TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN UTN-FRC

TUP

UTN

Facultad Regional Córdoba

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

# METODOLOGIA DE SISTEMAS I

## Unidad Temática 1: Sistema de Información - Metodologías

Material de Estudio

2do Año – 4to Cuatrimestre

CC BY ND NC

V.0.1

# Índice

## Introducción a los Sistemas de Información

Para comenzar con el estudio de esta unidad, necesitamos repasar algunos conceptos claves relacionados con la Teoría General de los sistemas y el enfoque sistémico.

También nos proponemos, complementar los conocimientos adquiridos en las asignaturas vinculadas al estudio de las organizaciones como sistemas y afines ya que, dentro de las primeras actividades prácticas de esta unidad trabajaremos sobre el análisis y resolución práctica de situaciones problemáticas relacionadas con pequeñas organizaciones y Pymes. Otra de las metas de esta unidad, tiene que ver con el análisis de sistemas que contribuya a que el futuro egresado de la carrera pueda evaluar opciones de desarrollar software como un futuro emprendimiento laboral integrando conocimientos y habilidades tanto del aspecto de los lenguajes de programación como del área del analista de sistemas.

Con respecto a la resolución de los casos prácticos; nos proponemos que el estudiante reconozca aspectos claves dentro del caso de estudio y pueda, mediante una producción escrita, describir en forma correcta y precisa cada una de las consignas presentadas utilizando un vocabulario pertinente a esta asignatura.

Para comenzar, arrancamos con una revisión de conceptos fundamentales del enfoque sistémico y la teoría general de los sistemas. Luego se continuará con el estudio de los sistemas de información como solución informática que contribuye a la toma de decisiones y los procesos para la gestión de servicios IT entre otros beneficios.

**Objetivo del sistema**: es el fin o finalidad, reconocido como la razón de ser del sistema. La redacción debe ser completa, es decir, que debe tener lo que hace, cómo lo hace y dónde lo hace. Cuando se lee el objetivo debe quedar claro que se trata de ese sistema y no de otro, es decir, debe ser lo más específico posible. Se redacta con verbos en infinitivo.

Por ejemplo, el objetivo “comercializar medicamentos” define el objetivo de cualquier sistema que se encargue de vender medicamentos. Por lo tanto, no estaría definiendo el fin del sistema bajo análisis de forma completa y específica; debe indicar que tipo de productos, en que forma de venta, con que cobertura.

Por ejemplo “Comercializar medicamentos y productos de perfumería y farmacia tanto para clientes particulares como beneficiarios de obras sociales según convenio vigente”.

**Ambiente del sistema**: El ambiente del sistema es todo aquello que se relaciona con el sistema y sobre lo que el sistema no tiene control y, por lo tanto, está fuera. El ambiente condiciona al sistema en el logro de sus objetivos. Cuando el

sistema es una organización, podemos encontrar las siguientes entidades: Competencia, entes reguladores, sindicatos, clientes, proveedores, entre otros.

**Alcance de un sistema:** El alcance de un sistema son las actividades o funciones que cumplen los elementos del sistema para alcanzar el objetivo. Se redactan en un listado, donde cada alcance se identifica con un verbo en infinitivo. Esta lista debe ser completa y abarcar el cumplimiento total del objetivo del sistema.

Por ejemplo, el alcance de una farmacia podría ser el siguiente:

- Recibir pedido de medicamentos del cliente.
- Verificar si hay stock del medicamento solicitado.
- Preparar pedido para entregar al cliente.

**Entrada-proceso-salida:**

***Entradas***: Llamados también insumos o input corresponden con aquellos elementos que introducidos desde el ambiente y en forma frecuente permiten el desarrollo del proceso. Las entradas son de distinto tipo, por ejemplo: materias primas, energía, dinero, e información.

***Proceso***: Son el conjunto de actividades que se desarrollan en el sistema para permitir que las entradas se transformen en salidas. También se pueden considerar como las funciones que se desempeñan para alcanzar los resultados o los alcances del sistema.

