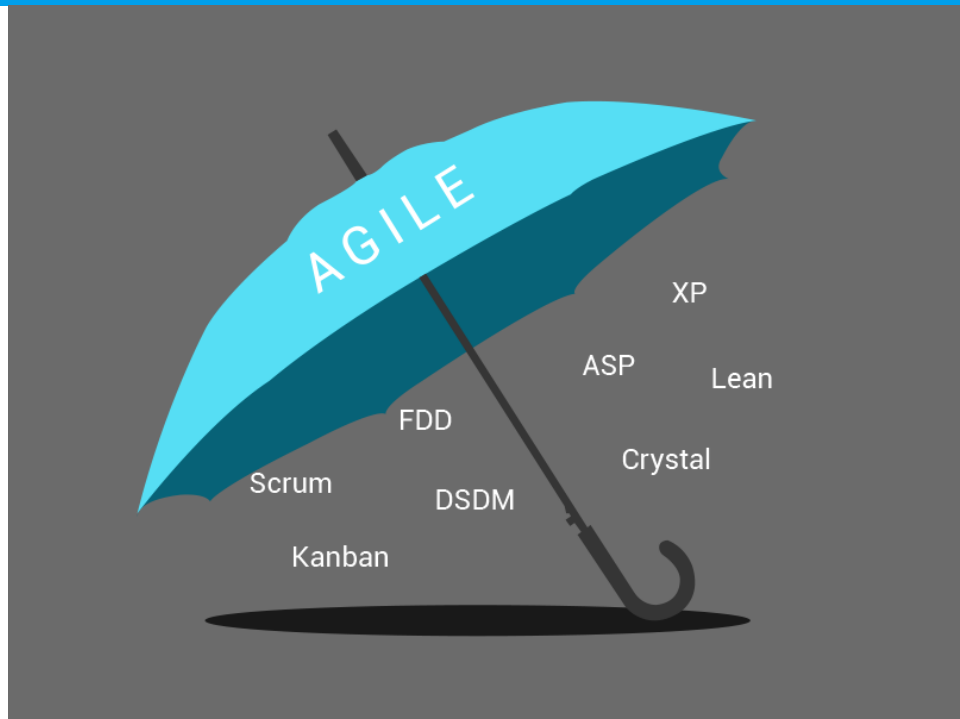
AGILIDAD



QUÉ ES LA METODOLOGÍA ÁGIL?

- Agil es una palabra que la industria de la tecnología de la información usa para describir un método alternativo de gestión de proyectos.
- El método Ágil es un proceso que permite al equipo dar respuestas rápidas e impredecibles a las valoraciones que reciben sobre su proyecto. Crea oportunidades de evaluar la dirección de un proyecto durante el ciclo de desarrollo. Los equipos evalúan el proyecto en reuniones regulares, llamadas sprints o iteraciones.
- El método ágil es un **proceso de empoderamiento** que ayuda a las empresas a diseñar y crear el producto idóneo. El proceso de gestión es muy beneficioso para las compañías de software porque les permite analizar y mejorar su producto durante el desarrollo del mismo. Esto da a las empresas la capacidad de fabricar un producto valioso, de manera que se mantengan competitivas en el mercado.

MANIFIESTO POR EL DESARROLLO ÁGIL DE SOFTWARE

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar.

VALORES AGILES

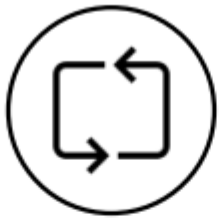
Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

12 Principios del Manifiesto Ágil

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivadas. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre como ser mas efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Satisfacer al **usuario**

Aceptamos los **cambios**



Entrega **frecuente**

Negocio e IT juntos

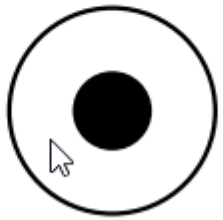
Personas motivadas

Conversación **cara a cara**

Producto **funcionando**

Ritmo **sostenible**

Excelencia técnica



Sencillez

Emergen Arq y Diseño

El equipo **reflexiona**

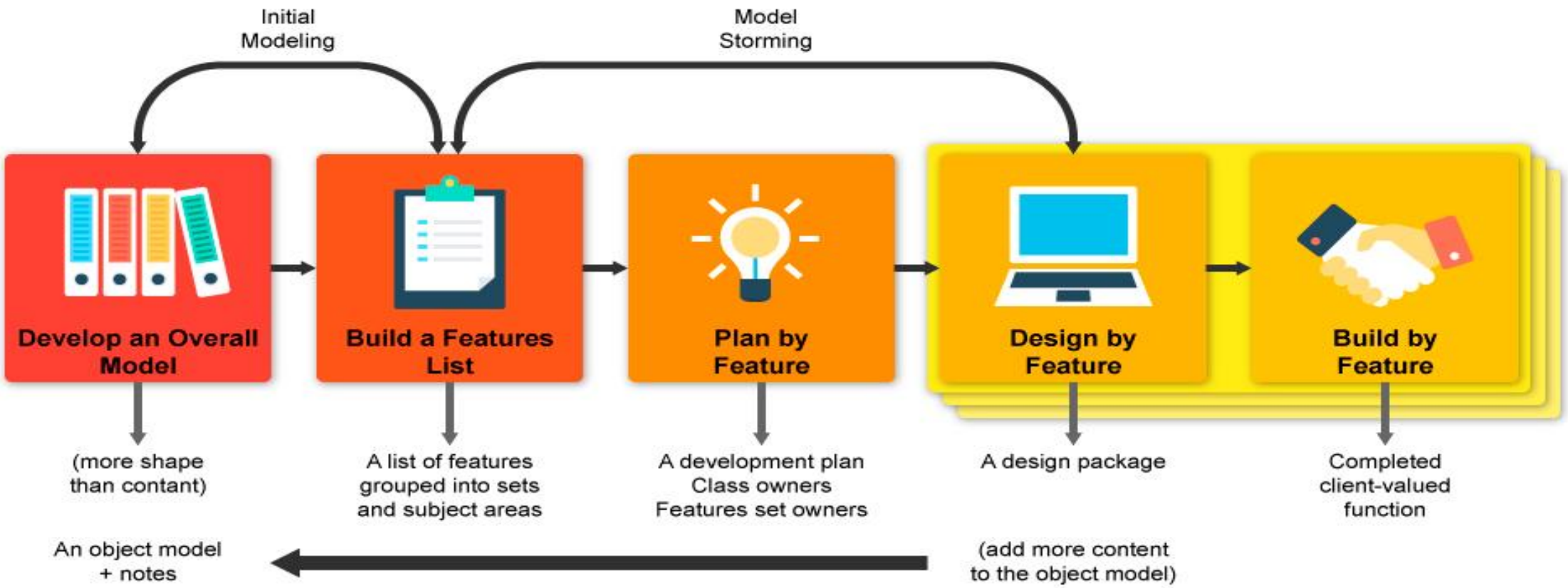
FEATURE DRIVEN DEVELOPMENT

CARACTERISTICAS

- Es una metodología ágil y adaptativo para desarrollo de sistemas.
- No cubre el proceso de desarrollo completo, se enfoca en el diseño y fases de construcción.
- Diseñado para trabajar con otras actividades de desarrollo de software.
- FDD consiste de 5 procesos secuenciales, 2 de los cuales son iterativos.
- FDD provee métodos, técnicas, y guías.

