrido
24+(40+8+1+4+3)*2=136 Horas

Tiempo Desperdiciado
3(8)+(8+2+2)(2)=

Eficiencia
64.7%

Conocimientos y habilidades
Conocimiento del proceso o actividad a mapear.
Criterios de verificación
Acuerdo de los involucrados de que estas son las tareas necesarias para completar el proceso o actividad. Acuerdo de los involucrados de que los tiempos estimados son correctos. Ya sea para completar las tareas y los tiempos de espera entre tareas.
Métricas
Tiempo estimado dedicado por todos los involucrados a esta a practica(Hrs) Tiempo real transcurrido(Hrs)

VSM-FS	**Práctica**
Mapa de flujo de Valor de estado Futuro o Mapa de cadena de valor de estado Futuro(Value Stream Map Future State)	

Objetivo	
Crear una representación gráfica de un proceso en donde queden plasmadas todas las actividades de planeación y creación de un producto se software con una propuesta de mejora al proceso presente.	
Entrada	**Resultado**
Condición Conocimiento del proceso de los miembros de la compañía. *Producto* VSM-PS (Mapa de flujo de Valor de estado Presente o Mapa de cadena de valor de estado Presente)	*Condición* El equipo realizo un análisis sobre el proceso y detecto las posibles fuentes de desperdicio. *Productos* Mapa de flujo de valor de estado futuro. Propuesta de mejora al proceso. Plan de Cambios.
Guía	
Actividades	

Definiciones.

Valor. El valor de un producto de software es una cualidad que es descrita en la medida que el producto satisface las necesidades del cliente. Algunas cualidades de un producto de software son:

Correcto
Confiable
Robusto
Eficiente
Amigable
Verificable
Reusable
Portable
Interoperable
Productivo
A Tiempo
Visible
Cohesivo
Desacoplado
Comprensible
Mantenible

Cadena de valor O flujo de valor. Es un concepto teórico que permite describir el desarrollo de las actividades de una organización. Visualizándolo como una cadena de eslabones, que inicia con el cliente y termina con el cliente. Cada eslabón debe agregar valor al producto de software.

Tiempo total de trabajo. En un proceso o actividad, es la suma de los tiempos usados para completar las tareas propias del proceso o actividad.

Tiempo desperdiciado. En un proceso o actividad, es la suma de los tiempos donde no se realiza ninguna tarea.

Tiempo Transcurrido. En el proceso o actividad el tiempo total transcurrido, la suma del tiempo total de trabajo más el tiempo desperdiciado.

Eficiencia. Indicador en forma de porcentaje que muestra cuando del tiempo transcurrido ha sido dedicado al trabajar.

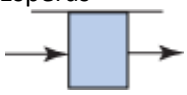
Desperdicios

Los 7 desperdicios en el desarrollo de software según Lean.

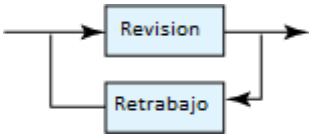
1. Trabajo parcialmente hecho
2. Reaprendizaje
3. Características Extra
4. Transferencias
5. Atrasos
6. Cambio de tareas
7. Defectos

Nosotros debemos identificar las posibles fuentes de desperdicio haciendo un análisis en el cual podemos encontrar:

Esperas



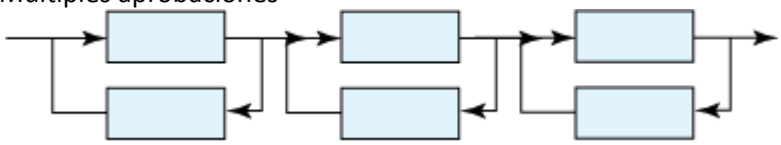
Revisiones Innecesarias



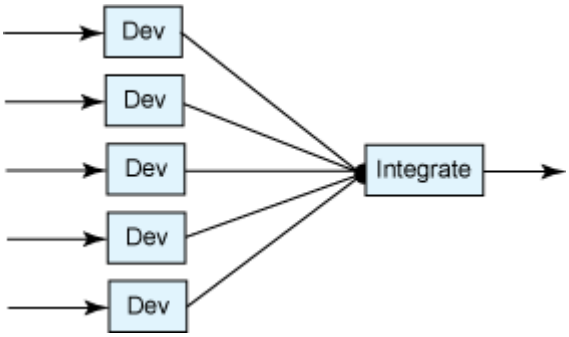
Callbacks. Cuando se establece comunicación de ida y vuelta entre una o más partes.

Listas. Las listas de tareas generan esperas. En general una lista de tareas en desarrollos agiles no debe ser más grande que las tareas que puedes completar en 2 iteraciones.

Múltiples aprobaciones



Concurrencia

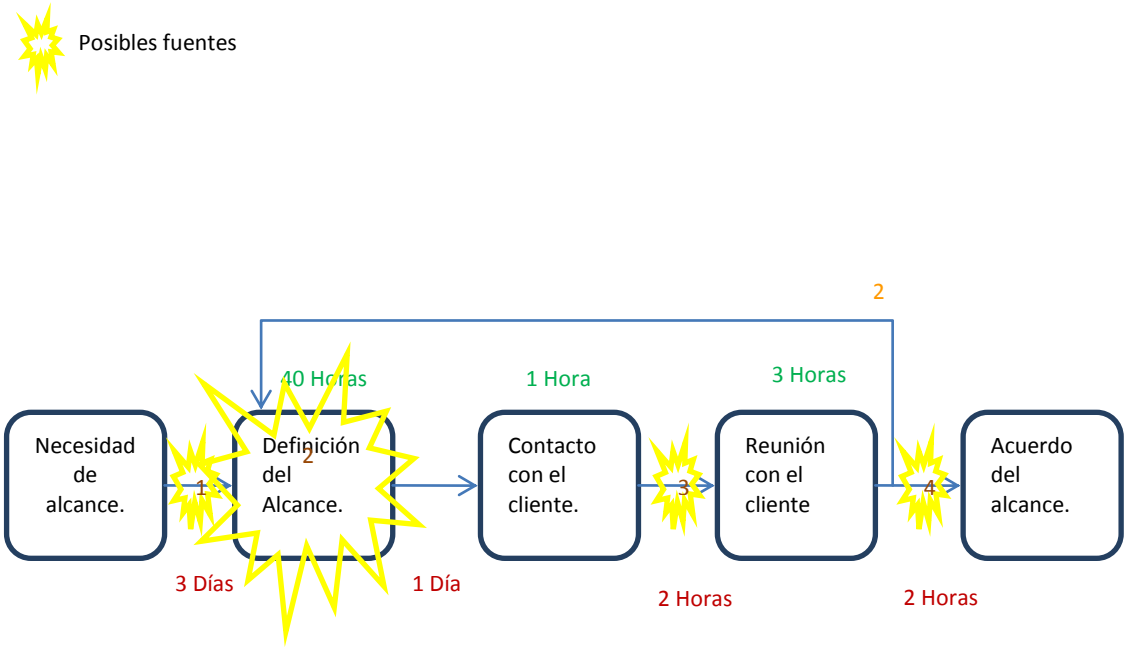


Largos ciclos de prueba. En proporción al esfuerzo dedicado al desarrollo.

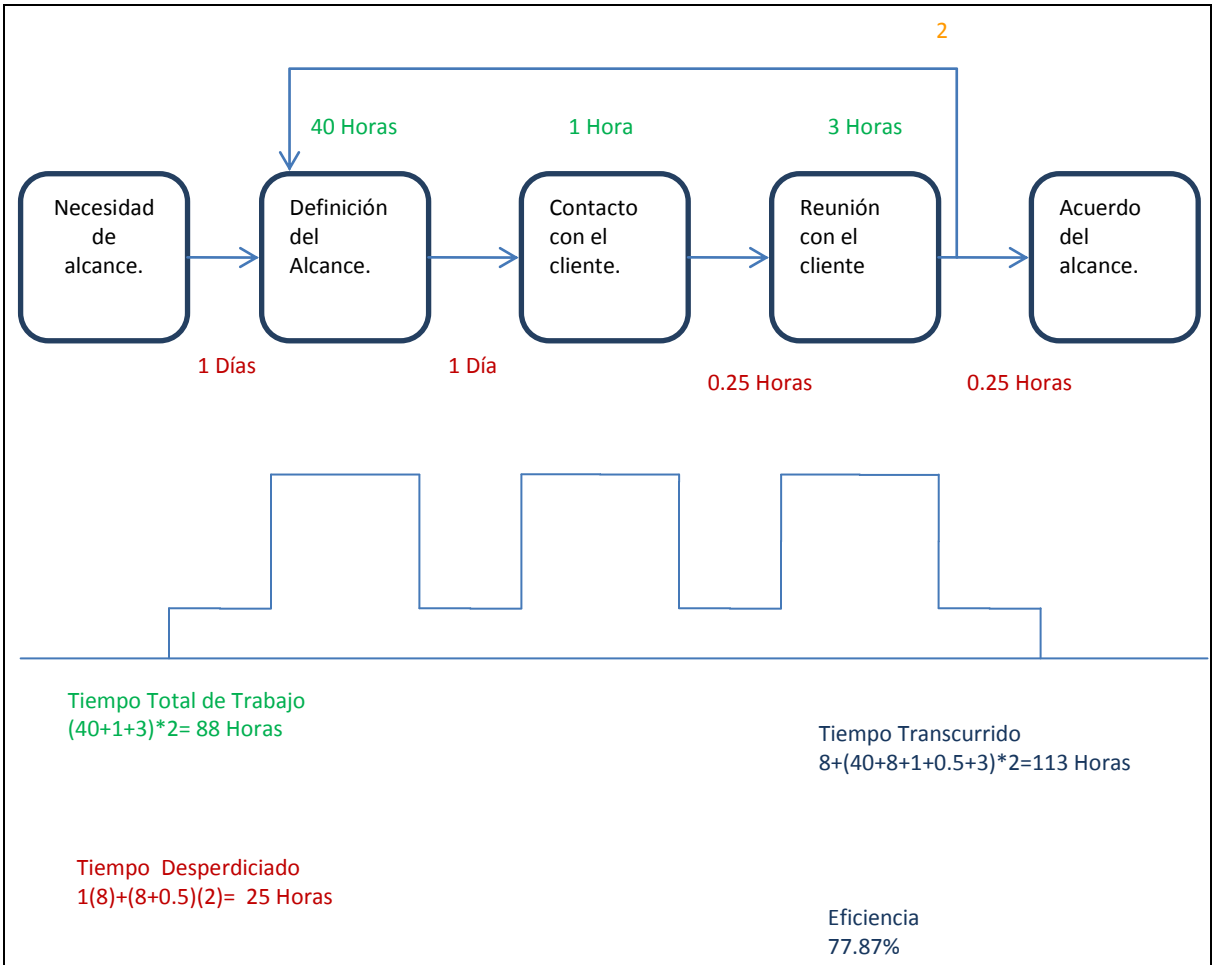
Creación del Mapa de flujo de Valor de estado Futuro

1. Identificar en el VSM-PS las posibles fuentes de desperdicio apoyados en la lista de 7 desperdicios de desarrollo de software según lean.
2. Analizamos posibles cambios para eliminar el desperdicio o al menos disminuirlo.

3. Dibujamos un nuevo mapa de flujo de valor que contenga los cambios que hemos analizado.
4. Creamos una propuesta de cambios y si es necesario un plan de cambios.
Ahora contemplamos un ejemplo de cómo se vería un VSM-PS donde identificamos las posibles fuentes de desperdicio:



Ahora un VSM-FS que muestra la propuesta de cambios.



Propuesta de cambios

1. La comunicación interna en la empresa, reducir el tiempo en que es asignado un proyecto a un equipo de trabajo de manera inicial puede ser reducido. Equipos pre-armados, planeación y liberación de recursos.
2. Dentro de la definición del alcance puede generarse desperdicio (por ejemplo, funcionalidades extra, trabajo parcialmente hecho, etc.) hay que analizar la actividad de "Definición del alcance".
3. Se puede hacer la reunión con el cliente de manera remota, una reunión con video-conferencia, así se ahorra el tiempo de transporte, en el VSM-PS se considera el caso en el que el cliente se encuentra en la misma ciudad, pero esto ayudara mas cuando el cliente no se encuentre en la misma ciudad.
4. Ver propuesta 3.