***Salidas***: También denominadas resultados o productos son aquellos elementos que egresan del sistema, cruzando los límites hacia el ambiente. Las salidas se encuentran directamente relacionadas con los objetivos.

**Recursos**: Cuando analizamos las organizaciones como sistemas identificamos sus elementos como Recursos. Las organizaciones son sistemas abiertos que interactúan con el ambiente y a partir de la interacción obtienen recursos. Es así que los recursos son los medios de los que dispone el sistema para realizar las actividades y por tanto permiten alcanzar el logro de los objetivos. Los recursos provienen del AMBIENTE, y el sistema los incorpora cada vez que los necesita. Pero también es importante notar que los recursos se encuentran dentro del sistema por lo que la organización planifica su distribución, los organiza, controla su funcionamiento y gestiona su desarrollo. Los RECURSOS implican para la organización un costo y por ello es importante lograr una adecuada identificación y uso.

## Información y sistema de información

### ¿Qué es la información?

Todos sabemos con certeza que la información no es un dato aislado difícil de entender por si solo que carece de utilidad si se encuentra aislado o sin otros grupos e datos que le permitan crear un mensaje coherente. Los datos por lo general son utilizados para comprimir información con la finalidad de facilitar el almacenamiento y su transmisión a otros dispositivos.

Por el contrario, la información tiende a ser muy extensa ya que se encuentra integrada por un conjunto de datos de diferentes tipos y representa un mensaje que tiene sentido comunicacional.

La información es el conjunto de datos adecuadamente procesados, para que puedan proveer un mensaje que contribuya a la toma de decisiones a la hora de resolver un problema, además de incrementar el conocimiento, en los usuarios que tienen acceso a dicha información.

Así, podemos afirmar que las funciones principales que tiene la información son:

- Aumentar el conocimiento del usuario.
- Reducir la incertidumbre de lo que no se conoce.
- Ayudar a la toma de decisiones.
- Permitir realizar el control de tares y eventos.

### La información en las organizaciones

Desde hace mucho tiempo las organizaciones han reconocido la importancia de la administración adecuada de todos sus recursos (tales como materia prima, mano de obra, etc.) y además, han reconocido que la información es uno de los recursos más importantes con los que cuenta cualquier tipo de organización.

Tal es el punto de importancia de la información que se considera un recurso fundamental y decisivo en las operaciones de las mismas y sobre todo, como un apoyo fundamental para la toma de decisiones y para la reducción de la incertidumbre a nivel estratégico, ayudando a las organizaciones a ser más competitivas para acompañar frente a la demanda del mercado actual.

Finalmente, podemos agregar, que es un recurso vital para cualquier proyecto, pero fundamentalmente, si se trata de un proyecto para desarrollar un software.

### Clasificación de la información por su relación con la toma de decisiones

Es importante destacar que, tanto la información que se genera, como también la que se necesita en los distintos niveles de la organización, posee ciertas características.

En este sentido, la información se clasifica como técnica, táctica o estratégica según los mismos niveles definidos para las organizaciones, como podemos observar en la figura (1).


Figura (1)

*Información Estratégica*: información para la toma de decisiones a largo plazo que pone en juego todos los recursos de la organización.

*Información Táctica*: información necesaria para la coordinación y manejo de los recursos y actividades de la organización. Decisiones a mediano plazo.

*Información Técnica*: información necesaria para la realización de las tareas operativas de la organización. Decisiones a muy corto plazo.

## Sistema de información

Un sistema de información (SI) es un conjunto de elementos orientados al tratamiento y administración de datos e información, generados y organizados para cubrir una necesidad o un objetivo.