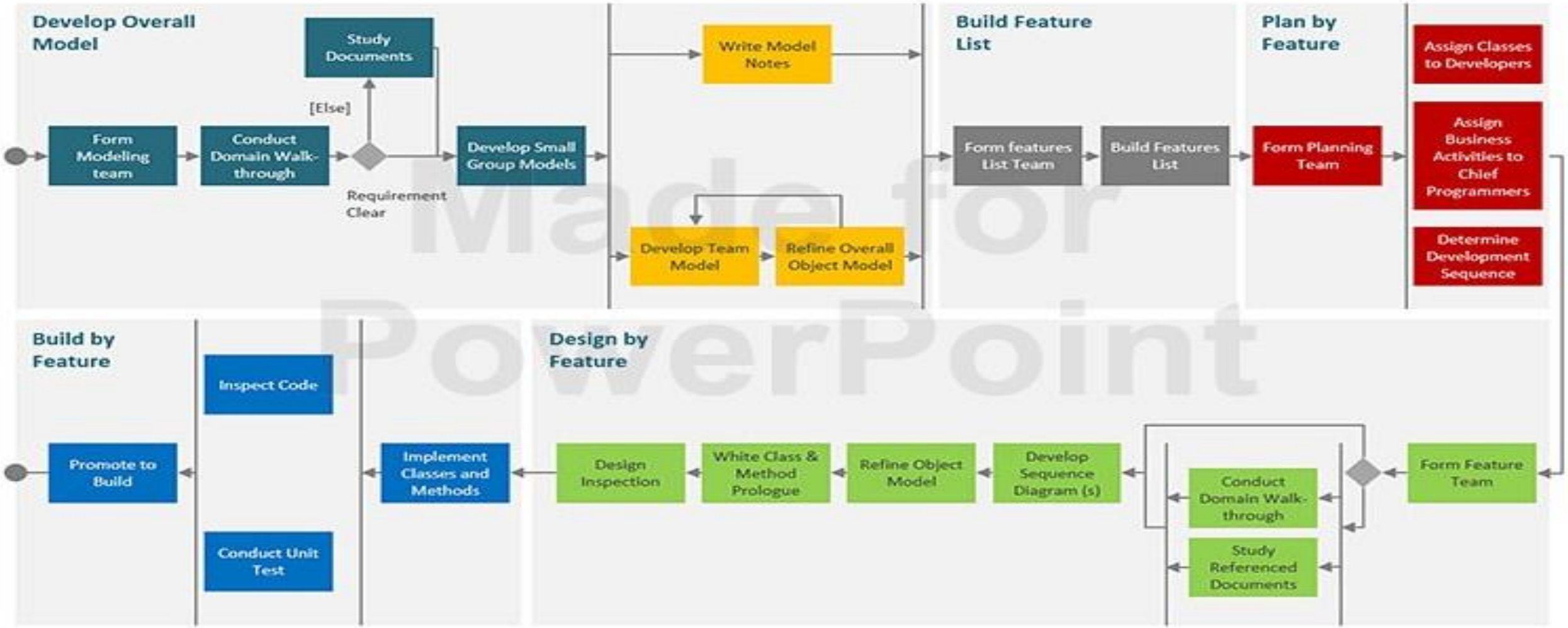
FEATURE DRIVEN DEVELOPMENT

PROCESO



DESAROLLO DE UN MODELO

GENERAL



DESARROLLO DE UN MODELO GENERAL

- Definir alcance, contexto y requerimientos.
- Presentar al jefe de arquitectos y al equipo una descripción de alto nivel del sistema.
- Se designa como Domain Expert a la persona que se hace cargo de este proceso.
- El dominio de la aplicación se divide en diferentes áreas y descripciones paso a paso con el propósito de producir Object Models (Clases) para cada área de dominio
- Simultáneamente el modelo global es construido

DESARROLLO DEL MODELO

CREAR LISTA DE FUNCIONALIDADES

- **Crear la lista de requerimientos**
- **Validar la lista de requerimientos**

UN PLAN POR FUNCIONALIDAD

- Crear un plan de alto nivel
- Los requerimientos son ordenados de acuerdo a prioridad y dependencias
- La lista de requerimientos es asignado al jefe de programadores
- Las clases identificados son asignados a cada desarrollador

DISEÑO Y CONSTRUCCION POR

FUNCIONALIDAD

- Se selecciona un conjunto de requerimientos
- Se implementa el requerimiento
- Iteración de hasta 2 semanas
- En cada iteración:
  - Inspección al diseño
  - Codificación
  - Unit Testing
  - Integración e inspección del código

ROLES

- Project Manager
- Jefe de Arquitectos – Responsable del diseño de alto nivel
- Development Manager – Dirige el desarrollo diariamente y resuelve conflictos
- Jefe de Programadores
  - Desarrollador Senior
  - Dirige equipos pequeños
  - Selecciona requerimientos

ROLES CONT.

- Class Owner – Responsable de la clase asignado
- Domain Expert
  - Puede ser un usuario, cliente, patrocinador o analista de negocios
  - Es la persona que conoce el dominio de la aplicacion y transfieren su conocimiento a los desarrolladores
- Domain Manager – Dirige a los Domain Experts
- Release Manager – Controla el progreso del proceso

ROLES CONT 2.

- Language Lawyer/Language Guru – Conocedor de un lenguaje de programacion o tecnología
- Build Engineer – Desarrollador.
- Toolsmith – Responsable para construir pequeñas herramientas de soporte para el desarrollo
- System Administrator – Responsable de la administración de servidores
- Tester
- Deployer
- Technical Writer

KANBAN

UN VISTAZO A KANBAN…

El sistema de producción de Toyota y el desarrollo de software tienen elementos en común no a nivel práctico aunque sí a nivel de principios, observando Kanban desde un punto de vista muy general, éste en realidad consiste en un elemento dentro del sistema de producción.

Lean y Kanban son conceptos nacidos en el área de manufactura por ende que un traslado directo no puede realizarse ya que en manufactura se fabrican piezas repetitivamente mientras que en software siempre se crean productos diferentes por lo tanto las propiedades internas de Kanban son distintas.

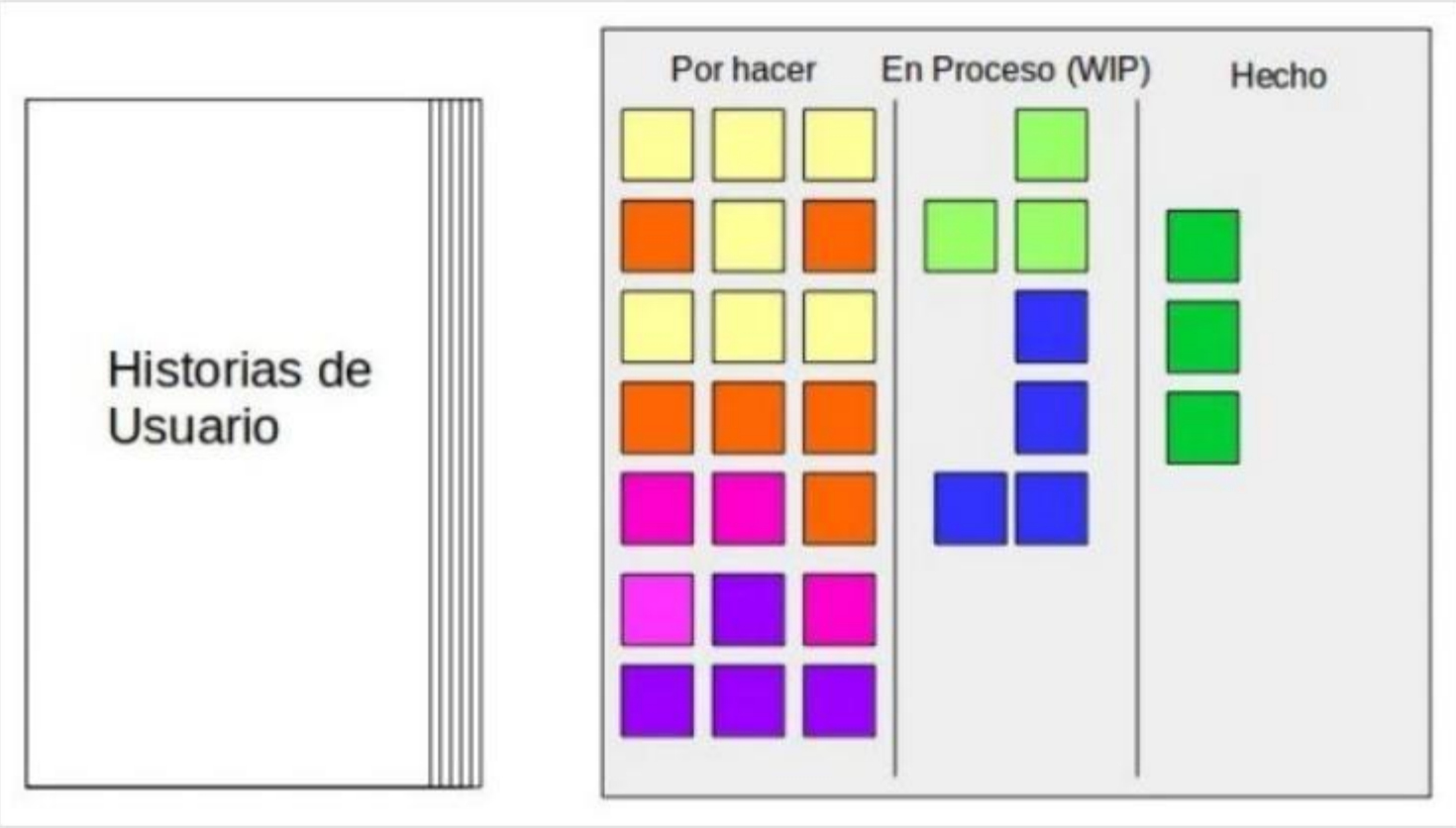
Por otro lado, Lean, Kanban, Kaizen hacen un "anillo al dedo" para las metodologías ágiles en conceptos de alto nivel.