Conocimientos y habilidades
Conocimiento del proceso o actividad a mapear. Conocimiento en el Lean Software Development para reconocer los desperdicios y generar propuestas de mejora.
Criterios de verificación
Acuerdo de que esas son las tareas necesarias para completar el proceso o actividad. La evaluación y verificación de que las propuestas de cambio y el plan de cambio sean factibles

para la organización y la condición actual del entorno de negocio.
Métricas
Tiempo de Trabajo total (Hrs) Tiempo Desperdiciado(Hrs) Tiempo Transcurrido(Hrs) Eficiencia (%)

2.3.2 Pra2. Retroalimentación.

La retroalimentación tiene como objetivo hacer mas estrecha la relación entre el desarrollo y el cliente. Mediante reuniones cortas con los clientes se determina la fase actual de desarrollo y se proponen los ajustes necesarios para introducir mejoras en el futuro. Durante las reuniones, tanto los clientes como el equipo de desarrollo, logran aprender sobre el alcance del problema y buscan posibles soluciones para un mejor desarrollo. Por lo tanto, los clientes comprenden mejor sus necesidades basándose en el resultado de los esfuerzos del desarrollo y los desarrolladores aprenden a satisfacer mejor estas necesidades.

2.3.3 Retroalimentación extendida

Otra idea para extender la retroalimentación es a través de la integración del cliente en el ambiente de desarrollo para concentrar la comunicación en las soluciones futuras y no en las soluciones actuales, promoviendo así el nacimiento de la solución a través del diálogo con el cliente.

2.3.4 Pra3. Iteraciones

El objetivo de las iteraciones es generar plazos de entrega mas cortos .Las Iteraciones promueven la retroalimentación, la comunicación (interna y con el cliente) el uso efectivo de los recursos así como el ver metas a corto plazo. Crean flexibilidad y mejor respuesta a los cambios y corrección de defectos.

Al planear las iteraciones usamos la especificación que sea necesaria, casos de uso, historias de usuario o products backlog. Al inicio de cada iteración se hace una reunión de equipo para evaluar el trabajo de la iteración (dificultad y prioridad) así planear el costo de la iteración. El equipo debe verificar que las actividades de esa iteración puedan ser completadas en el tiempo planeado para la iteración.

2.3.5 Pra4. Sincronización.

El objetivo de la sincronización es concientizar de las dependencias de trabajo a los involucrados en el proyecto y propagar rápidamente la información. Generalmente se dividen las tareas a trabajar pero una parte de una tarea (código) necesitará de alguna parte de otra tarea (código) lo cual hace necesario que estas se sincronicen, tener un repositorio común en donde se pueda trabajar en un ambiente donde los cambios sean rápidamente difundidos y el código necesario pueda ser usado por todos.

2.3.6 Pra5. Desarrollo basado en conjuntos

El objetivo del desarrollo basado en conjuntos es crear espacios de solución donde existe más flexibilidad y opciones. El desarrollo basado en conjuntos nos da un espacio de solución a los problemas implantados por el proyecto, sin definir una solución definitiva.

Así tenemos opciones para una solución permitiéndonos tomar una decisión definitiva lo más tarde posible. Con mejor información puede, en el mejor de los casos, que la solución emerja por si misma, claro ya basada en el aprendizaje acumulado por parte del cliente y del equipo de desarrollo.

2.3.7 Pra6. Pensando en opciones

El objetivo de pensar en opciones es buscar la flexibilidad en las decisiones relacionadas al proyecto. El desarrollo de software está siempre asociado con cierto grado de incertidumbre, los mejores resultados se alcanzan con un enfoque basado en opciones por lo que se pueden retrasar las decisiones tanto como sea posible hasta que éstas se basen en hechos y no en suposiciones y pronósticos inciertos. Cuanto más complejo es un proyecto, más capacidad para el cambio debe incluirse en éste, así que debe permitirse el retraso de los compromisos importantes y cruciales. El enfoque iterativo promueve este principio: la capacidad de adaptarse a los cambios y corregir los errores, ya que un error podría ser muy costoso si se descubre después de la liberación del sistema.

2.3.8 Pra7. Responder hasta el último momento

El objetivo de responder es llevarles opciones rápidamente a los clientes, lo que implica, retrasar algunas decisiones cruciales hasta que los clientes hayan reconocido mejor sus necesidades.

2.3.9 Pra8. Tomar decisiones

El objetivo de tomar decisiones es cuando saber tomarlas. La adaptación tardía a los cambios no previene las costosas decisiones delimitadas por la tecnología. Esto significa que es necesaria la planificación en el proceso, pensando en esto, las actividades de planificación deben centrarse en las diferentes opciones y se les adapta a la situación actual; así como, se deben clarificar las situaciones confusas estableciendo las pautas para una acción rápida. Evaluar las diferentes opciones es eficaz tan pronto como queda claro que solo una opción será viable, pero proporcionando la flexibilidad necesaria para una tardía toma de decisiones.

2.3.10 Pra9. Sistemas Pull

El objetivo de los sistemas Pull es describir una forma de trabajo en donde el trabajador solicita lo que necesita hacer. Hay dos formas en las que se puede asegurar que un trabajador (en nuestro caso un desarrollador) haga un uso efectivo de su trabajo (Tiempo laboral), diciéndole que hacer o que el mismo encuentre que hacer. En un ambiente tan dinámico como el desarrollo solo la segunda opción funciona.

Cuando las cosas funcionan rápidamente, no hay suficiente tiempo para que la información viaje en la pirámide organizacional, por esto una forma efectiva es que localmente se sepa que hacer y que ha sido hecho, para que cada desarrollador pueda saber qué hacer cuando no tenga una tarea por realizar.

2.3.10.1 Pra9.1 Just in Time

La práctica de producción Just In Time aplicada a la industria de desarrollo de software, tiene como objetivo reconocer las necesidades específicas para el desarrollo y el ambiente o contexto del negocio involucrado y permitir que el equipo de desarrollo se organice y divida las tareas autónomamente para lograr el resultado necesario para una iteración específica.

Al principio, el cliente dispone los requisitos necesarios. Esto podría ser simplemente presentar los requisitos en pequeñas fichas o historias y los desarrolladores estimarán el tiempo necesario para la implementación de cada tarjeta. Así, la organización del trabajo cambia en un sistema autopropulsado: cada mañana durante una reunión inicial cada

miembro del equipo evalúa lo que se ha hecho ayer, lo que hay que hacer hoy y pregunta por cualquier nueva entrada necesaria de parte de sus colegas o del cliente. Esto requiere la transparencia del proceso, que es también beneficioso para la comunicación del equipo.

2.3.11 Pra10. Teoría de filas.

El objetivo de Teoría de filas es reconocer y minimizar lo mas posible la formación de filas en le proceso de desarrollo. Una de las peores cosas que suceden en la vida son las filas, hacer fila para pagar el teléfono, para hacer un trámite, estar en la fila esperando a que alguien nos atienda o algo este disponible, es tiempo muerto. Hay que evitar las filas o al menos reducir el tiempo que se está en esta, puede ser manteniendo una secuencia constante del uso de los recursos susceptibles a las filas y manteniendo constante nivel de servicio así también anticipando la demanda para que esta no sobrecargue el sistema.

2.3.12 Pra11. El costo de atrasar

El objetivo de esta práctica es proporcionar una serie de consejos que ayuden a evitar los atrasos. Los atrasos son muy costosos, más costosos de lo que a veces podemos creer y hay que evitarlos a toda costa. Lamentablemente en la industria del desarrollo de software los atrasos son muy comunes, ya que en esta industria los desarrolladores deben balancear entre terminar correctamente una tarea o cumplir con los costos y tiempos. Una forma de ayudar a balancear esto es que cada miembro del equipo tenga una idea de los costos y del negocio y como son impactados cuando hay un atraso y bien pues ayudar tratar de reducir el impacto tomando mejores decisiones enfocadas al éxito económico.

2.3.13 Pra12. Autodeterminación (Work-Out)

El objetivo del Work-Out es permitir que el flujo de la información pase de los trabajadores a sus superiores y que estos reciban la información la analicen y la usen en favor del proyecto. Existe una creencia en la mayoría de las empresas acerca de la toma de decisiones en la organización: los administradores y ejecutivos tienen que decirle a los trabajadores cómo hacer su trabajo. En esta practica llamada Work-Out, los roles cambian: a los directivos se les enseña a escuchar a los desarrolladores, de manera que éstos puedan explicar mejor qué acciones podrían tomarse, así como ofrecer sugerencias para mejoras. Los administradores de proyecto más experimentados simplemente han

declarado la clave de éxito de los proyectos: "Buscar la buena gente y dejarles hacer su propio trabajo".

2.3.14 Pra13. Motivación

El objetivo de la Motivación es mantener en mente que todos los involucrados en un desarrollo son personas y deben ser escuchadas y respetadas. Las personas podrían ser los recursos desde el punto de vista de una hoja de datos estadísticos, pero en el desarrollo de software, así como cualquier organización de negocios, las personas necesitan algo más que la lista de tareas y la seguridad de que la lista no será alterada durante la realización de las tareas. Las personas necesitan motivación y un propósito superior para el cual trabajar: un objetivo alcanzable dentro de la realidad con la garantía de que el equipo puede elegir sus propios compromisos. Los desarrolladores deberían tener acceso a los clientes; el líder(es) de equipo debe(n) proporcionar apoyo y ayuda en situaciones difíciles, así como asegurarse de que el escepticismo no arruine el espíritu de equipo.

2.3.15 Pra14. Liderazgo

El objetivo del Liderazgo es mantener alto le espíritu y cohesión del equipo gracias a las habilidades y atención de el líder. En cualquier desarrollo hay un líder o un grupo de líderes que toman la batuta de un proyecto, ya sea por sus capacidades técnicas y/o personales o por las características de su rol dentro del proyecto, figuras reconocidas serian los líderes técnicos, líder de equipo, líderes sociales y los administradores de proyectos. En los líderes y administradores recaen varias actividades de liderazgo, las herramientas presentadas por Lean están primeramente enfocadas a los líderes para que estos puedan esparcirlas en sus equipos de trabajo. Uno de sus principales objetivos como líderes es mantener la cohesión de sus equipos, evitar que estos usen antipatrones [1] y hacer que sus equipos lleguen a las metas se han impuesto.

---

[1] Los antipatrones son soluciones negativas que presentan más problemas que los que solucionan. (Brown , Malveau, McCormick, & Mowbray, 1998, pág. 6)

2.3.16 Pra15. Experiencia

La experiencia es una herramienta muy útil que es usada cuando hay problemas ya que en base a ella podemos llegar más rápidamente a una solución. Lo mejor que puede hacer una organización o un equipo de trabajo es usar la experiencia de sus miembros, que ellos contribuyan para crear soluciones.

2.3.17 Pra16. Refactorización

El objetivo de la refactorización es mejorar la facilidad de comprensión del código o cambiar su estructura y diseño y eliminar código muerto, para facilitar el mantenimiento en el futuro. Una de las maneras más saludables hacia una arquitectura de integración es la refactorización. Cuantas más funcionalidades se añaden a las del sistema, mas se pierde del código base para futuras mejoras. Así como en la Programación extrema (XP), la refactorización es mantener la sencillez, la claridad, la cantidad mínima de funcionalidades en el código. Las repeticiones en el código son signo de un mal diseño de código y deben evitarse.

2.3.18 Pra17. Pruebas.

El objetivo de las pruebas es verificar que el producto realiza las tareas para las que fue diseñado. El completo y automatizado proceso de construcción debe ir acompañada de una suite completa y automatizada de pruebas, tanto para desarrolladores y clientes que tengan la misma versión, sincronización y semántica que el sistema actual. Al final, la integridad debe ser verificada con una prueba global, garantizando que el sistema hace lo que el cliente espera que haga. Las pruebas automatizadas también son consideradas como parte del proceso de producción y, por tanto, si no agregan valor deben considerarse residuos. Las pruebas automatizadas no deberían ser un objetivo, sino, un medio para un fin; específicamente para la reducción de defectos.