Además, sabemos que las actividades básicas de un sistema de información son tres: entrada, procesamiento y salida de información. (Figura 2)


Figura (2)

Al igual que cualquier sistema abierto, un sistema de información tiene entradas, que son los datos obtenidos a partir de la operatoria de la organización, tiene un proceso, que tiene que ver con operaciones realizadas con los datos a fin de producir información (captación, verificación, ordenación, clasificación, cálculo, sumarización, etc.) y la salida que es el producto que se obtiene al procesar los datos y que se distribuye a los usuarios del sistema de información en forma de informes, listados, estadísticas, consultas por pantalla, etc. Además, dentro de las salidas del sistema de información se encuentran aquellas que son de retroalimentación, que ingresan nuevamente al sistema para su análisis. Como ejemplo de este tipo de salidas podemos nombrar la información de control. (Figura 3)

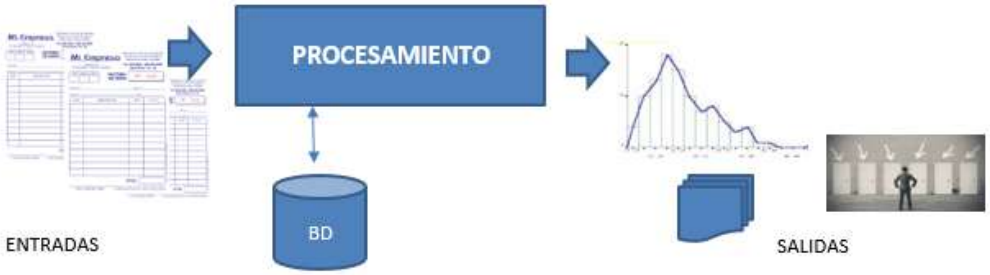



Figura (3)

Veamos un ejemplo.

“Office Mujer” es una empresa dedicada a la venta de prendas de vestir para mujeres que trabajan en oficinas y otras instituciones similares que necesitan vestirse con ropa formal pero contemporánea. Esta empresa trabaja tanto con ventas mayoristas como ventas en forma minorista. Además, la empresa se encuentra organizada de la siguiente forma (Figura 4):

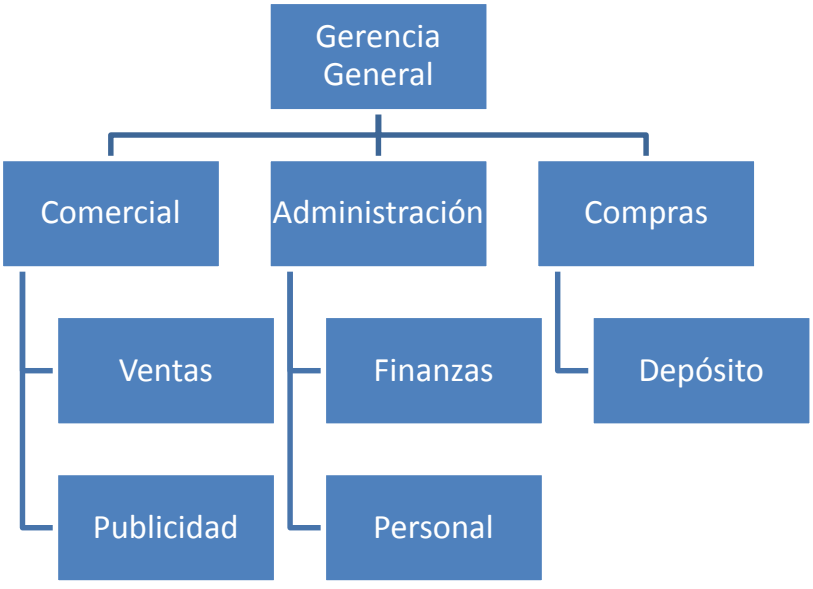



Figura (4)