Este diagrama muestra que Kanban se encuentra dentro de Lean que es un conjunto de metodologías para manufactura a su vez dentro de Kaizen que es la mejora continua. Los eventos Lean dentro de Kaizen son un conjunto de actividades que están encaminadas a realizar una pequeña mejora en un proceso mismo que requiere de poca inversión pero que representa un pequeño incremento en mejora continua.

DESCRIPCION DEL PROCESO

1. Las historias de usuario son recolectadas es importante no menos preciar ninguna característica ya que las historias cortadas representan funcionalidades incompletas e inútiles para el cliente.
2. Las historias de usuario son separadas en funcionalidades y cada una es evaluada para ser desarrollada por un equipo.
3. Las funcionalidades son evaluadas por el cliente y si este las aprueba sin cambio alguno, entonces es integrado al producto y sino es devuelta para realizar cambios.
4. Las funcionalidades pasan a formar parte del producto final.

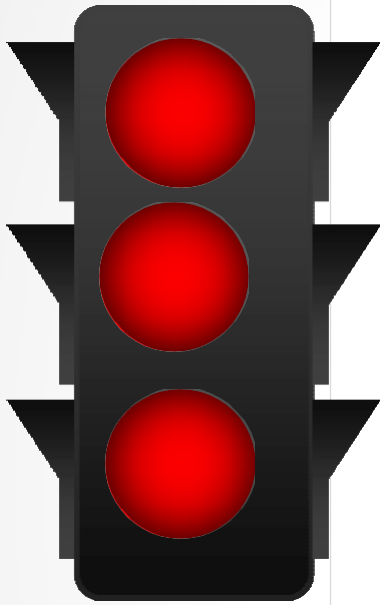
DESCRIPCION DEL PROCESO II



VENTAJAS

- Facilidad de entendimiento y exposición de información a todos los miembros involucrados.
- Facilidad de integración con metodologías ágiles (Scrumban). Acepta el ingreso de cambios a último momento confacilidad.
- El más adecuado para proyectos que se encuentran en mejora contínua.
- No todas las desventajas de Kanban surgidas en manufactura aplican en la industria del softwar puesto que la implementación interna es diferente.

DESVENTAJAS



- Menor efectividad en situaciones de Recursos Compartidos: Las órdenes no frecuentes vuelven ineficientes a kanban ya que se tiene que asegurar una producción suficiente por parte de un proceso mientras que a su vez el proceso que es no-frecuente es ejecutado.
- Kanban asume sistemas de producción repetitivos dada la naturaleza de su creación en el área de manufactura.
- Posiblemente Kanban pueda arrojar productos de baja calidad que requieren de ser retrabajados. Kanban funciona a manera de semáforo para administrar el tráfico y así cumplir con las necesidades del cliente indicando cuando empezar, cuando bajar el ritmo y cuando parar. Cualquier variabilidad o evento no esperado puede afectar el funcionamiento del sistema provocando que se generen señales confusas.
- Calidad: El sistema Kanban lleva los niveles de inventario cerca de 0 lo que en caso de alta incertidumbre e interrupciones en la red de transporte representa un peligro ya que eso significa que los clientes se quedan sin suministro de partes.

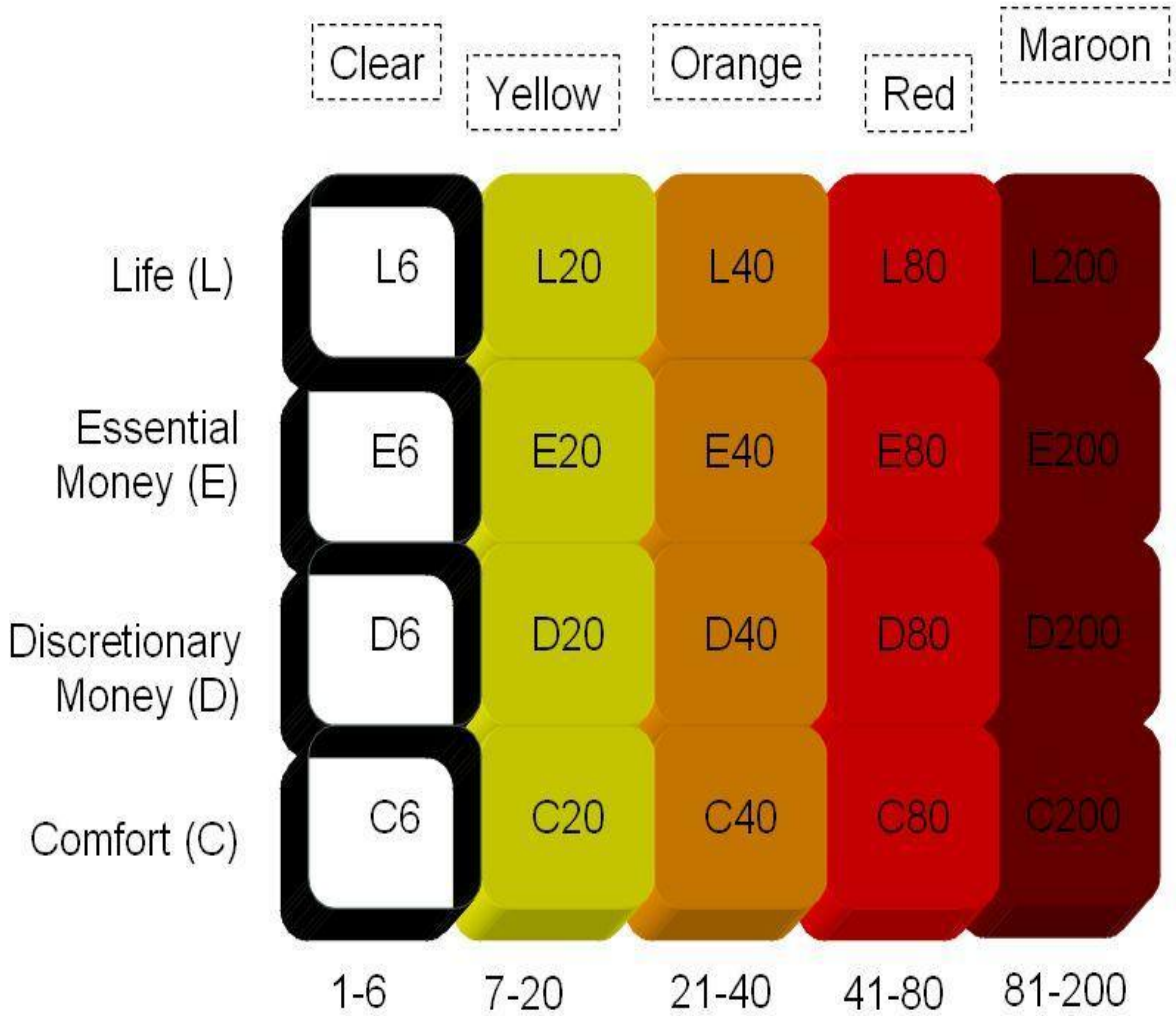
CRISTAL

CARACTERISTICAS

- Son una familia de metodologías ágiles, donde cada una de ellas está adecuada para un tipo de proyecto.
- Los proyectos pueden caracterizarse por dos dimensiones: Color y Dureza
- Cada miembro de la familia cristal esta asociado a un color que indica el nivel de "heaviness" de la metodologia (colores oscuros son mas pesados)

- Crystal sugiere elegir color apropiado basado en el tamaño del proyecto y su criticidad.
- Proyectos grandes, que necesitan más coordinación y comunicación, se asocian con colores más oscuros.
- Proyectos en los que un fallo pueda causar mayores problemas, también se asocian con colores más oscuros.

LA FAMILIA


Clear
Yellow
Orange
Red
Maroon
Life (L)
L6
L20
L40
L80
L200
Essential Money (E)
E6
E20
E40
E80
E200
Discretionary Money (D)
D6
D20
D40
D80
D200
Comfort (C)
C6
C20
C40
C80
C200
1-6
7-20
21-40
41-80
81-200

Clear, para equipos de hasta 6 personas o menos.

Amarillo, de entre 10 y 20 personas.

Naranja, para equipos entre 20 y 50 personas.

Roja, entre 50 y 100 personas.

Marron,,entre 50 y 100 personas.

CLEAR

- Diseñado para proyectos pequeños (D6, máximo 6 personas)
- Un equipo usando Crystal Clear debería estar localizado en una oficina compartida
- Incrementos de 2 a 3 meses.

AMARILLO

- Diseñado para proyectos de tamaño medio (D40, hasta 40 personas máximo)
- Proyectos de 1 a 2 años de duración
- Incrementos de hasta 4 meses

.