2.3.19 Pra18. Métricas

El objetivo de las métricas es obtener datos concretos y cuantificables que reflejen de una u otra manera las características de un producto o el ambiente relacionado.Medir es una herramienta para saber cuál es el rendimiento para poder optimizar, y hay que recordar que hay que ver el sistema entero para poder optimizar, una creencia errónea es que si cada parte del sistema es optimizada el sistema se encontrara optimizado. Si hay algo que no pueda ser medido por completo, no hay que medirlo del todo porque una medición a

medias nos dará una optimización a medias lo cual es igual o peor que no haber optimizado, lo mejor que podemos hacer es estandarizar las fases de desarrollo abstrayéndolas y así podemos ver como cada fase debería ser hecha y asi podemos medirla. Especifica, crea un plan y mide el desempeño respecto a este hay que encontrar cuales son las variaciones con respecto al plan. Descompone los grandes problemas en pequeñas tareas, así mide cada tarea de manera individual.

2.3.20 Pra19. Contratos

Los contratos por definición son un acuerdo entre 2 partes o más, que buscan un intercambio de bienes y/o servicios, para los movimientos agiles los contratos tienen grandes dificultades ya que estos en la mayoría de los casos son de precio fijo, los cuales hacen necesaria una corrección al contrato y al precio si es que se encuentran cambios muy grandes que afecten al desarrollo pactado, la flexibilidad inherente de los desarrollos agiles necesita que esta flexibilidad este reflejada en el contrato, en la confianza que debe existir entre el cliente y los desarrolladores para así mejorar y facilitar el ambiente ágil y beneficioso de desarrollo.

2.4 LSD y el manifiesto ágil

LSD es considerada una metodología ágil de desarrollo de software ya que coincide con muchos de los principios descritos en el manifiesto ágil (Anexo 2.).

Encontramos coincidencias en temas como el de las entregas tempranas, las cuales buscan crear productos funcionales que permiten iteraciones. Desde el punto de vista LSD las iteraciones cortas permiten crear conocimiento y que el cliente descubra sus necesidades haciendo posible la incorporación de ajustes sin que estos afecten al desarrollo de manera significativa y si aumenten el valor del producto para el cliente.

También encontramos coincidencias en el tema de la planeación, en el que LSD se evita hacer compromisos los cuales después acarren grandes cantidades de re trabajo.

LSD y el manifiesto ágil tienen como foco en el valor para el cliente. Como negocio, como persona y como organización el valor del producto de software es en la medida de que

este producto satisface las necesidades del cliente y facilita sus procesos (productivos, administrativos, etc…).

Y por ultimo, el manifiesto ágil y LSD concuerdan en el lado humano del desarrollo de software, dando gran valor a las personas que integran los equipos de desarrollo. Se hace énfasis a la comunicación entre personas, así como atender a las necesidades de los equipos de trabajo, todo esto, buscando un ambiente que promueva la armonía y comunicación efectiva. Elementos necesarios como cualquier otro recurso material para que un proyecto cumpla con sus objetivos de tiempo, costo y calidad.

3 ISO/IEC 29110 Perfil Básico.

La guía de Administración e Ingeniería para Perfil Básico aplica a una Organización Pequeña (Empresa, organización, departamento o proyecto de hasta 25 personas) dedicada al desarrollo de software.

La guía provee un marco de Administración del Proyectos y Procesos de Implementación de Software que integra las prácticas basadas en la selección del estándar ISO/IEC 12207 - Sistemas e Ingeniería de Software - Procesos del Ciclo de Vida del Software: 2008 e ISO/IEC 15289 Sistemas e Ingeniería de Software - Procesos del Ciclo de Vida del Software - directrices para el contenido de los productos de información (documentación) sobre procesos de ciclo de vida del software:2006 elementos estándares.

El documento, Ingeniería de Software — Perfiles del Ciclo de Vida para Organizaciones Pequeñas (OPs) — Parte 5-1-2: Guía de Administración e Ingeniería – Perfil Básico OP (Guía), está destinado para ser utilizado por una OP para establecer procesos para implementar cualquier enfoque o metodología de desarrollo, incluyendo, por ejemplo, ágil, evolutivo, incremental, guiado por pruebas etc., basada en las necesidades de la OP o del proyecto. Que en el caso de este trabajo de tesis se hace una adaptación para la implementación de procesos de una metodología ágil (Lean Software Development)

Usando la Guía, la organización pequeña puede obtener beneficios en los siguientes aspectos:

- Contar con un listado, acordado con el cliente, de los requerimientos del proyecto y los productos esperados del mismo;

- Se lleva a cabo un proceso de administración disciplinado, que proporciona visibilidad y acciones correctivas sobre los problemas y desviaciones del proyecto;

- Se sigue un proceso sistemático de implementación de software, que satisface las necesidades del cliente y garantiza productos de calidad.

Para el uso de la Guía, la organización pequeña necesita cumplir con las siguientes condiciones:

- El enunciado de trabajo del proyecto debe estar documentado;

- La viabilidad del proyecto debe ser analizada de manera previa;

- El equipo del proyecto, incluyendo el administrador del proyecto, debe haber sido asignado y entrenado; y

- Se debe contar con bienes, servicios e infraestructura disponible para iniciar el proyecto.

El propósito del proceso de Administración del Proyecto es establecer y realizar de modo sistemático las tareas del proyecto de implementación de software, las cuales permiten cumplir con los objetivos del proyecto dentro del marco de calidad, tiempo y costo esperados.

El propósito del proceso de Implementación de Software es sistematizar la realización de las actividades de análisis, diseño, construcción, integración y pruebas para productos de software nuevos o modificados, de acuerdo a los requerimientos especificados.

Ambos procesos se encuentran interrelacionados (ver Figura 1).

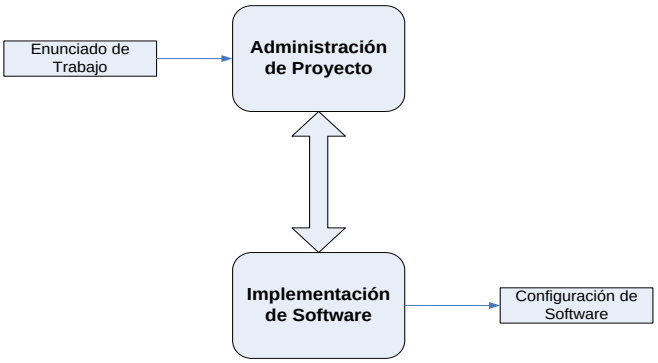


Figura 1 – Procesos guía del perfil básico

4 KUALI-BEH.

KUALI-BEH describe los conceptos comunes y sus relaciones presentes en cualquier proyecto de software. KUALI-BEH es una propuesta mexicana a la solicitud del Object Management Group A Foundation for the Agile Creation and Enactment of Software Engineering Methods.

KUALI-BEH se compone de dos vistas: estática y operacional (Figura 1).

- La vista **estática** proporciona un marco de trabajo para la definición de las diferentes formas de trabajo de los practicantes. Estas formas de trabajo son organizadas en métodos compuestos por prácticas. Este conocimiento se estructura en una infraestructura de métodos y prácticas que pueden ser aplicadas por los practicantes.

- La vista **operacional** está relacionada con la ejecución del proyecto de software. Esta vista proporciona a los practicantes mecanismos para ejecutar y adaptar un método con sus prácticas de acuerdo a las necesidades de los principales interesados y a un contexto específico.

Los conceptos de KUALI-BEH se pueden aplicar para definir métodos y prácticas de cualquier proyecto de software, independientemente del tamaño, complejidad, modelo de ciclo de vida utilizado o tecnología.

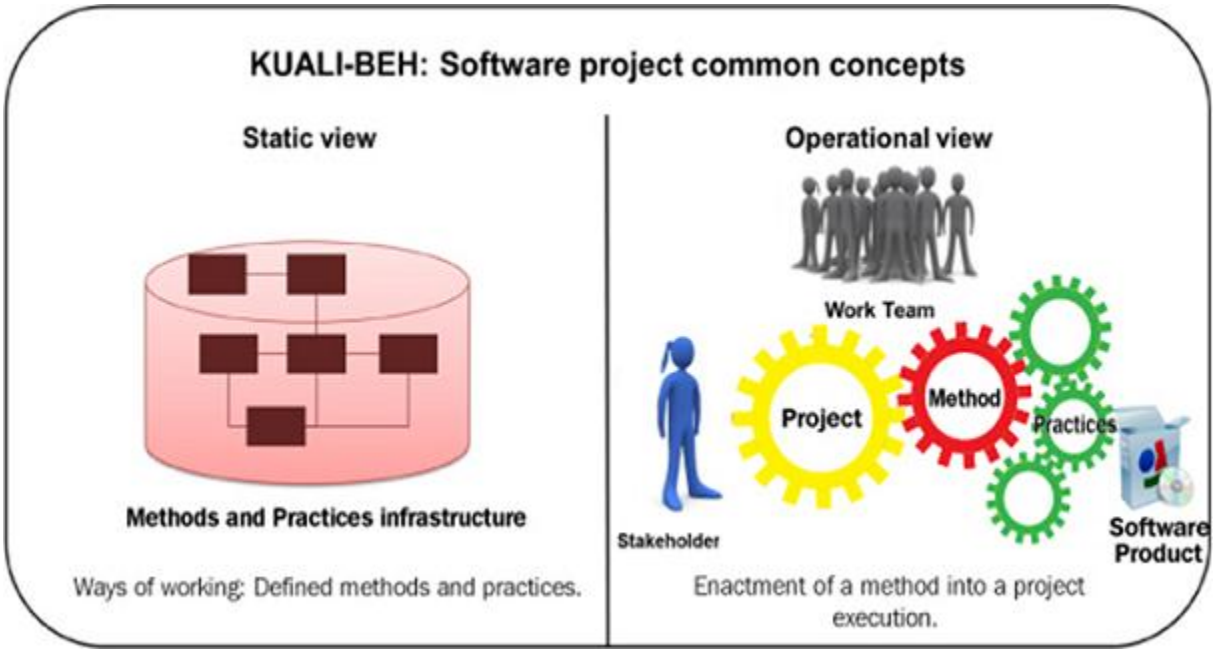


Figura 1 – Vistas

Para este proyecto de tesis usaremos la definición de prácticas de la vista estática, dada una plantilla. (Anexo 1

Capítulo 3. Procesos

El objetivo de este capítulo es abordar el tema de estudio desarrollado dentro de los procesos del ISO/IEC 29110 Perfil Básico haciendo las recomendaciones pertinentes según las herramientas y filosofía de Lean Software Development. Para cumplir con ese objetivo se tomaron las tablas de definición de tareas y productos del ISO/IEC 29110 y se modificaron como a continuación se muestra para dar cabida a las recomendaciones LSD así como mostrar la relación de esas recomendaciones con las practicas y principios descritos en el capitulo anterior.

Definición de la tabla de tareas con las recomendaciones LSD

Rol	**Lista de tareas**	**Productos de entrada**	**Productos de salida**	**Practicas y principios Lean sugeridos**
Siglas de los roles involucrados	Definición de la tarea según el ISO/IEC29110 **Recomendaciones de LSD*	Productos de entrada según el ISO/IEC29110	Productos de salida según el ISO/IEC29110	Practicas y principios de LSD relacionados

Definición de la tabla de productos con la visión LSD.

	Nombre	**Descripción**	**Visión Lean**
#	Nombre del producto según el ISO/IEC29110	Descripción del producto según el ISO/IEC29110	Características a considerar según el LDS

1 Proceso de Administración del Proyecto (AP)

El proceso de Administración del Proyecto (AP) utiliza el enunciado de trabajo del cliente para elaborar el plan del proyecto. Las tareas de evaluación y control del proyecto comparan el progreso del proyecto contra el plan del proyecto y se toman las acciones

necesarias para eliminar desviaciones o incorporar cambios al plan del proyecto. La actividad de cierre del proyecto consiste en la entrega de la configuración de software, producido por la Implementación de Software, y se obtiene la aceptación por parte del cliente para formalizar el fin del proyecto. Un repositorio del proyecto se establece para guardar los productos de trabajo y controlar sus versiones durante el proyecto.