- Gerencia General: encargada de evaluar el rendimiento de la empresa, elevando informes de los resultados obtenidos al dueño de la empresa y controla el desempeño de las otras gerencias.
- Comercial: se encarga de gestionar y controlar todo el movimiento de las ventas de la empresa, tiene a su cargo los departamentos de Ventas y Publicidad.
- Ventas: se encarga de realizar la venta del producto, como así también su facturación y cobro.
- Publicidad: se encarga de realizar las promociones y campañas publicitarias necesarias para la venta de ropa.
- Administración: se encarga de la gestión contable de la empresa. De este sector dependen Finanzas y Personal.
- Finanzas: realiza las cobranzas de las deudas a los clientes y pago a proveedores.
- Personal: se encarga de llevar todos los aspectos relacionados con los empleados de la empresa.
- Compras: está encargada de las compras de prendas de vestir como de los insumos necesarios para el funcionamiento de la empresa (papel, mostradores, etc.) De este sector depende Depósito.
- Depósito: encargado de recibir y organizar las prendas, ordenarlas por talles y tipo de colores.

**Se pide:**

1. Definir objetivo del sistema.
2. Representar Entrada-proceso-Salida.
3. Identificar el ambiente y límite del sistema.
4. Identificar alcance del sistema.
5. Identificar Sistemas y subsistemas.
6. Enumerar Recursos del sistema.

Veamos la resolución del caso práctico.

1. Objetivo: Comercialización de prendas de vestir femeninas en forma mayorista y minorista.
2. Representar las E-P-S.

| ENTRADAS | PROCESOS | SALIDAS |
|---|---|---|
| **Prendas de vestir.** | Controlar stock, organizar y embalar prendas. | Producción de prendas de vestir listas para la venta. Orden de compra Proveedor. |
| **Horas trabajadas, asistencia, inasistencias, aportes, retenciones.** | Calcular liquidaciones de sueldo. | Recibo de sueldo. |
| **Pedido de clientes.** | Atender al cliente, ofrecer productos. | Presupuesto. |
| **Pago de clientes.** | Asentar pago y emitir comprobante. | Factura Cliente. |
| **Listas de precios proveedor.** | Analizar y seleccionar proveedor. | Lista de precio y proveedor. |
| **Entregas proveedor.** | Atender al proveedor, controlar entrega. | Factura proveedor. |
| **Necesidades de moda y temporada.** | Analizar demanda y organizar publicidad. | Promociones y descuentos. |

Tabla (1)

3. ***Ambiente del sistema***: proveedores, competencia, clientes, bancos y tarjetas con los que trabaja.

   ***Límites***: desde que se solicitan las prendas de vestir a los proveedores hasta que se le cobra al cliente.

4. ***Alcances***:
   - Realizar selección de proveedores
   - Realizar y enviar pedido a proveedores.
   - Recibir la mercadería comprada.
   - Realizar pago a proveedores
   - Atender a los clientes
   - Cobrar a los clientes
   - Controlar el depósito.
   - Seleccionar el personal de la empresa.
   - Pagar sueldos a los empleados
   - Realizar campañas publicitarias y promociones respecto a las prendas que vende.

5. ***Sistema***: OfficeMujer

   ***Metasistema***: Cámara de Comercio de la ciudad de Córdoba.

   ***Subsistema***: Gerencia General, Comercial, Administración y Compras.

6. ***Recursos Humanos***: personal de nivel gerencial y empleados de las distintas áreas.

   ***Recursos Materiales***: todo tipo de insumos (papel, cintas de embalaje, cartuchos de tinta).

***Recursos Financieros***: dinero, cheques.

***Recursos Tecnológicos***: computadoras, calculadoras, agendas electrónicas.

## Toma de decisiones

Uno de los conceptos importantes que se destaca entre las funciones de la información es el de "Toma de Decisiones", ya que la TD es la estrategia de solución a un problema; mediante la elección de una alternativa entre varias posibles. En el proceso de toma de decisión es muy importante el rol que tiene la información ya que puede reducir el número de alternativas, o clarificar un poco más los beneficios y riesgos de cada una, para el tomador de decisiones.