NARANJA

- Diseñado para proyectos de tamaño medio (D40, hasta 40 personas máximo)
- Proyectos de 1 a 2 años de duración
- Incrementos de hasta 4 meses .

ROJO

- Diseñado para proyectos de tamaño medio (D40, hasta 40 personas máximo)
- Proyectos de 1 a 2 años de duración
- Incrementos de hasta 4 meses

MARRON

- Diseñado para proyectos de tamaño medio (D40, hasta 40 personas máximo)
- Proyectos de 1 a 2 años de duración
- Incrementos de hasta 4 meses

PRACTICAS CRYSTAL

- Entrega incremental.
- Seguimiento por hitos, basados en entregas del producto.
- Usuario involucrado en el desarrollo.
- Regresión automatizado.
- Dos usuarios viendo el reléase.
- Taller para productos y ajustes de metodología en la mitad de cada incremento.

PROGRAMACION EXTREMA

CARACTERISTICAS

- La Programación Extrema es exitosa debido a que pone énfasis en la satisfacción del cliente.
- En lugar de entregar un producto de software completo a futuro, este proceso entrega el software a medida que se necesita y satisfaciendo la necesidad del cliente.
- La Programación Extrema permite a los desarrolladores responder con confianza ante los cambios en los requisitos de los clientes, incluso al final del ciclo de vida.

CARACTERISTICAS CONT.

- La Programación Extrema enfatiza el trabajo en equipo.
- Los gerentes, clientes y desarrolladores son socios iguales en un equipo colaborativo.
- La Programación Extrema implementa un entorno simple y efectivo que permite a los equipos ser altamente productivos.

- El equipo se auto-organiza alrededor del problema para resolverlo de la manera más eficiente posible.

CARACTERISTICAS CONT. 2

- Los programadores extremos se comunican constantemente con sus clientes y con sus compañeros de equipo.
- La Programación Extrema mantiene su diseño simple y limpio.
- Recibe retroalimentación y comentarios debido a que el software es testeado desde el primer día.
- Se entrega el sistema a los clientes lo antes posible y se implementan los cambios sugeridos.
- La Programación Extrema mejora un proyecto de software con cinco valores esenciales como la Sencillez, Comunicación, Retroalimentación, Respeto y Coraje.

VALORES EN LA PROGRAMACION EXTREMA

Sencillez

Se hace lo necesario y solicitado pero no más, esto maximiza el valor creado en la inversión realizada hasta el momento.
Dar simples y pequeños pasos hacia la meta y mitigar fallas a medida que estas ocurren.

Retroalimentacion

Se toma el compromiso de cada Iteración de forma seria, se entrega un software funcionando en cada iteración.
Realizar un demo del producto de forma temprana y se presta atención a los posibles cambios necesarios en el producto.

Comunicacion

Todos son parte del equipo y la comunicación es cara a cara todos los días.
Trabajar todos juntos en todo, desde los requisitos hasta la codificación.
Crear juntos la mejor solución para el problema.

Respeto

Cada miembro de equipo da y siente el respeto que merece como un miembro valioso del equipo.
Los desarrolladores respetan la experiencia de los clientes y viceversa.
La gerencia respeta el derecho a aceptar responsabilidad y recibir autoridad sobre el trabajo del equipo.

Coraje

Mostrar datos reales sobre el progreso y las estimaciones.
No se documentan excusas para un fracaso al contrario se documenta para tener éxito.
No existen temores por que nadie trabaja solo.
Adaptación a los cambios cuando ocurran.

REGLAS DE LA PROGRAMACION EXTREMA

PLANIFICACION

- ❑ Se deben escribir las historias de usuario.
- ❑ Crear el Plan de Lanzamiento (Release Planning) y el calendario de lanzamiento.
- ❑ Hacer pequeños lanzamientos de forma frecuente.
- ❑ Dividir el proyecto en Iteraciones.
- ❑ Cada iteración comienza con una planificación.

MANAGING

- ❑ Dar al equipo un espacio de trabajo abierto y dedicado.
- ❑ Establecer un ritmo sostenible.
- ❑ Reunirse de pie cada día (Daily Stand up meeting).
- ❑ Medir la velocidad del proyecto.
- ❑ Arreglar/solucionar problemas cuando se rompe el uso de XP

DISEÑO

- ❑ Simplicidad.
- ❑ Elegir una metáfora para el sistema.
- ❑ Usar tarjetas CRC para las sesiones de diseño.
- ❑ Crean soluciones Spike para reducir el riesgo.
- ❑ Ninguna funcionalidad es adicionada antes.
- ❑ Refactor donde y cuando sea posible.

REGLAS DE LA PROGRAMACION EXTREMA

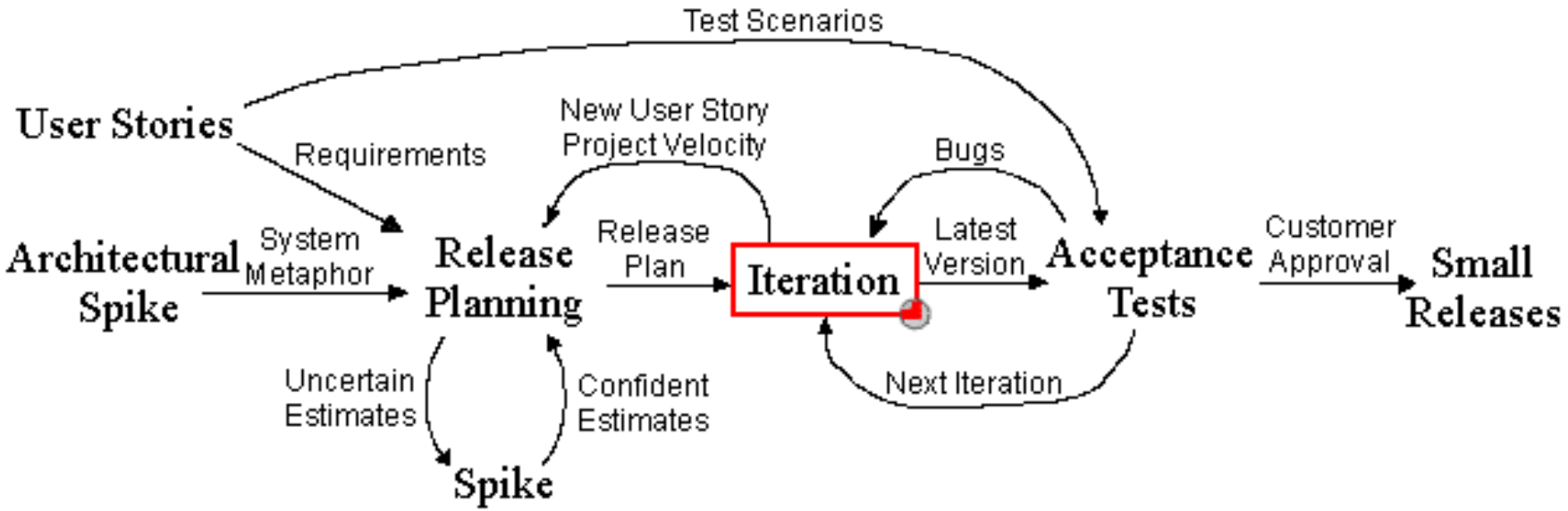
CODIFICACION

- El cliente está siempre disponible.
- El Código debe escribirse según las normas acordadas.
- Codificar primero las Pruebas Unitarias.
- Todo el código debe ser programado en parejas (Pair Programming).
- Sólo un par (Pair) integra el código a la vez.
- Integración Continua → Integrar a menudo.
- Integración Continua → Tener una computadora dedicada para la Integración.

PRUEBAS

- Todo el código debe tener Pruebas Unitarias.
- Todas las Pruebas Unitarias en el código deben ser exitosas al momento de correrlas antes de que este sea liberado.
- Crear su correspondiente unit test cuando se encuentra un error en el código.
- Las Pruebas de Aceptación son realizadas con frecuencia y los resultados de estas son publicados.

PRUEBAS


Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural
Spike
System
Metaphor
Release
Planning
Release
Plan
Iteration
Latest
Version
Acceptance
Tests
Customer
Approval
Small
Releases
Uncertain
Estimates
Confident
Estimates
Spike
Next Iteration

HISTORIAS DE USUARIO

- Las Historias de Usuario tienen el mismo propósito que los Casos de Uso, pero no son lo mismo.
- Se utilizan para crear estimaciones de tiempo en la Reunión de Planificación de Lanzamiento.
- Las Historias de Usuario se utilizan en lugar de usar un documento de Requisitos grande.
- Son escritas por los clientes como funcionalidades que el sistema debe hacer por ellos.
- Las Historias de Usuarios impulsan la creación de las Pruebas de Aceptación.
- Cada Historia de Usuario obtiene una estimación de 1, 2 o 3 semanas en "tiempo de desarrollo ideal".