1.1 Listas de Productos del proceso de administración

1.1.1 AP productos de entrada

Nombre	Origen
Enunciado del Trabajo	Cliente
Configuración del Software	Implementación de Software
Solicitud de Cambio	Cliente Implementación de Software

1.1.2 AP productos de salida

Nombre	Destino
Plan del proyecto	Implementación de Software
Documento de Aceptación	Alta Dirección
Repositorio del Proyecto	Implementación de Software
Minuta	Cliente
Configuración del Software	Cliente

1.1.3 AP productos internos

Nombre
Solicitud de Cambio
Registro de Defectos

Minuta
Lista de Verificación
Reporte de Avance
Respaldo del Repositorio del Proyecto

1.2 Lista de tareas del proceso de administración con recomendaciones LSD

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
AP LT ET	AP.1.1 Revisar el *Enunciado de trabajo.* **Se recomienda que el Equipo de Trabajo tenga acceso y pueda revisar en conjunto el Enunciado de trabajo. De esta manera desde un inicio se promueve una visión clara y acordada del proyecto.*	Enunciado de trabajo	Enunciado de trabajo [revisado]	Pra14. Liderazgo Pra13. Motivación Pra11. El costo de atrasar Pra12. Autodeterminación (Work-Out)
AP CL	AP.1.2 Definir con el cliente las *Instrucciones de entrega* para cada uno de los entregables especificados en el *Enunciado de trabajo.* **Hay que recordar que dentro de la visión Lean, se recomienda siempre el contacto cara a cara. Se recomienda también mantener la documentación al mínimo posible, teniendo en mente la practicidad de los documentos generados y la disponibilidad de estos para todo el ET.*	Enunciado de trabajo [revisado]	Plan del Proyecto - Instrucciones de entrega	Pri1.Eliminar los desperdicios Pra1. Value Stream Mapping
AP LT	AP.1.3 Identificar las tareas específicas a realizar para producir los entregables y sus componentes de software	Enunciado de trabajo [revisado]	Plan del Proyecto	Pri1.Eliminar los desperdicios

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
ET	identificados en el *Enunciado de trabajo.* Incluir las tareas del proceso de Implementación de Software sobre verificación, validación y revisiones con los Clientes y el Equipo de Trabajo para asegurar la calidad de los productos. Identificar las tareas para realizar las *Instrucciones de entrega.* Documentar las *Tareas.* **Se recomienda también mantener la documentación al mínimo posible, teniendo en mente la practicidad de los documentos generados y la disponibilidad de estos para todo el ET.*		- Tareas	Pra1. Value Stream Mapping
AP LT ET	AP.1.4 Establecer la *Duración estimada* para realizar cada tarea. **Se recomienda permitir que el ET se organice (Autodeterminación). Para esto se reúne el ET completo para poder establecer estimaciones lo mas aproximadas posibles a la realidad para así poder acordar la duración de cada una de las tareas definidas.*	Plan del Proyecto - Tareas	Plan del Proyecto - Duración estimada	Pra12. Autodeterminación (Work-Out) Pra15. Experiencia Pra1. Value Stream Mapping
AP LT	AP.1.5 Identificar y documentar los recursos: humanos, materiales, equipo y herramientas, estándares, incluyendo la capacitación requerida para que el Equipo de Trabajo pueda realizar el proyecto. Incluir las fechas en el calendario cuando sean requeridos los recursos y la capacitación. **Se recomienda que el ET comparta el espacio físico donde se llevara acabo el*	Enunciado de trabajo	Plan del Proyecto - Recursos	Pra10. Teoría de filas.

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
	desarrollo así si surge alguna duda o algún problema puede ser atacado por todos los miembros del equipo aportando todos ideas y experiencia para poder obtener la mejor solución posible. También así se eliminan los tiempos de espera para comunicarse, sea para desplazarse, para recibir contestación de un correo electrónico o para que un recurso este disponible. Recordar que se recomienda en todo caso la comunicación cara a cara. Dentro de los recursos materiales hay que analizar cuales de estos pueden ser concurrentes y posibles candidatos a cuellos de botella, los cuales deben ser previstos y de ser posible eliminados, ya sea mediante una calendarización o la redundancia de el recurso.			
AP LT LS	AP.1.6 Establecer la *Estructura del Equipo de Trabajo,* asignando los roles y responsabilidades acordes a los *Recursos.* **Se recomienda que los equipos se formen tomando en cuenta habilidades, experiencia y capacidades las cuales fomenten un espacio que promueva un ambiente de trabajo eficiente y dinámico, con buen espíritu. El Líder Social aportara su ayuda para promover el espíritu de equipo y con el conocimiento de sus compañeros y respectivas personalidades podrá hacer recomendaciones para formar los equipos.*	Plan del Proyecto - Recursos	Plan del Proyecto - Estructura del Equipo de Trabajo	Pra12. Autodeterminación (Work-Out) Pra13. Motivación Pra14. Liderazgo
AP	AP.1.7 Asignar la estimación inicial y las fechas determinadas para cada tarea	Plan del Proyecto	Plan del Proyecto	Pra3. Iteraciones

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
LT ET	con el fin de crear el *Calendario de las Tareas del Proyecto* considerando los recursos asignados, la secuencia y dependencia de las tareas. **Se recomienda tomar en cuenta a todos los miembros del equipo para que aporten a la creación del calendario de tareas y este sea visto como un acuerdo y no como una imposición.*	- Tareas - Duración estimada - Estructura del Equipo de Trabajo	- Calendario de las Tareas del Proyecto	Pra5. Desarrollo basado en conjuntos Pra11. El costo de atrasar Pra15. Experiencia Pra16. Refactorización
AP	AP.1.8 Calcular y documentar la *Estimación del Esfuerzo y Costo.* **Acuerdo documentado de todo lo aportado por el ET así como por las necesidades del cliente. Debe estar disponible para todo el ET.*	Plan del Proyecto - Calendario de las Tareas del Proyecto - Recursos	Plan del Proyecto - Estimación del Esfuerzo y Costo	Pri1.Eliminar los desperdicios Pra11. El costo de atrasar Pra1. Value Stream Mapping
AP LT ET	AP.1.9 Identificar y documentar los riesgos que pueden afectar al proyecto. **Se recomienda que el ET aporte en el análisis e identificación de posibles riesgos que puedan afectar al proyecto, se deben tomar en cuenta aspectos internos de la organización, así como propios del equipo, el entorno social y personal.*	Todos los elementos previamente definidos	Plan del Proyecto - Identificación de Riesgos del Proyecto	Pri1.Eliminar los desperdicios Pra11. El costo de atrasar
AP LT	AP.1.10 Documentar la *Estrategia de Control de Versiones* en el *Plan del Proyecto.* **Se recomienda que este documento se mantenga disponible a todos los miembros del ET.*		Plan del Proyecto - Estrategia de Control de Versiones	Pra1. Value Stream Mapping Pri1.Eliminar los desperdicios
AP	AP.1.11 Generar el *Plan del Proyecto*	Todos los elementos	Plan del Proyecto	Pri1.Eliminar los

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
	integrando los elementos previamente identificados y documentados. **Se recomienda que este documento se mantenga disponible a todos los miembros del ET.*	previamente definidos	- Tareas - Duración estimada - Recursos - Estructura del Equipo de Trabajo - Calendario de las Tareas del Proyecto - Estimación del Esfuerzo y Costo - Identificación de Riesgos del Proyecto - Estrategia de Control de Versiones - Instrucciones de Entrega	desperdicios Pra1. Value Stream Mapping Pra9. Sistemas Pull Pra9.1 Just in Time Pra4.Sincronización
AP LT	AP.1.12 Incluir la descripción del producto, el alcance, los objetivos y los entregables en el *Plan del Proyecto*. **Se recomienda que este documento se mantenga disponible a todos los miembros del ET. Este documento hace referencia a un consolidado de otros documentos generados anteriormente.*	Enunciado de Trabajo - Descripción del Producto - Alcance - Objetivos - Entregables	Plan del Proyecto - Descripción del Producto - Alcance - Objetivos - Entregables	Pri1.Eliminar los desperdicios Pra1. Value Stream Mapping
AP LT ET	AP.1.13 Verificar y obtener aprobación del *Plan del Proyecto.* Verificar que todos los elementos del *Plan del Proyecto* son viables y coherentes. Los resultados encontrados son documentados en *Listas de*	Plan del Proyecto	Listas de Verificación Plan del Proyecto [verificado]	Pra14. Liderazgo Pra15. Experiencia

Rol	Lista de tareas	Productos de entrada	Productos de salida	Practicas y principios Lean sugeridos
	Verificación y las correcciones son realizadas hasta que el documento es aprobado por el AP. **Para esta verificación se recomienda involucrar al ET.*			
AP CL	AP. 1.14 Revisar y obtener la aprobación del *Plan del Proyecto.* El Cliente revisa y acepta el *Plan del Proyecto*, asegurándose que los elementos contenidos en el *Plan del Proyecto* corresponden con el *Enunciado de Trabajo*. **Se recomienda que mientras se lleva a cabo esta reunión se tome una grabación ya sea en audio o video de la misma para que después esta pueda ser mostrada al equipo de trabajo para así hacer una transferencia de conocimiento mas transparente y rápida. En el mejor de los casos el ET podría estar presente para dar cabida a que aporte comentarios.*	Plan del Proyecto [Verificado]	Minuta Plan del Proyecto [Aceptado]	Pra2. Retroalimentación. Pra7. Responder hasta el último momento Pra8. Tomar decisiones
AP LT	AP.1.15 Establecer el repositorio del proyecto usando la *Estrategia de Control de Versiones.* **Se recomienda que este repositorio sea un espacio abierto al ET y este disponible en todo momento durante el tiempo que dure el proyecto.*	Estrategia de Control de Versiones	Repositorio del Proyecto	Pri3.Crear Conocimiento

2 Proceso de Implementación de Software (IS)

La ejecución del proceso de Implementación de Software (IS) es impulsada por el plan del proyecto. El proceso de Implementación de Software comienza con una actividad inicial de revisión del plan del proyecto. El plan del proyecto guiará la ejecución del análisis de requerimientos del software, arquitectura y diseño detallado de software, construcción de software, integración y prueba del software, y actividades de entrega de producto.

Para eliminar los defectos, verificación del producto, validación y las tareas de prueba, son incluidas en las actividades del flujo de trabajo.

El cliente establece un enunciado de trabajo como una aportación al proceso de Administración del Proyecto y recibe una configuración de software como resultado de la ejecución del proceso de Implementación de Software.