Tomar una decisión supone escoger la mejor alternativa entre las posibles, se necesita información sobre cada una de estas alternativas y sus consecuencias respecto a nuestro objetivo ya que la realización de la acción elegida genera nueva información que se integrará a la información existente para servir de base a una nueva decisión, origen de una nueva acción y así sucesivamente tal como lo muestra la Figura (5)

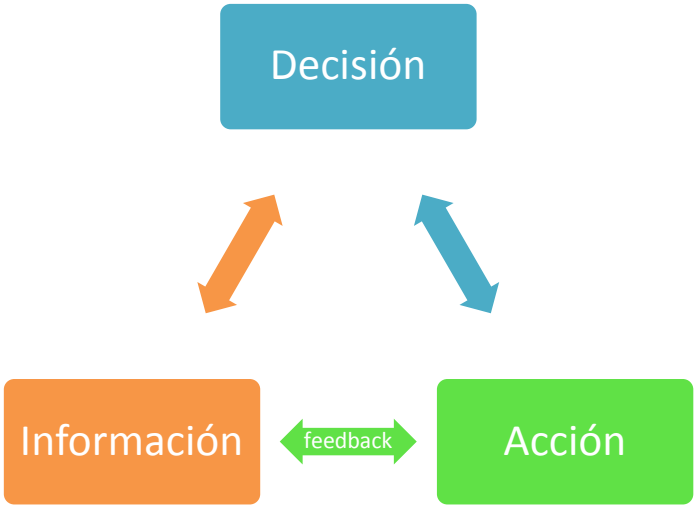



Figura (5)

El proceso de la toma de decisiones en una organización comienza con la detección de una situación que rodea algún problema. Seguidamente viene el análisis y la definición del problema.

Las decisiones se clasifican en función de la posición jerárquica o nivel administrativo ocupado por el decisor. Desde este planteamiento distinguiremos:

1. *Decisiones estratégicas (o de planificación)*: son decisiones adoptadas por decisores situados en la cúspide de la

pirámide jerárquica o altos directivos. Por ejemplo: decisiones relativas a dónde se deben localizar las plantas productivas, cuáles deben ser los recursos de capital y qué clase de productos se deben fabricar.

2. *Decisiones tácticas o de pilotaje*: Son decisiones tomadas por directivos intermedios. Tratan de asignar eficientemente los recursos disponibles para alcanzar los objetivos fijados a nivel estratégico. Por ejemplo: decisiones relacionadas con la distribución del presupuesto o la planificación de la producción.

3. *Decisiones operativas*: adoptadas por quienes se sitúan en el nivel más inferior. Son las relacionadas con las actividades rutinarias de la empresa.

Si consideramos a la información como un recurso fundamental tanto para la operación de las tareas de la organización, como para la toma de decisiones, debemos obtener la máxima utilidad de la misma, para ello debe ser administrada de forma correcta, teniendo en cuenta la producción, distribución, seguridad, almacenamiento y recuperación de la información como ocurre con cualquier otro recurso de la organización.

En conclusión, podemos afirmar que se requiere contar con un sistema de información confiable, oportuno y actualizado, que permita comprender claramente la naturaleza del problema a resolver ya que los sistemas de información se desarrollan para permitir la gestión y administración de toda la información necesaria para el correcto desarrollo de las actividades de una organización. De hecho, en muchas ocasiones el objetivo de un Sistema de información (SI) es apoyar a las actividades de una organización.

Ahora, si además dicho sistema emplea computadoras, entonces se lo clasifica como sistema informático.

## Análisis de un caso práctico: Panadería Gonzales S.A.

La panadería “Gonzales” se dedica a la manufactura y venta minorista de productos de panadería, y además realiza reventa de otros productos tales como lácteos, harina, azúcar, etc. La estructura de la empresa es la siguiente: | Gerencia: se encarga de controlar todas las actividades efectuadas en la empresa y realizar la toma de decisiones referidas al manejo general de los recursos, la distribución de tareas y responsabilidades y la apertura de una nueva sucursal o el lanzamiento de un nuevo producto.