PLAN DE LANZAMIENTO (RELEASE PLANING)

- Después de que se escriben las Historias de Usuario, se puede usar una Reunión de Planificación para crear un Plan de Lanzamiento.
- El Plan de Lanzamiento específica qué Historias de Usuario se implementarán para cada lanzamiento del sistema y las fechas de esos lanzamientos.
- El Plan de Lanzamiento proporciona un conjunto de Historias de Usuario para que los clientes elijan durante la Reunión de Planificación de la próxima Iteración.
- Estas Historias de Usuario seleccionadas se traducen luego en tareas de programación individuales que se implementarán durante la Iteración y así completar las Historias de Usuario.

ITERACION

- El desarrollo iterativo agrega agilidad al proceso de desarrollo.
- Dividir el proyecto de desarrollo en iteraciones de 1 a 3 semanas de duración. Una semana es la mejor opción aunque parezca muy corta.
- Mantener la longitud de la Iteración de forma constante durante todo el proyecto. Es esta constante la que hace que la medición del progreso y la planificación sean simples y confiables en XP.
- Tener una Reunión de Planificación de Iteración al comienzo de cada Iteración para planificar qué se hará. La planificación constante es una manera fácil de mantenerse al tanto de los requisitos cambiantes de los usuarios.
- Seguir el progreso durante cada Iteración. Si parece que todas las tareas no serán terminadas, se llama a otra Reunión de Planificación de Iteración, donde se vuelve a estimar y eliminar algunas de las tareas.

PLAN DE ITERACION(ITERATION PLANING)

- Al comienzo de cada Iteración se llama a una Reunión de Planificación de Iteración, esto para producir el plan de tareas de programación de esa iteración.
- Cada Iteración tiene de 1 a 3 semanas de duración.
- Las Historias de Usuario son elegidas en cada Iteración por el cliente primero según el orden más valioso para el.

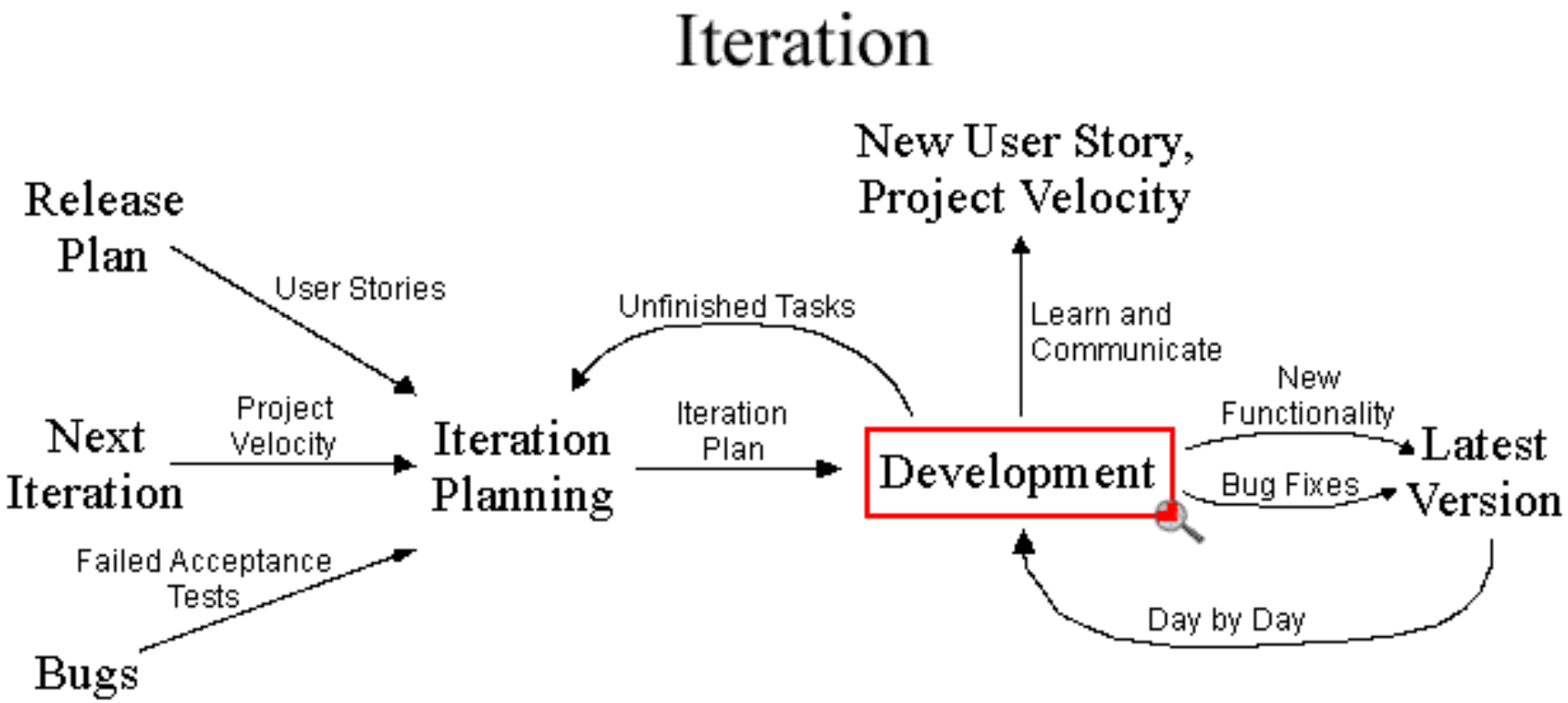
REUNION DIARIA (STAND UP MEETING)

- La comunicación entre todo el equipo es el objetivo de la reunión diaria (Stand Up Meeting).
- La reunión diaria de cada mañana se usa para comunicar problemas, soluciones y promover el enfoque del equipo.
- Todos se ponen de pie en un círculo para evitar largas discusiones.

VELOCIDAD DEL PROYECTO

- La velocidad del proyecto es una medida de la cantidad de trabajo que se está realizando en su proyecto.
- Para medir la velocidad del proyecto, se suma las estimaciones de las historias de usuario que se terminaron durante la iteración.
- Se suma también las estimaciones de las tareas finalizadas durante la iteración.
- Ambas medidas se utilizan para la Planificación de la Iteración.
- Durante la Reunión de Planificación de la Iteración, los clientes pueden elegir el mismo número de Historias de Usuario donde su puntaje sea igual a la velocidad del proyecto medida en la iteración anterior.

ITERACION


Iteration
New User Story,
Project Velocity
Release
Plan
User Stories
Unfinished Tasks
Learn and
Communicate
New
Functionality
Project
Velocity
Iteration
Plan
Next
Iteration
Iteration
Planning
Development
Latest
Version
Bug Fixes
Failed Acceptance
Tests
Day by Day
Bugs