2.1 Listas de productos del proceso de implementación

2.1.1 IS productos de entrada

Nombre	Fuentes
Plan del Proyecto	Administración del Proyecto
Repositorio del Proyecto	Administración del Proyecto

2.1.2 IS productos de salida

Nombre	Destino
Configuración de software • *Especificación de Requerimientos* • *Diseño de Software* • *Registro de Rastreo* • *Componentes de Software* • *Software* • *Casos y Procedimientos de Prueba* • *Reporte de Pruebas*	Administración del Proyecto

• *Manual de Operación* • *Manual de Usuario* • *Manual de Mantenimiento*	
Solicitud de Cambio	Administración del Proyecto

2.1.3 IS productos internos

Nombre
Listas de Validación
Listas de Verificación

2.2 Lista de tareas del proceso de implementación con las recomendaciones LDS

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
AP LT ET	IS1.1 Revisar el *Plan del Proyecto* actual con los miembros del Equipo de Trabajo con la finalidad de lograr un entendimiento común y obtener su compromiso con el proyecto. **Esta tarea se recomienda ser retomada al inicio de cada iteración, ciclo de desarrollo o Sprint.*	Plan del Proyecto	Plan del Proyecto [revisado]	Pri6.Respetar a las personas
LT ET	IS1.2 Establecer o actualizar el ambiente de implementación. **Se recomienda establecer un ambiente cooperativo ya común en los desarrollos de software actuales, el cual sea flexible y robusto manteniendo la idea de que cada cambio debe ser rápidamente*	Plan del Proyecto [revisado]		Pra4. Sincronización.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	difundido para lograr ver las dependencias.			
LT ET	IS.2.1 Asignar tareas a los miembros del Equipo de Trabajo de acuerdo a cada rol, basadas en el Plan del Proyecto actual. **Basado en la experiencia, el equipo auto administrado encontrar como dividirse las tareas así como establecer el ritmo y asignación de tareas.*	Plan del Proyecto [revisado]		Pra9. Sistemas Pull Pra9.1 Just in Time Pra12. Autodeterminación (Work-Out)
AN CL	IS.2.2 Documentar o actualizar la Especificación de Requerimientos Identificar y consultar fuentes de información (cliente, usuarios, sistemas previos, documentos, etc.) con el propósito de obtener nuevos requerimientos. Analizar los requerimientos identificados para determinar el alcance y la viabilidad. Generar o actualizar la Especificación de Requerimientos. **Se recomienda recordar cuan importante es la comunicación clara y fluida con el cliente, así mismo hacer que la información pueda fluir dentro de la organización y que si hay decisiones que puedan ser atrasadas atrasarlas para obtener mas mejores condiciones posibles para tomar una decisión.*	Plan del Proyecto -Descripción del Producto	Especificación de Requerimientos	Pra2. Retroalimentación. Pra2.1 Retroalimentación extendida Pra4. Sincronización. Pra7. Responder hasta el último momento Pra8. Tomar decisiones
AN LT	IS.2.3 Verificar y obtener la aprobación de la Especificación de Requerimientos.	Especificación de Requerimientos	Listas de Verificación Especificación de	Pra4. Sincronización.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	Verificar que la Especificación de Requerimientos sea correcta, pueda ser probada y sea coherente con la Descripción del Producto. Adicionalmente, revisar que los requerimientos estén completos, sin ambigüedades ni contradicciones. Los resultados encontrados se documentan en Listas de Verificación y las correcciones se realizan hasta que el documento tenga el visto bueno del AN. Si fueran necesarios cambios significativos, se propone una Solicitud de Cambio. **Todo cambio debe ser difundido tan pronto sea aprobado.*	Plan del Proyecto -Descripción del Producto	Requerimientos [verificado] Solicitud de Cambio [propuesta]	
CL AN	IS.2.4 Validar y obtener la aprobación de la Especificación de Requerimientos, Validar que la Especificación de Requerimientos satisfaga las necesidades y sea acorde a las expectativas, incluyendo la usabilidad de la interfaz de usuario. Documentar en las Listas de Verificación los resultados encontrados y realizar las correcciones hasta que el documento sea aprobado por el CL. en Resultados de validación. Las correcciones se realizan hasta que el documento tiene el visto bueno del CL. **Se recomienda que para mejorar la calidad y tiempo usado en este tipo de solicitudes de aprobación no hay mejor*	Especificación de Requerimientos [verificada]	Listas de Validación [validado] Especificación de Requerimientos [validado]	Pra2. Retroalimentación. Pra2.1 Retroalimentación extendida

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	forma que teniendo presente al cliente en el sitio de desarrollo o si no al menos a un representante del cliente el cual pueda llevar la información a la persona responsable de la aprobación.			
AN	IS.2.5 Documentar la versión preeliminar del Manual de Usuario o actualizar el manual existente. (opcional) **Se recomienda analizar el momento en que se puede crear una versión preliminar.*	Especificación de Requerimientos [validado]	Manual de Usuario [preeliminar]	Pra7. Responder hasta el último momento Pri1.Eliminar los desperdicios
AN LT	IS.2.6 Verificar y obtener la aprobación del Manual de Usuario. Verificar la consistencia del Manual de Usuario con la Especificación de Requerimientos. Los resultados encontrados se documentan en la Lista de Verificación y las correcciones se realizan hasta que el documento es aprobado por el AN. Si fueran necesarios cambios significativos, se propone una Solicitud de Cambio. (opcional)	Manual de Usuario [preeliminar] Especificación de Requerimientos	Lista de verificación [preliminar, verificada] Manual de Usuario [preeliminar, verificado] Solicitud de Cambio [propuesta]	Pri1.Eliminar los desperdicios
LT	IS.2.7 Incorporar la Especificación de Requerimientos y el *Manual de Usuario a la Configuración de Software en la línea base. *(opcional)	Especificación de Requerimientos [validado] *Manual de Usuario [preeliminar, verificado]	Configuración de software Especificación de Requerimientos [validada, en línea base] *Manual de Usuario del Software [preeliminar,	Pri1.Eliminar los desperdicios

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
			verificado, en línea base]	
LT AN DIS	IS.3.1 Asignar tareas a los miembros del Equipo de Trabajo de acuerdo a su rol para el Plan del Proyecto actual. **Basado en la experiencia, el equipo auto administrado encontrar como dividirse las tareas así como establecer el ritmo y asignación de tareas.*	Plan del Proyecto		Pra9. Sistemas Pull Pra9.1 Just in Time Pra12. Autodeterminación (Work-Out)
AN DIS	IS.3.2 Comprender la Especificación de Requerimientos. **Se recomienda la participación del cliente para mejorar probabilidad de asimilación de la información proporcionada.*	Especificación de Requerimientos [validado, en línea base]		Pra2. Retroalimentación. Pra2.1 Retroalimentación extendida
AN DIS	IS.3.3 Documentar o actualizar el Diseño de Software. Analizar la Especificación de Requerimientos para generar el diseño arquitectónico, su conformación en subsistemas y componentes de software, definir interfaces internas y externas. Describir a detalle, la apariencia y el comportamiento de la interfaz, con base en la Especificación de Requerimientos de tal forma que los recursos para su implantación puedan preverse. Proporcionar el detalle de los componentes de software y sus interfaces para permitir la construcción en una forma clara. Generar o actualizar el Registro de	Especificación de Requerimientos [validado, en línea base]	Diseño de Software Registro de Rastreo	Pra5. Desarrollo basado en conjuntos Pra6. Pensando en opciones Pra7. Responder hasta el último momento Pra8. Tomar decisiones Pra16. Refactorización

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	Rastreo. **Se recomienda tener varias consideraciones al realizar el diseño del software, pensar en varias opciones para posibles soluciones, procurar atrasar las decisiones cruciales de diseño hasta el momento en que la solución parezca casi natural y contemos con la mayor cantidad de información disponible del entorno del negocio del cliente. Se necesita también tener en mente el reuso de componentes y refactorización.*			
AN DIS	IS.3.4 Verificar y obtener la aprobación del Diseño de Software. Verificar que la documentación del Diseño de Software sea correcta, viable y consistente con la Especificación de Requerimientos. Verificar que el Registro de Rastreo contenga las relaciones adecuadas entre los requerimientos y los elementos del Diseño del Software. Los resultados encontrados son documentados en una Lista de Verificación y las correcciones se realizan hasta que el documento ha sido aprobado por el DIS. Si fueran necesarios cambios significativos, se propone una Solicitud de Cambio. **Todo cambio debe ser difundido tan pronto sea aprobado.*	Diseño del Software Registro de Rastreo Especificación de Requerimientos [validado, en línea base]	Lista de Verificación Diseño de Software [verificado] Registro de Rastreo [verificado] Solicitud de Cambio [propuesta]	Pra4. Sincronización.
DIS	IS.3.5 Establecer o actualizar los Casos y Procedimientos de Prueba para pruebas	Especificación de Requerimientos [validado, en línea	Casos y Procedimientos de Prueba	Pra2. Retroalimentación. Pra2.1 Retroalimentación

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	de integración basados en la Especificación de Requerimientos y el Diseño de Software. El cliente provee datos de prueba, en caso de ser necesarios. **Toda información resultante debe ser difundida tan pronto sea posible. Recordar que el propósito de las pruebas no es encontrar errores.*	base] Diseño de Software [verificado, en línea base]		extendida Pra4. Sincronización. Pra17. Pruebas.
DIS AN	IS.3.6 Verificar y obtener la aprobación de los Casos y Procedimientos de Prueba. Verificar la consistencia entre la Especificación de Requerimientos, Diseño de Software y los Casos y Procedimientos de Prueba. Los resultados encontrados se documentan en Listas de Verificación y las correcciones son realizadas hasta que el documento es aprobado por el AN. **Toda información resultante debe ser difundida tan pronto sea posible. Recordar que el propósito de las pruebas no es encontrar errores.*	Casos y Procedimientos de prueba. Especificación de Requerimientos [validado, en línea base] Diseño de Software [verificado, en línea base]	Listas de Verificación. Casos y Procedimientos de Prueba [verificado]	Pra4. Sincronización. Pra17. Pruebas.
DIS	IS.3.7 Actualizar el Registro de Rastreo incorporando los Casos y Procedimientos de Prueba **Toda información resultante debe ser difundida tan pronto sea posible. Recordar que el propósito de las pruebas*	Casos y Procedimientos de Prueba [verificados] Registro de Rastreo [actualizado]	Registro de Rastreo [actualizado]	Pra4. Sincronización. Pra17. Pruebas.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	no es encontrar errores. .			
LT	IS.3.8 Incorporar el Diseño de Software, y el Registro de Rastreo a la Configuración de Software como parte de la línea base. Incorporar los Casos y procedimientos de Prueba al Repositorio del Proyecto. **Toda información resultante debe ser difundida tan pronto sea posible.*	Diseño de Software [verificado] Casos y Procedimientos de Prueba [verificados] Registro de Rastreo [verificado]	Configuración de Software Diseño de Software [verificado, en línea base] Casos y Procedimientos de Prueba [verificados] Registro de Rastreo [verificado, en línea base]	Pra4. Sincronización.
LT PR	IS.4.1 Asignar tareas a los miembros del Equipo de Trabajo en relación a su rol, de acuerdo al Plan del Proyecto actual **Basándose en la experiencia, se recomienda que el equipo auto administrado debe encontrar como dividirse las tareas así como establecer el ritmo y asignación de tareas.*	Plan del Proyecto		Pra9. Sistemas Pull Pra9.1 Just in Time Pra12. Autodeterminación (Work-Out)
PR	IS.4.2 Entender el Diseño de Software **Para la comprensión de un sistema complejo no hay mejor conocimiento que le obtenido mediante la experiencia.*	Diseño de Software [verificado, en línea base]		Pra15. Experiencia
PR	IS.4.3 Construir o actualizar los Componentes de Software basados en la parte detallada del Diseño de Software	Diseño de Software [verificado, acorde al plan inicial] Registro de Rastreo [verificado, acorde al	Componentes de Software	Pra4. Sincronización. Pra16. Refactorización Pra1. Value Stream Mapping