Además, el gerente se encarga de realizar planes estratégicos a largo plazo para llevar a cabo las decisiones tomadas. | Producción: realiza la fabricación de los productos, la toma de decisiones referidas a las cantidades a producir y la asignación

de recursos humanos y materiales para la producción, coordina la mano de obra, instalaciones, herramientas, se encarga del almacenamiento y control de materias primas y productos de reventa, y además entrega los productos terminados a comercialización.

Este departamento consta de dos áreas:

- *Fabricación*: dentro de ella se realiza la manufactura de los productos, operan 12 personas, que trabajan 8 hs. aproximadamente, en turnos rotativos, durante las 24 hs.
- *Control de Calidad*: efectúa la verificación de la calidad de los productos terminados, comparándolos con límites preestablecidos. Esta actividad la realiza uno de los socios Gerentes.
- *Comercialización*: efectúa la venta en mostrador y distribución de los productos al cliente. Además, desde esta área se realiza la promoción y publicidad de nuevos productos, estudios de mercado y se toman decisiones referidas a los precios de los productos y la asignación de los turnos del personal del área. Consta de dos áreas:
- *Ventas*: tiene 7 personas en los puestos de venta.
- *Distribución*: realiza el traslado de los productos elaborados desde la casa Central a los clientes mayoristas.
- *Administración*: realiza todo lo relativo al manejo de los fondos de la empresa, como así también lleva el registro de toda la documentación de las actividades de la empresa. De ella dependen:
- *Cuentas Corrientes*: donde se lleva actualizado el saldo de las cuentas de los clientes.
- *Cobranzas*: donde se realiza el cobro de las facturas de los clientes.
- *Sueldos*: que se encarga de la registración de los datos de los empleados, control de asistencias, como así también la emisión de los recibos de sueldos y el pago de los mismos.
- *Compras*: efectúa las actividades relativas al reabastecimiento de materias primas y productos de reventa cuando se detectan faltantes en el stock o cuando Producción le solicita alguna cantidad de materias primas que va a necesitar para fabricar los productos. Para ello se realizan las tareas de selección de proveedores y emisión de las órdenes de compras a los mismos. Para la administración contable la empresa contrata los servicios de una Asesoría Contable que se encarga del control impositivo y de verificar la documentación de las transacciones efectuadas.

En la Organización se está realizando un plan para el lanzamiento de una nueva línea de bizcochos dulces, para ello, se han planteado una serie de actividades a realizar en el área de producción.

### ¿Qué tipo de sistema es y qué decisiones se toman?

Se trata de un Sistema Decisional (**DSS**) en donde se toman decisiones referidas al manejo general de los recursos, la distribución de tareas y responsabilidades y la apertura de una nueva sucursal o el lanzamiento de un nuevo producto.

En este caso, se necesitan tomar decisiones referidas a las cantidades a producir y la asignación de recursos humanos y materiales para la producción. Además, se necesita tomar decisiones referidas a los precios de los productos y la asignación de los turnos del personal del área de comercialización.

### ¿Qué sistemas componen a la organización?

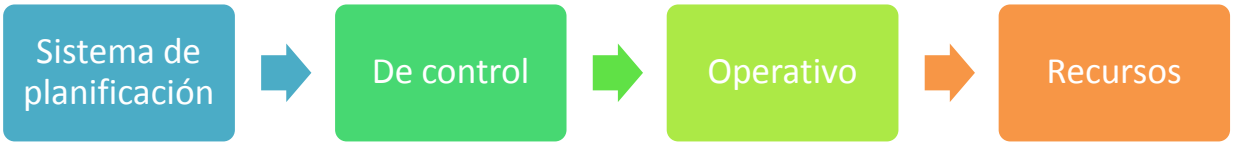



Figura (6)

La panadería necesita un Sistema de Planificación que se encargue de:

- Realizar planes estratégicos a largo plazo para llevar a cabo las decisiones tomadas referidas al manejo general de los recursos, la distribución de tareas y responsabilidades y la apertura de una nueva sucursal o el lanzamiento de un nuevo producto.
- Planificación de la producción y asignación de recursos a la misma.