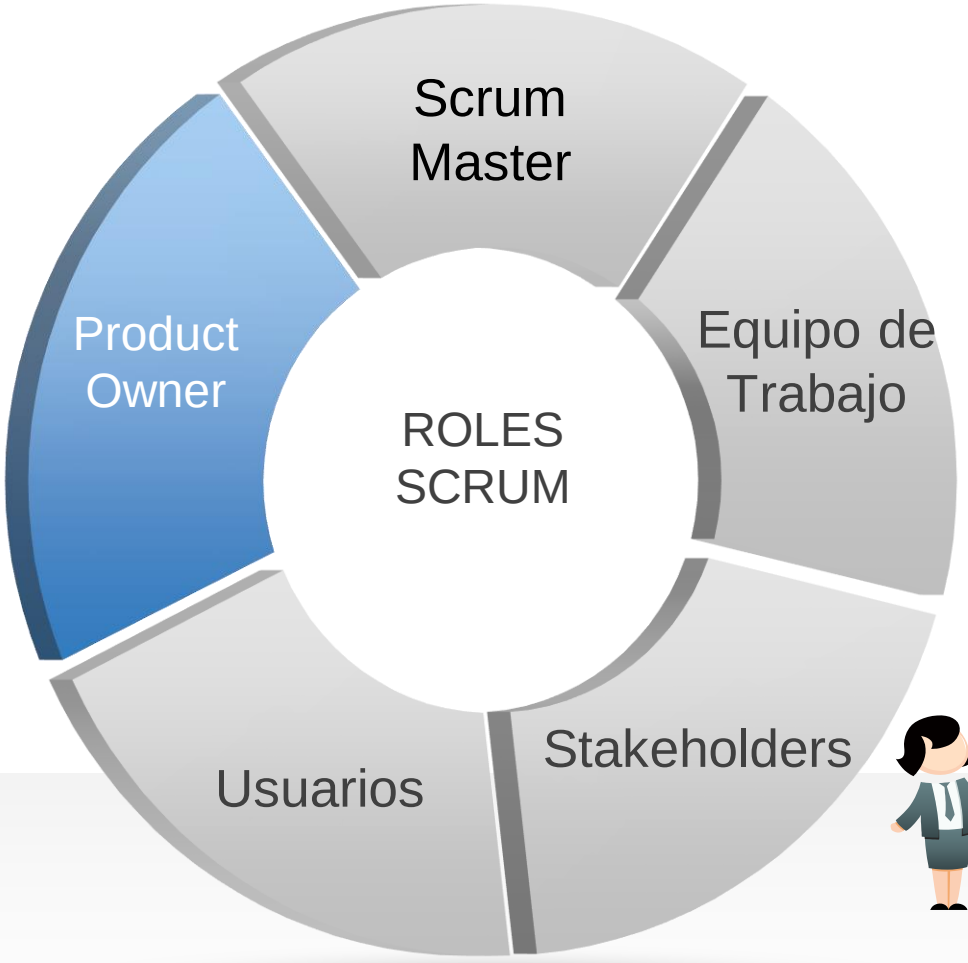
SCRUM

Scrum

Scrum es un marco de trabajo que nos permite encontrar prácticas emergentes en dominios complejos, como la gestión de proyectos de innovación. No es un proceso completo, y mucho menos, una metodología. En lugar de proporcionar una descripción completa y detallada de cómo deben realizarse las tareas de un proyecto, genera un contexto relacional e iterativo, de inspección y adaptación constante para que los involucrados vayan creando su propio proceso. Esto ocurre debido a que no existen ni mejores ni buenas prácticas en un contexto complejo. Es el equipo de involucrados quien encontrará la mejor manera de resolver sus problemáticas. Este tipo de soluciones serán emergentes.

Roles Scrum

Roles Scrum



PRODUCT OWNER

- Determinar la **visión** del producto.
- Gestionar las **expectativas de los stakeholders.**
- Recolectar los **requerimientos.**
- Determinar y conocer en detalle las **características funcionales** de alto y de bajo nivel.
- Generar y mantener el **plan de entregas** (*release plan*): fechas de **entrega** y **contenidos** de cada una.
- Maximizar la **rentabilidad** del producto
- Determinar las **prioridades** de cada una de las características por sobre el resto
- Cambiar las prioridades de las características según avanza el proyecto, acompañando así los cambios en el negocio
- Aceptar/rechazar el producto construido durante el Sprint y proveer **feedback** valioso para su evolución.
- Participar de la revisión del Sprint junto a los miembros del Equipo de Desarrollo para obtener feedback de los *stakeholders*.

Roles Scrum

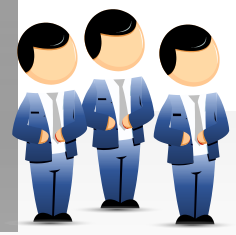
SCRUM MASTER

- Remueve impedimentos.
- Velar por el **correcto empleo** y **evolución** de Scrum
- Facilitar el **uso de Scrum** a medida que avanza el tiempo. Esto incluye la responsabilidad de que todos asistan a tiempo a las daily meetings, reviews y retrospectivas.
- Asegurar que el equipo de desarrollo sea **multifuncional** y eficiente.
- **Proteger** al equipo de desarrollo de distracciones y trabas externas al Proyecto.
- Detectar, monitorear y **facilitar la remoción de los impedimentos** que puedan surgir con respecto al proyecto y a la metodología.
- Asegurar la **cooperación** y **comunicación** dentro del equipo.

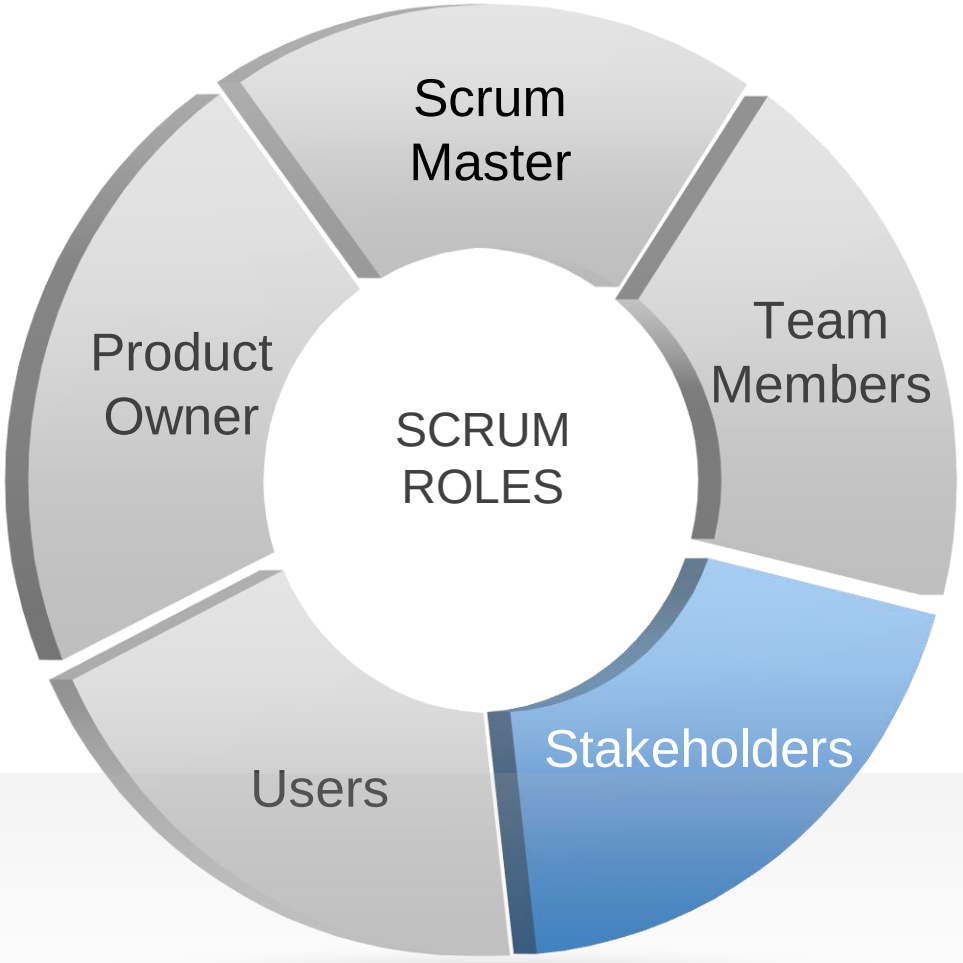
Roles Scrum

TEAM MEMBERS

- Tiene la responsabilidad de entregar el producto.
- Formado por 7±2 personas con las habilidades transversales necesarias para realizar el trabajo (diseñador, desarrollador, etc).