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	**Se deben tomar en cuenta el re uso y la refactorización así como las dependencias existentes entre los componentes.*	plan inicial]		
PR	IS.4.4 Diseñar o actualizar los casos de pruebas unitarias y aplicarlos para verificar que los Componentes de Software implementan la parte detallada de Diseño de Software. **Toda información resultante debe ser difundida tan pronto sea posible. Recordar que el propósito de las pruebas no es encontrar errores.*	Componentes de Software	Componentes de Software [probados]	Pra4. Sincronización. Pra17. Pruebas.
PR	IS.4.5 Corregir los defectos encontrados hasta lograr una prueba exitosa. *Se recomienda recordar la priorización de los defectos y si estos se pueden llevar a otra iteración y pues no caer en ciclos de prueba y error.*	Componentes de Software [probados]	Componentes de Software [corregidos]	Pra3. Iteraciones 17. Pruebas.
PR	IS.4.6 Actualizar el Registro de Rastreo incorporando Componentes de Software construidos o modificados. **Toda información resultante debe ser difundida tan pronto sea posible.*	Componentes de Software [corregidos] Registro de Rastreo [verificado, acorde al plan inicial]	Registro de Rastreo [actualizado]	Pra4. Sincronización.
LT	IS. 4.7 Incorporar Componentes de Software y Registro de Rastreo para la Configuración de Software como parte del plan inicial. **Toda información resultante debe ser*	Componentes de Software [corregidos] Registro de Rastreo [actualizado]	Configuración de Software Componentes de Software [actualizados, en línea base] Registro de Rastreo	Pra4. Sincronización.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	difundida tan pronto sea posible.		[actualizado, en línea base]	
LT PR	IS.5.1 Asignar tareas a los miembros del Equipo de Trabajo en relación a su rol, de acuerdo al Plan del Proyecto actual *Basado en la experiencia, se recomienda que el equipo auto administrado debe encontrar la manera de como dividirse las tareas así como establecer el ritmo y asignación de tareas.*	Plan del Proyecto		Pra9. Sistemas Pull Pra9.1 Just in Time Pta12. Autodeterminación (Work-Out)
PR	IS.5.2 Entender los Casos y Procedimientos de Prueba. Establecer o actualizar el ambiente de prueba. **Para la comprensión de un sistema complejo no hay mejor conocimiento que le obtenido mediante la experiencia.*	Casos y Procedimientos de Prueba [verificados]		Pra15. Experiencia Pra17. Pruebas.
PR	IS.5.3 Integrar el Software usando los Componentes de Software y actualizar los Casos y Procedimientos de Prueba para las pruebas de integración, conforme sea necesarias. **Se recomienda recordar que el propósito de las pruebas no es encontrar errores.* *Toda información resultante debe ser difundida tan pronto sea posible.*	Componentes de Software [corregidos, en línea base] Casos y Procedimientos de Prueba [verificados] Registro de Rastreo [actualizado, en línea base]	Software Casos y Procedimientos de Prueba	Pra4. Sincronización. Pra17. Pruebas.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
PR CL	IS.5.4 Realizar pruebas de Software usando Casos y Procedimientos de Prueba para la integración y documentar los resultados en el Reporte de Pruebas. **Se recomienda recordar que el propósito de las pruebas no es encontrar errores.* *Toda información resultante debe ser difundida tan pronto sea posible.*	Software Casos y Procedimientos de Prueba	Software [probado] Reporte de Pruebas	Pra4. Sincronización. Pra17. Pruebas.
PR	IS.5.5 Corregir los defectos encontrados y realizar una prueba de regresión hasta satisfacer el criterio de término. **Se recomienda recordar que se debe hacer la priorización de los defectos y si estos se pueden llevar a otra iteración y pues no caer en ciclos de prueba y error.*	Software [probado], Reporte de Pruebas Casos y Procedimientos de Prueba Registro de Rastreo [actualizado, en línea base]	Software [corregido] Reporte de Pruebas [defectos eliminados]	Pra3. Iteraciones Pra17. Pruebas.
PR	IS.5.6 Actualizar el Registro de Rastreo en caso de ser necesario. **Toda información resultante debe ser difundida tan pronto sea posible.*	Software [corregido] Registro de Rastreo [actualizado, en línea base]	Registro de Rastreo [actualizado]	Pra4. Sincronización.
PR	IS.5.7 Documentar el Manual de Operación o actualizar el manual actual, en caso de ser apropiado. **Siempre es buena idea dejar un registro que deje testimonio claro y conciso de*	Software [probado]	Manual de Operación	Pra4. Sincronización.

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	las particularidades del sistema, que sirva de experiencia.			
PR DIS	IS.5.8 Verificar y obtener la aprobación del Manual de Operación, en caso de ser necesario (vea IS.5.7) Verificar la consistencia del Manual de Operación con el Software. Los resultados encontrados son documentados en las Listas de Verificación y se realizan las correcciones hasta que el documento sea aprobado por DES. **Toda información resultante debe ser difundida tan pronto sea posible.*	Manual de Operación [probado] Software [probado]	Lista de Verificación Manual de Operación [verificado]	Pra4. Sincronización.
AN	IS.5.9 Documentar el Manual de Usuario o actualizar el actual, si aplica. **Toda información resultante debe ser difundida tan pronto sea posible.*	Software [probado] Manual de Usuario [preliminar] (opcional)	Manual de Usuario	Pra4. Sincronización.
AN CL	IS.5.10 Verificar y obtener la aprobación del Manual de Usuario, en caso de ser necesario (vea IS.5.9). Verificar la consistencia del Manual de Usuario con el Software. Los resultados encontrados son documentados en Listas de Verificación y se realizan las correcciones hasta que el documento sea aprobado por CL. **Se recomienda que para mejorar la calidad y tiempo usado en este tipo de solicitudes de aprobación no hay mejor*	Manual de Usuario [probado] Software [probado]	Listas de Verificación Manual de Usuario [verificado]	Pra2. Retroalimentación. Pra2.1 Retroalimentación extendida

Rol	Lista de tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
	forma que teniendo presente al cliente en el sitio de desarrollo o si no al menos a un representante del cliente el cual pueda llevar la información a la persona responsable de la aprobación.			
LT	IS.5.11 Incorporar los Casos y Procedimientos de Prueba, Software, Registro de Rastreo, Reporte de Pruebas, Manual de Operación y Manual de Usuario a la Configuración de Software como parte de la línea base. **Toda información resultante debe ser difundida tan pronto sea posible.*	Casos y Procedimientos de Prueba Software [probado] Reporte de Pruebas Registro de Rastreo [actualizado] Manual de Operación [verificado] Manual de Usuario [verificado]	Configuración de Software Casos y Procedimientos de Prueba [en línea base] Software [probado, en línea base] Registro de Rastreo [actualizado, en línea base] Reporte de Pruebas [en línea base] Manual de Operación [verificado, en línea base] Manual de Usuario [verificado, en línea base]	Pra4. Sincronización.

Roles	Lista de Tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
LT ET	IS.6.1 Asignar tareas a los miembros del equipo de trabajo relacionadas con su rol, de acuerdo al *Plan del Proyecto* actual. **Basado en la experiencia, el equipo auto administrado encontrar como dividirse las tareas así como establecer el ritmo y asignación de tareas.*	*Plan del Proyecto*		Pra9. Sistemas Pull Pra9.1 Just in Time Pra12. Autodeterminación (Work-Out)
DIS	IS.6.2 Comprender la *Configuración de Software.* **Para la comprensión de un sistema complejo no hay mejor conocimiento que le obtenido mediante la experiencia.*	*Configuración de Software*		Pra15. Experiencia
DIS	IS.6.3 Documentar el *Manual de Mantenimiento* o actualizar el actual. **Toda información resultante debe ser difundida tan pronto sea posible.*	*Configuración de Software*	*Manual de Mantenimiento*	Pra4. Sincronización.

Roles	Lista de Tareas	Productos de Entrada	Productos de Salida	Practicas y principios Lean sugeridos
DIS LT	IS.6.4 Verificar y obtener la aprobación del *Manual de Mantenimiento*. Verificar la consistencia del *Manual de Mantenimiento* con la *Configuración de Software*. Los resultados encontrados son documentados en *Listas de Verificación* y las correcciones se realizan hasta que el documento sea aprobado por LT. **se recomienda que el LT se apoye en el ET.* *Toda información resultante debe ser difundida tan pronto sea posible.*	*Manual de Mantenimiento* *Configuración de Software*	*Listas de Verificación* *Manual de Mantenimiento [verificado]*	Pra4. Sincronización. 2.2.1.1 Pri6.Respetar a las personas
LT	IS.6.5 Incorporar el *Manual de Mantenimiento* a la línea base de la *Configuración de Software.* **Toda información resultante debe ser difundida tan pronto sea posible.*	*Configuración de Software* *Manual de Mantenimiento [verificado]*	*Configuración de Software* *Manual de Mantenimiento [verificado, en línea base]*	Pra4. Sincronización.
LT	IS.6.6 Llevar a cabo la entrega de acuerdo a las *Instrucciones de Entrega.* **Todo lo definido como acuerdo entre el cliente y los demás involucrados.*	*Plan del Proyecto* *- Instrucciones de Entrega* *Configuración de Software*	*Configuración de Software [liberado]*	Pra19. Contratos

2.3 Roles

*Rol agregado por el LSD

	Rol	Abreviatura	Competencias
1.	Administrador del Proyecto	AP	Capacidad de liderazgo con experiencia para toma decisiones, planeación, administración de personal, delegación y supervisión, conocimiento de finanzas y desarrollo de software.
2.	Analista	AN	Conocimiento y experiencia que permita deducir, especificar y analizar los requerimientos. Conocimiento en diseño de interfaces de usuario y criterios ergonómicos. Conocimiento de técnicas de revisión. Conocimiento de técnicas de edición. Experiencia en desarrollo y mantenimiento de software.
3.	Cliente	CL	Conocimiento de los procesos del Cliente y habilidad para explicar los requerimientos del Cliente. El Cliente (representante del Cliente) debe tener autoridad para aprobar los requerimientos y sus cambios. El Cliente incluye representantes de los usuarios con la finalidad de asegurar que el ambiente de operación sea dirigido de forma correcta. Conocimiento y experiencia en el dominio de la aplicación.
4.	Diseñador	DIS	Conocimiento y experiencia en los componentes y diseño de arquitectura. Conocimiento de técnicas de revisión. Conocimiento y experiencia en la planeación y ejecución de pruebas de integración. Conocimiento de técnicas de edición. Experiencia en desarrollo y mantenimiento de software.

5.	Equipo de Trabajo	ET	Conocimiento y experiencia de acuerdo a sus roles dentro del proyecto: LT, AN, DIS,LS, y/o PR. Conocimiento de los estándares usados por el Cliente y/o por la OP.
6.	Líder Técnico	LT	Conocimiento y experiencia en el dominio del proceso de software.
7.	Programador	PR	Conocimiento y/o experiencia en programación, integración y pruebas unitarias. Conocimiento de técnicas de revisión. Conocimiento de técnicas de edición. Experiencia en desarrollo y mantenimiento de software.
8.	Líder Social*	LS	Capacidad de socialización. Atención a los detalles tales como el estado anímico del equipo de trabajo.

3 Descripción de Productos

*NC. No hay comentarios.

	Nombre	Descripción	Visión Lean
1.	*Casos y Procedimientos de Prueba*	Elementos necesarios para probar el código. Los Casos de prueba pueden incluir: - Identificación del caso de prueba - Elementos a probar - Especificaciones de entrada - Especificaciones de salida - Ambiente requerido - Requerimientos de procedimientos especiales - Dependencias de interfaz Los Procedimientos de Prueba pueden incluir: - Identificación de: nombre de la	Son definidos como tal para mantener un acuerdo en como se realizaran las pruebas. Las pruebas dependen del miembro del equipo que genera el código. Como mínimo necesario se prueba la funcionalidad en condiciones en las que se espera opere el código, es decir, entradas y salidas y ambiente de operación. El propósito de las pruebas es comprobar que el código hace lo que se espera, no es encontrar

	Nombre	Descripción	Visión Lean
		prueba, descripción de la prueba y la fecha de terminación de prueba - Identificación de posibles problemas de aplicación - Nombre de la persona que realiza el procedimiento de prueba - Identificación de los requerimientos previos - Identificación de los pasos del procedimiento, incluyendo el número de paso, la acción requerida por el probador y los resultados esperados Los estados aplicables son: verificado y en línea base.	defectos.
2.	*Componente de Software*	Conjunto de unidades de código relacionadas. Los estados aplicables son: prueba unitaria, corregido y en línea base.	Un componente de software es un conjunto de líneas de cogido escritas para realizar una tarea.
3.	*Configuración de Software*	Un conjunto de productos de software identificados de forma única y consistentes, incluyendo: - Especificación de Requerimientos - Diseño de Software - Registro de Rastreo - Componentes de Software - Software - Casos y Procedimientos de Prueba - Reporte de Pruebas / Resultados de la Pruebas - Manual de Operación - Manual de Usuario - Manual de Mantenimiento Los estados aplicables son: entregado y aceptado.	El propósito de la Configuración del Software es establecer y mantener la integridad de los productos de software a través del ciclo de vida del proceso de software. En Lean, se establece un control auto impulsado en el equipo de trabajo. Hay varias técnicas usadas en los desarrollos de software ágil, así que hay varias opciones de donde escoger, lo mínimo necesario a controlar es : • Entregas al cliente. Al final de cada iteración se muestra al cliente el trabajo hecho. • Documentos generados. Se recomienda solo generar documentos breves y estrictamente necesarios y estos deben ser de alto

	Nombre	Descripción	Visión Lean
			nivel. Código. Todo código generado para hacer posible el entregable debe ser mantenido en un repositorio personal y compartido, la comunicación es primordial en Lean así como el hacer posible que todos los miembros del equipo tengan lo que necesiten para el desarrollo (Herramientas, equipos y recursos) estos deben estar cerca y disponibles.
4.	*Diseño de Software*	Información textual y gráfica de la estructura del software. Esta estructura puede incluir las siguientes partes: Diseño Arquitectónico del Software (alto nivel) - Describe la estructura global del *Software*: - Identifica los *Componentes de Software* requeridos - Identifica las relaciones entre los *Componentes de Software* - Consideraciones requeridas: - características de desempeño de software - hardware, software e interfaces humanas - características de seguridad - requerimientos de diseño de base de datos - manejo de errores y atributos de recuperación Diseño Detallado del Software (bajo nivel) - incluye detalles de los *Componentes de Software* para facilitar su construcción y prueba dentro del ambiente de desarrollo; - Proporciona detalle del diseño (puede ser representado como un prototipo, diagrama de flujo, diagrama entidad-relación, pseudo código, etc.)	Descomposición del producto a desarrollar en partes mínimas necesarias para iniciar, continuar o terminar el producto. El diseño es acordado por el equipo de trabajo. Se pueden hacer varios diseños para que en algún punto el mejor diseño sea el elegido y con base en los diseños competidores tomar las mejores ideas de todos y crear un diseño hibrido, cabe mencionar que una de las máximas a respetar en los diseños es la flexibilidad, el reuso, el bajo acoplamiento y la alta cohesión de los componentes de la arquitectura.