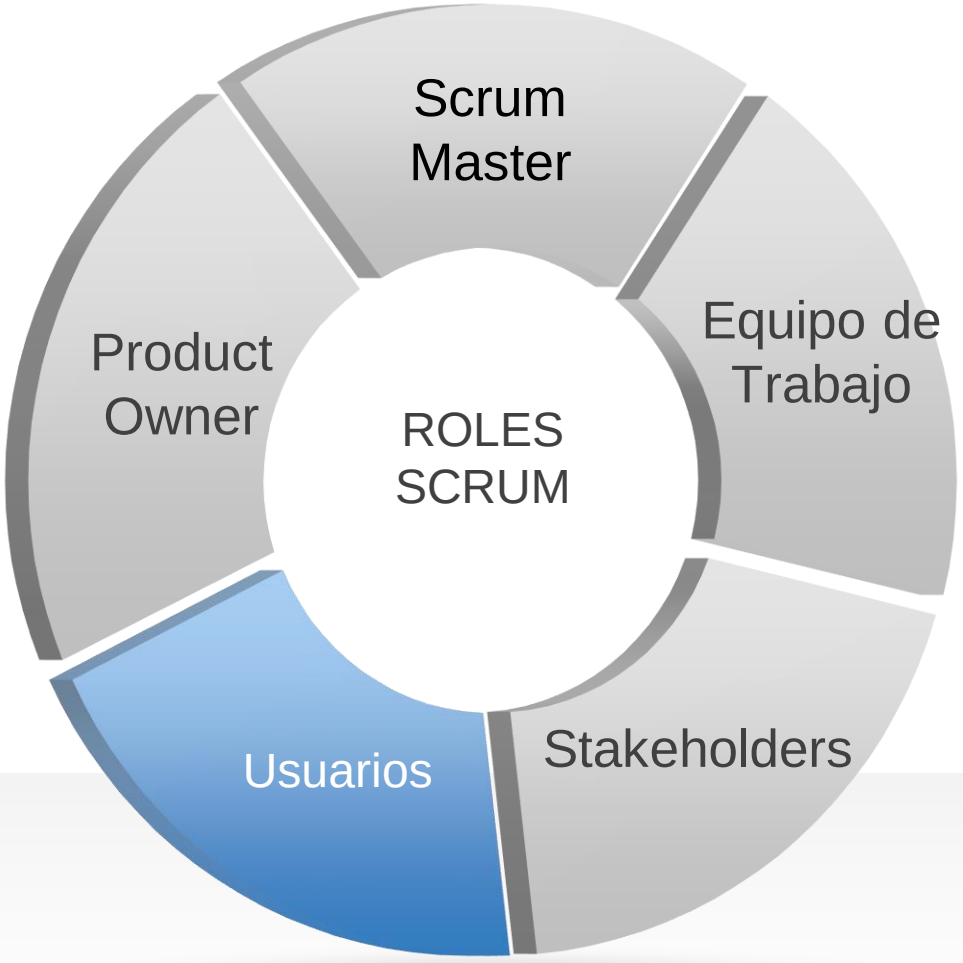
Roles Scrum



STAKEHOLDERS

- Se refiere a la gente que hace posible el proyecto y para quienes el proyecto producirá el beneficio acordado que lo justifica.
- Asesoran y observan
- Sólo participan directamente durante las revisiones del sprint.

Roles Scrum



USUARIOS

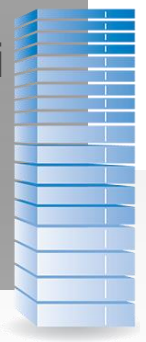
- Son aquellas personas para las que se desarrolla el producto.
- El rol mas importante de todos.
- Sin usuarios no necesitamos un equipo de trabajo, backlogs, nada en realidad.
- A pesar de que el Product Owner actua como proxy para los usuarios, ultimamente los usuarios determinan si un equipo esta entregando un product con valor.

Artefactos Scrum

Artefactos Scrum

PRODUCT BACKLOG

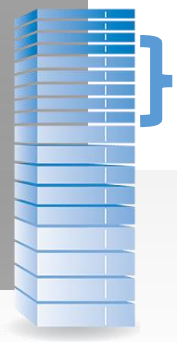
- Es una lista de cosas que expresa los requerimientos del producto.
- Representa todo el trabajo al que se aspira realizar y completar
- Cada item tiene un valor medible por el cliente.
- Cada item es priorizado.
- Para recorder: Si algo esta en el backlog, esto debe ser terminad. Si algo no esta en el backlog este no sera jamas trabajado.



Artefactos Scrum

SPRINT BACKLOG

- El Sprint Backlog es el conjunto de PBIs que fueron seleccionados para trabajar en ellos durante un cierto Sprint, conjuntamente con las tareas que el equipo de desarrollo ha Identificado que debe realizar para poder crear un incremento funcional potencialmente entregable al finalizar el Sprint.



Artefactos Scrum

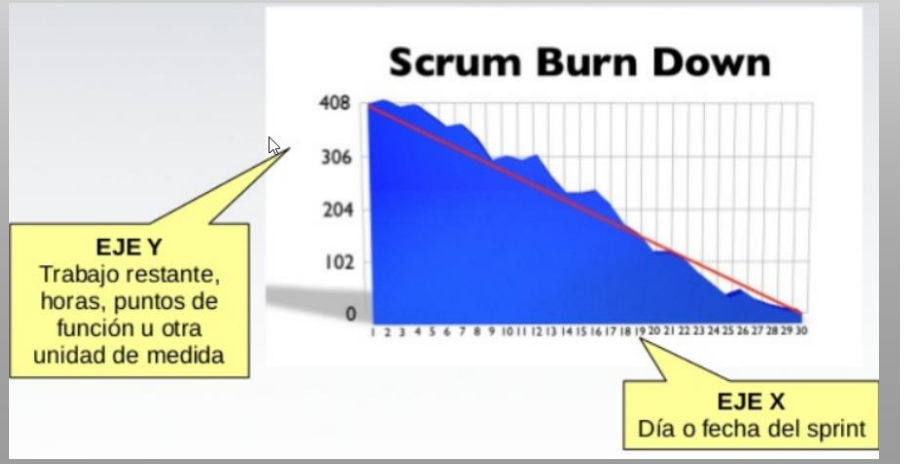
WORKING INCREMENT

Participación del cliente y transparencia sara que pueda guiar de manera regular los resultados del Proyecto.

Para asi lograr entregas cortas y regulares del product final (cada 2-4 semanas) para obtener feedback e irse acercando a las expectativas del cliente.

Artefactos Scrum

METRICS



Artefactos Scrum

EPICS, FUNCIONALIDADES , HISTORIAS

- Los backlogs estan hechos de lo siguiente:
  Epics:
  - reside en el Portafolio del backlog
  - Elaborado dentro las Funcionalidades
- Funcionalidades:
  - Reside en el Programa/Product backlog
  - Describe funcionalidades deseadas a alto nivel.
- Historias:
  - Reside en el Backlogs del equipo
  - Describe comportamientos deseados por el Sistema desde el punto de vista del usuario.

El Proceso

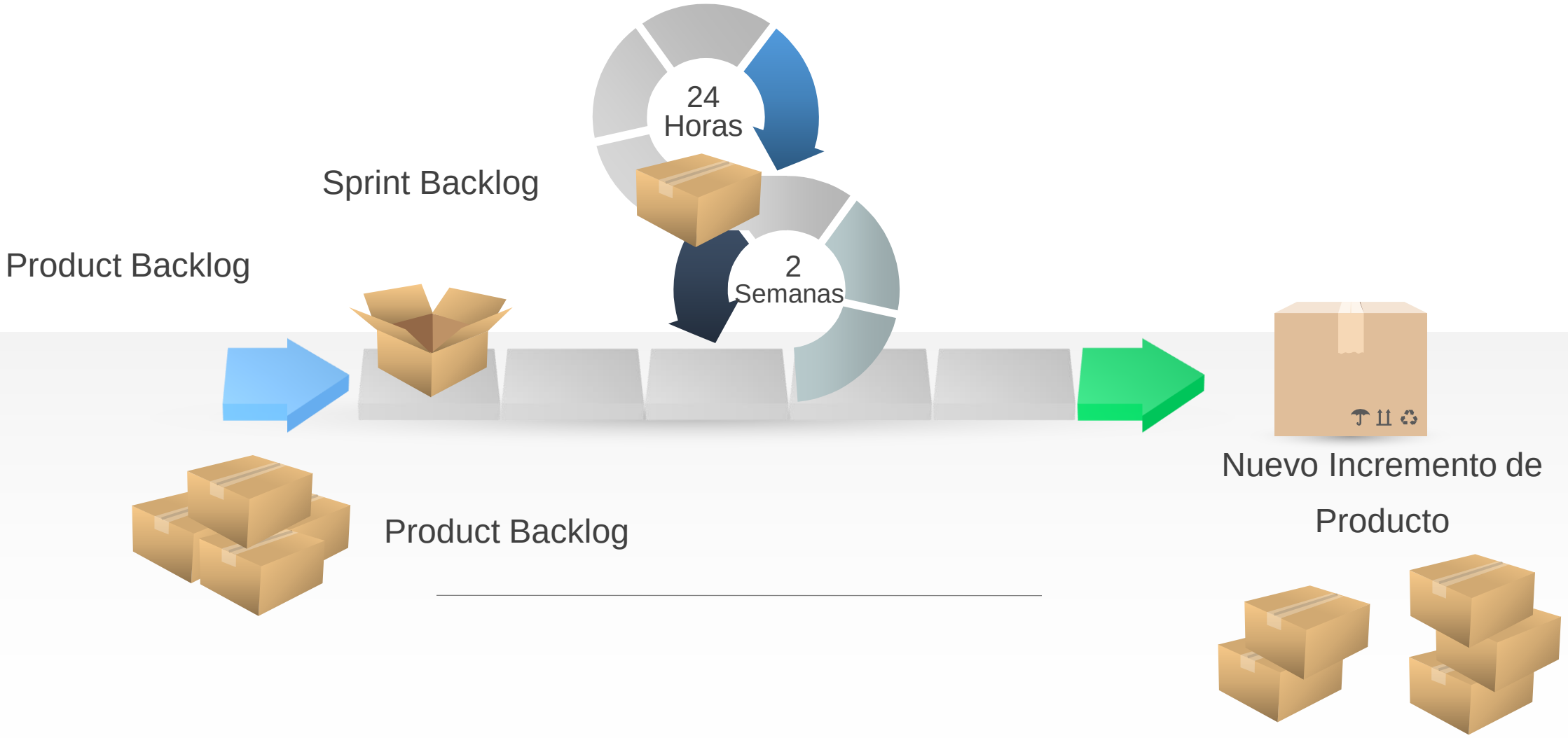
El Proceso Scrum

Ceremonia: un evento de importancia.

Ritual: actividades realizadas en un lugar, de acuerdo con una secuencia establecida y forma

Grooming
Planificación
Reporte Diario
Demo
Retrospectiva

El Proceso


24
Horas
Sprint Backlog
Product Backlog
2
Semanas
Nuevo Incremento de
Producto
Product Backlog

HISTORIAS DE USARIO

HISTORIAS DE USUARIO

Las Historias de Usuario surgieron en eXtremme Programming (XP) como una respuesta a una situación habitual en los proyectos de desarrollo do software: los clientes o especialistas de negocio se comunican con los equipos de desarrollo a través de extensos documentos conocidos como especificaciones funcionales. A su vez, las especificaciones funcionales son la documentación de supuestos y están sujetas a interpretaciones, lo que causa malos entendidos y que finalmente el software construido no se corresponda con la realidad esperada.

COMPONENTES DE UNA HISTORIAS DE USUARIO

- **Card (Ficha) –** Toda historia de usuario debe poder describirse en una ficha de papel pequeña. Si una Historia de Usuario no puede describirse en ese tamaño, es una señal de que estamos traspasando las fronteras y comunicando demasiada información que debería compartirse cara a cara.
- **Conversación –** Toda historia de usuario debe tener una conversación con el Product Owner. Una comunicación cara a cara que intercambia no solo información sino también pensamientos, opiniones y sentimientos.
- **Confirmación –** Toda historia de usuario debe estar lo suficientemente explicada para que el equipo de desarrollo sepa qué es lo que debe construir y qué es lo que el Product Owner espera. Esto se conoce también como Criterios de Aceptación.

<Título>
Como <*rol*>
quiero <*característica*>
para <*beneficio*>

Título: Generar el PDF del expediente
Como estudiante
quiero generar un PDF con mi expediente
para *guardar un resumen de mi expediente y poder entregarlo a quien me lo pida*

INVEST

Independent
Negotiable
Valuable
Estimable
Small
Testable

Una buena historia tiene que tener seis atributos:

- **Independiente:** dependencias entre las historias crean problemas de priorización y estimación
- **Negociable:** no son contratos, son recordatorios de conversaciones
- **Valiosa:** las historias deben ser valiosas para los que pagan el software
- **Estimable:** el tamaño de la historia debe poder ser estimado, aunque sea de forma gruesa.
- **Pequeña:** para poder estimarse correctamente, es recomendable que la historia sea pequeña; hay que dividir las épicas
- **Testable:** las historias deben ser probadas y los tests deben poder ser automatizados

CRITERIO DE ACEPTACION

- Las historias de usuario se completan con unos criterios de aceptación.
- Estos criterios nos sirven para confirmar la implementación que se está desarrollando, las pruebas a realizar y cómo verificar de que se ha completado el trabajo.

DEFINICION DE LISTO

- También conocido como **Definition of Ready**, es el conjunto de características que una Historia de Usuario debe cumplir para que el Equipo de Desarrollo pueda comprometerse a su entrega, es decir, incluirla en un Sprint Backlog.
- Una típica definición de listo podría ser:
  - La Historia de Usuario debe ser INVEST.
  - Todos sus pre-requisitos están resueltos (ej: dependencias con otros Equipos)

DEFINICION DE TERMINADO

- También conocido como Definition of Done, es el conjunto de características que una Historia de Usuario debe cumplir para que el equipo de desarrollo pueda determinar si ha terminado de trabajar en ella. Un típico criterio de “Terminado” podría ser: • Todos los criterios de aceptación funcionan correctamente • Todos los archivos fuentes están en el repositorio de código fuente y el build se ejecutó exitosamente

Identificación de **MVP** y Entregas: Entregable 1

- Venta de Evento
  - Crear Evento
    - Crear un evento confirmado
    - Ver listado de eventos confirmados
    - Modificar evento confirmado
    - Cancelar evento confirmado
  - Difundir Evento
    - Listar evento en sitio web
  - Responder Consultas
    - Publicar detalles de evento
    - Generar texto con fechas y valores para “Copy & Paste” en e-mail de respuesta
  - Visualizar Estado de Inscripciones
    - Dashboard de inscripciones a cursos
- Registración a Evento
  - Pre-Inscripción a Evento
    - Pre-Inscripción individual
  - Confirmación de Inscripción
    - Notificar Inscripción
    - Confirmar inscripción sin pago (pago a cuenta)
- Cobranza de Evento
  - Seguimiento de Cobros Pendientes
    - Listado de Pagos Pendientes por evento
  - Pago de Pre-inscripción
    - Pagar en/por
      - Efectivo
      - Cheque
      - Transferencia Bancaria
  - Procesar Cobro
    - Registro de Pago
    - Notificación de cobro a Gestor de Finanzas

Ejemplos de Historias de Usuario Entregable 1

Entrega 1 - Comercializar Eventos				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
1	Comercial	Crear un evento confirmado	Hacer el seguimiento del mismo	- Debe tener Nombre, Fecha, Descripción, Destinatarios, Programa, Instructor, Lugar, Ciudad, País, Capacidad, Precios y Promociones: SEB (Super Early Bird), EB (Early Bird), dto. en % para 2 personas y dto. en % para 3 o más personas. - Las promociones son opcionales. - Las fechas de SEB y EB deben ser anteriores a la fecha del evento - Por defecto SEB=30 días antes, EB=10 días antes, 2personas=-10%, 3+ personas=-15%. - Un evento puede ser público o privado
2	Comercial	Ver listado de eventos confirmados	No superponer eventos	- Mostrar Nombre, Ciudad y País - Muestra solo los futuros - Ordenado por fecha ascendente
3	Comercial	Modificar evento confirmado	Corregir cualquier error o re programarlo	- Permite modificar todos los campos.
4	Comercial	Cancelar evento confirmado	Dejar de seguirlo	- Desaparece del listado de eventos confirmados.
5	Comercial	Listar los eventos en un sitio web	Que los interesados	- Solo se listan los eventos públicos - Listado por fechas (a futuro)

Ejemplos de Historias de Usuario Entregable 1 Cont.

EPICS Agrupaciones de varias Historias de usuario

6	Comercial	Publicar los detalles de cada evento	Que los interesados puedan verlos	- Accesible desde el listado de eventos - Muestra los detalles de cada evento: Nombre, Fecha, Descripción, Destinatarios, Programa, Instructor, Lugar, Ciudad, País, Precios y Promociones
7	Comercial	Generar un texto con fechas y valores	Pegarlo en los e-mail de respuesta	- Debe generarlo agrupando por ciudad, cursos, fechas y precios de cada uno.
8	Comercial	Dashboard de inscripciones a cursos	Conocer el estado de completitud de cada curso	- Muestra los eventos con colores: Rojo, Naranja, Amarillo y Verde. Los criterios son: → Un evento debe estar al 50% al menos 15 días antes → Un evento debe estar al 75% al menos una semana antes → Un evento debe estar al 100% dos días antes - La varianza sobre esos números alteran los colores: → Menos del 50% (Rojo) → Del 50% al 75% (Naranja) → Del 75% al 90% (Amarillo) → Del 90% al 100% (Verde)
9	Interesado	Pre-Inscribirme	Iniciar la reserva de mi vacante	Debe solicitar Nombre*, Apellido*, Teléfono de Contacto*, Email*, Empresa/Carrera, Rol y Solicitar confirmación de que el asistente llevará notebook* si el curso lo requiere. * = obligatorio, el resto, opcional.
10	Comercial	Ser notificado de cada inscripción	Poder reaccionar en tiempo real frente a cada una	El email debe ser enviado a una dirección de correo configurable indicando los datos de contacto de la persona que realizó la inscripción.
11	Comercial	Confirmar la inscripción sin pago (pago a cuenta)	Financiar ciertas vacantes	Un pre-inscripto puede convertirse en inscripto sin haber realizado el pago.

Ejemplos de Historias de Usuario Entregable 1 Cont. 2

13	Interesado	Pagar en efectivo	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar en efectivo.
14	Interesado	Pagar con Cheque	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar con cheque.
15	Interesado	Pagar por Transferencia Bancaria	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar por transferencia bancaria.
16	Comercial	Registrar los Pagos	Realizar el seguimiento de los pagos	Una pre-inscripción puede convertirse en inscripción registrando el pago realizado (fecha, monto y forma de pago).
17	Gestor de Cobranzas	Ser notificado del cobro de un evento	Realizar el seguimiento de los pagos	Cada vez que una pre-inscripción se convierte en inscripción, se debe enviar un e-mail a una casilla configurable.