	Nombre	Descripción	Visión Lean
		- Proporciona el formato de entrada / salida de los datos - Proporciona la especificaciones del almacenamiento de los datos - Establece convenciones de nombrado de los datos - Define el formato de las estructuras de datos - Define los campos de datos y el propósito de cada elemento de los datos - Proporciona las especificaciones de la estructura del programa Los estados aplicables son: verificado y en línea base.	
5.	*Documento de Aceptación*	Documentación de la aceptación de los entregables del proyecto por parte del cliente. Puede tener las siguientes características: - Registro de la recepción de la entrega. - Identificación de la fecha de recepción. - Identificación de los elementos entregados - Registro de la verificación de los criterios de aceptación definidos por parte del cliente - Identificación de cualquier asunto pendiente (en caso de ser aplicable) - Firma de recibido por parte del cliente	Puede ser generado en papel o en digital o solo con la aceptación de manera verbal del cliente o encargado del cliente que este con el equipo de desarrollo. Si es verbal se recomienda ser grabado en voz o en video y guardado en el repositorio del proyecto

6.	*Enunciado de Trabajo*	Descripción del trabajo a realizar en relación al desarrollo de software. Puede incluir:
		
		- *Descripción del producto*
		  o Propósito
		  o Requerimientos generales del cliente
			El enunciado de trabajo describe de manera sencilla el propósito del desarrollo de software para este proyecto.

	Nombre	Descripción	Visión Lean
		- *Alcance,* que describa que sí incluye y qué no. - *Objetivos* del proyecto - *Entregables,* lista de productos a entregar al cliente Los estados aplicables son: revisado.	

7.	*Especificación de Requerimientos*	Identifica los requerimientos de software.
		
		Puede contener las siguientes características:
		
		- Introducción – descripción general del software y dentro del ámbito de negocio del cliente;
		
		- Descripción de requerimientos:
		  - Funcionalidad - necesidades que debe satisfacer el software cuando se usa en condiciones específicas. La funcionalidad debe ser adecuada, precisa y segura;
		
		  - Interfaz de usuario - definición de las características de la interfaz de usuario que permitan comprender y aprender el uso de software fácilmente para que el usuario sea capaz de realizar sus tareas eficientemente. Incluye la descripción del modelo de interfaz;
		
		  - Interfaces externas - definición de las interfaces con otro software o hardware;
		
		  - Fiabilidad - especificación del nivel de ejecución de software referente a su madurez, tolerancia a fallas y recuperación;
		
		  - Eficiencia - especificación del
			Describe los requerimientos necesarios para comenzar el desarrollo que desde un inicio deben ser difundidos a todos los involucrados en el proyecto. Dentro de Lean, estos requerimientos pueden ser cambiados o corregidos a lo largo del ciclo de desarrollo. Todo cambio debe ser aceptado y entendido por los involucrados en el proyecto.

	Nombre	Descripción	Visión Lean
		nivel de ejecución del software en relación con el tiempo y el uso de los recursos; - Mantenimiento - descripción de los elementos para facilitar la comprensión y ejecución de futuras modificaciones del software. - Portabilidad - descripción de las características de software que permiten transferirlo de un lugar a otro; - Diseño y construcción de las limitaciones/restricciones - necesidades impuestas por el cliente; - Interoperabilidad - Capacidad para usar o intercambiar información entre dos o más sistemas o componentes de software. - Reutilización – característica de cualquier producto o subproducto o de alguna de sus partes, para ser utilizado por varios usuarios como un producto final, en el desarrollo o ejecución de otros productos de software. - Legales y regulativos - necesidades impuestas por las leyes, regulaciones, etc. Cada requerimiento es identificado, único y es verificable o susceptible de evaluación. Los estados aplicables son: verificado, validado y en línea base.	

	Nombre	Descripción	Visión Lean
8.	*Listas de Validación*	Documenta la ejecución de la validación. Puede incluir el registro de: - Participantes - Fecha - Lugar - Duración - Lista de verificación - Elementos aprobados. - Elementos no aprobados. - Elementos pendientes de validar - Defectos identificados durante la validación	NC
9.	*Listas de Verificación*	Documenta la ejecución de la verificación. Puede incluir el registro de: - Participantes - Fecha - Lugar - Duración - Lista de verificación - Elementos aprobados. - Elementos no aprobados. - Elementos pendientes de verificar - Defectos identificados durante la verificación	No como tal de manera formal pero es una lista acordada entre el cliente y el equipo de desarrollo la cual expone los elementos entregados al cliente y si este los acepta, los rechaza o sugiere algún cambio. De manera verbal o escrita, se requiere la intervención de todo el equipo de desarrollo y el cliente.
10.	*Manual de Mantenimiento*	Describe la *Configuración de Software* y el entorno utilizado para el desarrollo y realización de las pruebas (compiladores, herramientas de diseño, construcción y pruebas). Puede tener las siguientes características: - Incluye o se refiere a todos los elementos de la *Configuración de Software* desarrollados durante la implementación - Identifica el ambiente utilizado para el desarrollo y realización de pruebas (compiladores, herramientas de diseño, construcción y pruebas). Está escrito en términos que el personal	NC

	Nombre	Descripción	Visión Lean
		que dará mantenimiento al software puede entender. Los estados aplicables son: verificado y en línea base.	

11.	*Manual de Operación*	Contiene la información necesaria para instalar y administrar el software. Puede tener las siguientes características:
		
		- Criterios para la operación.
		- Una descripción de cómo operar el producto, incluyendo:
		  – ambiente operativo requerido
		  – herramientas de apoyo y material requerido (por ejemplo, manuales de usuario)
		  – posibles alertas de seguridad
		  – preparativos para la puesta en marcha y secuencia de pasos
		  – preguntas frecuentes (FAQ)
		  – fuentes de información adicional y ayuda para operar el producto
		- Certificación y aprobaciones de seguridad
		- Garantía e instrucciones de reemplazo
		- Debe estar escrito en términos que el personal responsable de la operación pueda entender.
		
		Los estados aplicables son: verificado y en línea base.
			NC

12.	*Manual de Usuario*	Describe la forma de uso del software basado en las interfaces de usuario. Puede	NC

	Nombre	Descripción	Visión Lean
		tener las siguientes características: - Procedimientos del usuario para realizar tareas específicas utilizando el *Software.* - Procedimientos de instalación y desinstalación. - Breve descripción del uso previsto del *Software* (el concepto de operaciones). - Los recursos provistos y requeridos. - Entorno operacional requerido. - Facilidad para reportar problemas y asistencia. - Procedimientos para acceder y salir del *Software*. - Relación y explicación de los comandos del *Software* y de los mensajes del sistema hacia el usuario. - Advertencias, precauciones y notas para identificar riesgos y su forma de corrección. - Incluye procedimientos para solución de problemas y corrección de errores. Está escrito en términos comprensibles para los usuarios. Los estados aplicables son: preliminar, verificado y en línea base.	
13.	*Minuta*	Registro de los acuerdos establecidos con el cliente y/o el equipo de trabajo. Puede tener la siguiente información: - Propósito de la reunión - Asistentes - Fecha y lugar - Referencia a minutas previas - Qué se logró - Identifica cuestiones planteadas - Cualquier asunto abierto - Acuerdos - Próxima reunión (en caso	Se sugiere que sea de manera verbal con un registro ya sea de audio o video y guardada en el repositorio.

	Nombre	Descripción	Visión Lean
		necesario) El estado aplicable es: actualizado.	

14.	*Plan del Proyecto*	Presenta cómo serán ejecutados los procesos y actividades del proyecto para asegurar su conclusión exitosa, así como la calidad de los productos entregables. Puede incluir los siguientes elementos y características:
		
		- *Descripción de producto*
		  o Propósito
		  o Requerimientos generales del cliente
		- Descripción de *alcance* respecto a lo incluido y lo no incluido.
		- *Objetivos* del proyecto
		- *Entregables* – lista de productos a entregar al cliente.
		- *Tareas,* incluyendo verificaciones, validaciones y revisiones con el cliente y equipo de trabajo que permitan asegurar la calidad de los productos de trabajo. Las tareas pueden ser representadas como una Estructura de Desglose de Trabajo (EDT).
		- *Duración estimada de las tareas*
		- *Recursos* (humanos, materiales, estándares, equipos y herramientas), incluyendo la capacitación necesaria. Incluye la identificación y programación de los recursos.
		- *Composición del equipo de trabajo.*
		- *Calendario de las tareas del proyecto*, indicando la fecha de inicio y fecha de finalización previstas para cada tarea, y *las relaciones y dependencias entre ellas.*
		- *Esfuerzo y el costo estimado*
		- *Identificación de los riesgos del proyecto*
		- *Estrategia para el control de versiones*
			Una descripción general de los elementos necesarios para completar el desarrollo, recordar que este debe ser sumamente flexible, abierto al cambio ya que será modificado según se avance en el proyecto de acuerdo a las necesidades del cliente y el aprendizaje adquirido.

	Nombre	Descripción	Visión Lean
		- Herramientas de repositorio o mecanismos identificados - Localización y mecanismos de acceso para el repositorio especificado - Identificación y control de versiones definidos. - Respaldo y mecanismos de recuperación definidos - Mecanismos de almacenamiento, manipulación y entrega especificados (incluyendo archivo y recuperación). - *Instrucciones de entrega* - Elementos requeridos para la liberación del producto (por ejemplo, hardware, software, documentación, etc.) - Requerimientos de entrega - Tareas a realizar en orden secuencial - Liberaciones aplicables identificadas - Información de la versión de todos los componentes de software entregados - Procedimientos de respaldo y recuperación necesarios. Los estados aplicables son: verificado, aceptado, actualizado y revisado.	
15.	*Registro de Defectos*	Identifica las actividades establecidas para corregir una desviación o un problema relativo al cumplimiento de un plan. Puede contener la siguiente información: - el problema inicial - una solución - las acciones correctivas a tomar - responsable de la conclusión de las acciones definidas - fecha de apertura y fecha de cierre esperada	Documento informal generado en las pruebas, guarda la información relacionada a algún defecto si este es encontrado, puede ser almacenada en digital en el repositorio con acceso para todos los miembros del equipo.

	Nombre	Descripción	Visión Lean
		- un indicador de estado - acciones de seguimiento	
16.	*Registro de Rastreo*	Documenta la relación entre los requerimientos incluidos en la *Especificación de Requerimientos*, los elementos del *Diseño de Software*, los *Componentes de Software*, los *Casos y los Procedimientos de Prueba*. Puede incluir: - Especificación de los requerimientos por rastrear. - Proporciona el mapeo (hacia adelante y hacia atrás) de los requerimientos a los elementos del *Diseño de Software*, los *Componentes de Software*, los *Casos de Prueba y los Procedimientos de Prueba*. Los estados utilizados son: verificado, en línea base y actualizado.	Se recomienda sea conciso y de acceso a todos los miembros de equipo, ya que esta información puede servir para la distribución de tareas.
17.	*Reporte de Avance*	Registra el estado del proyecto contra el *Plan del Proyecto.* Puede tener la siguiente información: - Estado real de las tareas contra las tareas planeadas - Estado de los resultados reales contra los objetivos/metas establecidos - Estado de los recursos asignados reales contra los recursos planeados - Estado de los costos reales contra los presupuestos estimados - Estado calendario real contra el calendario planeado - Estado de los riesgos actuales con respecto a los identificados previamente - Registro de cualquier desviación de las tareas planeadas y su causa	Exposición de las actividades del equipo, se recomienda sea de manera oral. Con opción a retroalimentación por parte del cliente.

	Nombre	Descripción	Visión Lean
		El estado aplicable es: evaluado.	
18.	*Reporte de pruebas*	Documenta la ejecución pruebas. Puede incluir: - Un resumen de cada defecto. - Identificación del caso de prueba en cuestión. - Nombre del probador que encontró cada defecto. - Severidad de cada defecto. - Identificación de la(s) función(es) afectada(s) por cada defecto. - Fecha en que cada defecto fue originado. - Fecha en que cada defecto fue resuelto. - Nombre de la persona quién resolvió cada defecto. El estado aplicable es: en línea base.	Documento breve y conciso que esta a disposición de todos los miembros del equipo.
19.	*Repositorio del Proyecto*	Contenedor electrónico para almacenar los productos de trabajo y entregables del proyecto. Puede tener las siguientes características: - Almacena los productos del proyecto - Almacena los productos entregables ya liberados - Capacidades de almacenamiento y recuperación - Facilidad para navegar en su contenido - Enlista los contenidos y la descripción de los atributos - Comparte y transfiere productos de trabajo entre los grupos involucrados - Controles de acceso efectivos - Mantiene la descripción de los productos de trabajo - Recuperación de versiones anteriores de los productos de trabajo - Facilidad para reportar el estado de los productos	Un repositorio personal para cada involucrado en el proyecto y un repositorio del equipo. Abierto a todos los miembros. Todo lo relacionado al proyecto debe ser guardado por que todo es importante.

	Nombre	Descripción	Visión Lean
		- Los cambios a los productos de trabajo son rastreados a la *Solicitud de Cambio*. Los estados aplicables son: recuperado y actualizado.	
20.	*Respaldo del Repositorio de Proyecto*	Repositorio usado para respaldar el *Repositorio del Proyecto*, y en caso necesario recuperar la información.	Respaldo Periódico del repositorio del proyecto.
21.	*Software*	Elemento de software (código fuente y código ejecutable) para un cliente, integrado por un conjunto de *Componentes de Software*. Los estados aplicables son: probado y en línea base.	NC
22.	*Solicitud de cambio*	Requisición de una modificación para corregir un problema o incorporar una mejora en el software o en su documentación. Puede contener la siguiente información: - propósito del cambio - estado de la solicitud - información de contacto del solicitante - Sistema(s) impactado(s) - Impacto en la operación de sistemas existentes - Impacto en la documentación asociada - Criticidad de la solicitud y fecha en que se requiere Los estados aplicables son: propuesto, evaluado y aceptado.	Petición de cambio en algún elemento del desarrollo, expresada por el cliente o por algún o algunos miembros del equipo, se exponen los cambios solicitados, usualmente presentada en alguna interacción con el cliente, se recomienda sea almacenada en el repositorio como un archivo de voz de video o documento digital.

Capítulo 4. Validación

En este capitulo describe la validación realizada por un experto en metodologías y practicas agiles.

Para validar la interpretación descrita se invitó a un experto en prácticas y metodologías agiles para que revisara el documento y emitiera su opinión. Con este propósito se diseño un grupo de preguntas que respondería el experto.

A continuación las preguntas formuladas al experto validador junto con las respuestas expresadas por el mismo.

1.- ¿Desde su punto de vista, la visión proporcionada del Lean Software Development en este trabajo es coherente?, ¿Qué agregaría, tomando en cuenta su experiencia en métodos agiles de desarrollo de software?

Respuesta: *El trabajo concentra su definición de Lean Software Development en aquella presentada por Mary y Tom Poppendieck y publicadas en el libro Lean Software Development: An Agile Toolkit (Poppendieck & Poppendieck, Lean Software Development: An Agile Toolkit, 2003), además de otros textos por los mismo autores.*

La primera observación que tendría es que desde la perspectiva de considerar los métodos ágiles de desarrollo de software en general, el LSD es solo uno de varios métodos y esquemas que existen. En el texto se concentra en LSD, pero se pierde de vista todo el contexto del movimiento Ágil que surge con el Manifiesto Ágil, y los métodos de programación extrema. Recomendaría el contextualizar más la discusión de LSD con los elementos del manifiesto ágil.

Quizás sería importante, además de contextualizar, el poder justificar el porque se seleccionó LSD como el método entre los que existen en el movimiento ágil. Hasta donde yo puedo ver, en mi práctica profesional, hay otros métodos que son más comúnmente usados: por ejemplo SCRUM. El punto aquí no es decir que no se debió seleccionar LSD, si no declarar que es poco claro por qué se seleccionó.

En general el tratamiento de LSD es bastante detallado y creo que se analiza a un buen nivel de profundidad y detalle para poder discutirlo con respecto al ISO/IEC 29110 Perfil Básico. Lo que se extraña un poco es el contar con un ejemplo detallado que sirva para ilustrar los conceptos que se presentan en la sección 2. Introducir un ejemplo ayudaría a entender mejor el método.

2.- ¿De la aplicación de las practicas mencionadas dentro del proceso de administración del ISO/IEC 29110 Perfil Básico, considera cumplen con lo propuesto en LSD? ¿Qué agregaría, tomando en cuenta su experiencia en métodos agiles de desarrollo de software?

Respuesta: *No estoy familiarizado con el ISO/IEC 29110 Perfil Básico, a nivel de poder ver si algo de ese framework se dejó de incluir. Lo que puedo ver en cuanto a LSD, pero sobre todo a métodos ágiles es que se hace un buen mapeo.*

3.- ¿De la aplicación de las prácticas mencionadas dentro del proceso de implementación del ISO/IEC 29110 Perfil Básico, considera que cumplen con lo propuesto en LSD? ¿Qué agregaría, tomando en cuenta su experiencia en métodos agiles de desarrollo de software?

Respuesta: *No estoy familiarizado con el ISO/IEC 29110 Perfil Básico, a nivel de poder ver si algo de ese framework se dejó de incluir. Lo que puedo ver en cuanto a LSD, pero sobre todo a métodos ágiles es que se hace un buen mapeo.*

De las respuestas dadas por el experto sabemos que desde su punto de vista el tratamiento realizado a LSD es bastante detallado y que el análisis realizado tiene un buen nivel de profundidad. Características que permiten hacer un acercamiento coherente y detallado al ISO/IEC 29110 Perfil Básico.

Por otra parte LSD desde el punto de vista del experto hace un buen mapeo en relación a los métodos agiles y sus principios.

El experto recomendó ahondar más en la relación del LSD con elementos del manifiesto ágil motivo por el cual se agregó la sección “2.4 LSD y el manifiesto ágil”, en donde se relacionan de manera mas explicita los principios del manifiesto ágil con los principios y practicas de LSD.

En otra parte el experto sugiere incluir un ejemplo, esto se pensó incluir en trabajos futuros.

Capítulo 5. Conclusiones

El objetivo de este trabajo fue desarrollar una propuesta de interpretación de los principios y prácticas de Lean Software Development que se puedan aplicar a los dos procesos de ISO/IEC 29110 Perfil Básico (29110-5-1-2:2011). Para esto se realizaron las siguientes actividades:

- Se hizo una recopilación de prácticas y principios de LSD.
- Clasificaron y analizaron los principios y prácticas identificados aplicables a los procesos de ISO/IEC 29110 Perfil Básico.
- Se generó una relación entre los productos y tareas descritos ISO/IEC 29110 Perfil Básico con principios y prácticas de LSD.
- Se obtuvo la validación de la interpretación por un experto en metodologías y prácticas agiles.

Mediante las cuales se generó este trabajo el cual describe una fusión entre lo descrito en la norma ISO/IEC 29110 Perfil Básico y los principios y practicas pertenecientes a una metodología ágil (LSD).

Dando así una propuesta valida que contiene practicas y principios agiles enmarcados dentro de los procesos de una norma ISO/IEC con lo que una VSE puede ser certificada y así obtener una validación de la calidad por un instituto auditor permitiéndole ventajas competitivas dentro del ambiente de desarrollo de software.

Con lo que se concluye que el objetivo de este trabajo se cumplió de manera exitosa, realizando una aportación a los profesionales del área del desarrollo de software, a la industria del ramo y a la sociedad en general.

Como trabajo a futuro de propone usar la Interpretación de ISO/IEC 29110 Perfil Básico con principios y prácticas de Lean Software Development como marco de referencia para proyectos independientes dentro de organizaciones que usen el ISO/IEC 29110. De esta

manera obtener datos concretos mediante los cuales se analizara el rendimiento y eficacia de la interpretación. Una vez obtenido el análisis final se podrán generar los ajustes necesarios para mejorar la interpretación.

También como trabajo a futuro se propone en la creación de una interpretación que busque encontrar a las prácticas y principios LSD con una norma ISO/IEC para entidades u organizaciones de mayor tamaño con el propósito de mostrar que en teoría no importa el tamaño de la organización para que sea posible aplicar practicas y principios de las metodologías agiles.

Anexo 1.

Plantilla de definición de prácticas Kuali-Beh.

[Identificador]	**Práctica**
[nombre]	
Objetivo	
[objetivo]	
Entrada	**Resultado**
[Características esperadas]	*[Características esperadas]*
Guía	
Actividades	
[Lista de actividades]	
Tareas(Opcional)	
[Que hacer… que hacer…]	
Herramientas(Opcional)	
[Lista de herramientas propuestas]	
Conocimientos y habilidades	
[Habilidades, competencias y logros]	
Criterios de verificación	
[Criterio A, criterio B]	
Métricas	
[Métrica A, métrica B]	

Anexo 2.

Principios del manifiesto ágil. (Beck, y otros, 2001)

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

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.

El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

El software funcionando es la medida principal de progreso.

Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Bibliografía

29110-5-1-2:2011, I. T. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) — Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile.* Geneva: International Organization for Standardization (ISO).

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., y otros. (17 de Febrero de 2001). *Agile manifesto.* Recuperado el 4 de Mayo de 2013, de http://agilemanifesto.org/iso/es/

Brown , W. J., Malveau, R. C., McCormick, H. W., & Mowbray, T. J. (1998). *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.* Canada: John Wiley & Sons.

Hernández Meléndrez, E. (2006). *Sitios: Portal de Salud de Cuba*. Recuperado el 25 de Septiembre de 2012, de Portal de Salud de Cuba: http://www.sld.cu/

Oktaba, H. J., Morales Trujillo, M. E., & Dávila Muñoz, M. (02 de 06 de 2012). KUALI-BEH v1.1. *KUALI-BEH: Software Proyect Common Concepts*. México, D.F., México.

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* NJ: Addison-Wesley Professional.

Poppendieck, M., & Poppendieck, T. (2006). *Implementing Lean Software Development - from Concept to Cash.* NJ: Addison-Wesley Professional.

Pressman, R. S. (2005). *Ingenieria del Software, Un Enfoque Practico.* Mc Graw Hill.

Villaseñor Contreras, A., & Galindo Cota, E. (2007). *Conceptos y reglas de Lean Manufacturing.* Mexico: Limusa S.A.