Además, necesita un Sistema de Información que se encargue de:

- Realizar la registración de los planes y seguimiento de la producción.
- Registrar resultados de la producción.
- Registrar resultados de los controles de calidad.
- Actualizar stock.
- Registrar y consultar datos de los proveedores.
- Emitir órdenes de compra.
- Registrar la compra.
- Registrar pago a proveedores.
- Registrar y consultar clientes.
- Registrar la venta y emitir factura.
- Registrar cobro
- Actualizar cuentas corrientes de clientes.
- Registrar datos de los empleados.

- Registrar asistencias del personal.
- Emitir recibos de sueldo.
- Realizar registros contables.
- Emitir informes de producción.
- Emitir informes de compras.
- Emitir informes de ventas.
- Emitir informes de personal.
- Emitir informes contables.
- Como sistema de Control, la panadería necesita:
- Controlar y coordinar todas las actividades de la empresa.
- Controlar el avance de la producción.
- Realizar control de calidad de materias primas y productos terminados.
- Controlar las existencias de stock de materias primas y productos terminados.
- Realizar control sobre asistencia de empleados.
- Realizar controles administrativos y contables.

Como Sistema Operativo se necesita:

- Realizar la manufactura de los productos.
- Efectuar compra de materiales a proveedores.
- Almacenar materias primas y productos terminados y de reventa.
- Vender los productos.
- Realizar la distribución de los productos a los clientes.
- Realizar cobros a los clientes.
- Realizar el pago a proveedores.

Finalmente, respecto a los recursos, debemos tomar en cuenta:

- Recursos Humanos
  - Realizar reclutamiento de personal.
  - Realizar selección de personal.
- Recursos Financieros
  - Realizar solicitudes de préstamos.
- Recursos Tecnológicos
  - Realizar adquisición de máquinas amasadoras y mezcladoras.
  - Realizar adquisición de equipos de computación.
- Recursos Logísticos

- Realizar publicidad de los productos.
- Realizar estudios de mercado.

Para estudiar a fondo los sistemas que forman parte de una organización, puedes complementar tu estudio con la siguiente bibliografía:

Lectura recomendada para ampliar conceptos: Pungitore,J.L.(2007) “Sistemas Administrativos y Control Interno” Buenos Aires: Ed. Osmar Buyatti.

## Clasificación de los sistemas de información y servicios IT

Con el ejemplo práctico, vimos que existen diferentes tipos de sistemas que nos permiten llevar adelante el proceso de toma de decisión en forma correcta.

Vamos a hacer un recorrido por la historia de los primeros sistemas de información que surgieron. Así, podemos decir que los primeros SI fueron los TPS, en la década de los 60; que lograban la automatización de procesos operativos dentro de una organización y eran llamados frecuentemente Sistemas Transaccionales, ya que su función primordial consistía en procesar transacciones tales como pagos, cobros, pólizas, entradas, salidas, entre otras.

Entre los años 70 y 80 los Sistemas de Información cobran mayor interés en su papel como parte del proceso de toma de decisiones y de este modo surgen los DSS como Sistemas de Soporte a la Toma de Decisiones.

Finalmente, para nuestro abordaje, cabe mencionar los sistemas que de acuerdo con su uso u objetivo que cumple, se conocen como Sistemas Estratégicos, los cuales se desarrollan en las organizaciones con el fin de lograr ventajas competitivas, a través del uso de la tecnología de información.

Según la función a la que vayan destinados o el tipo de usuario final del mismo, los SI pueden clasificarse según muestra la (figura 7).

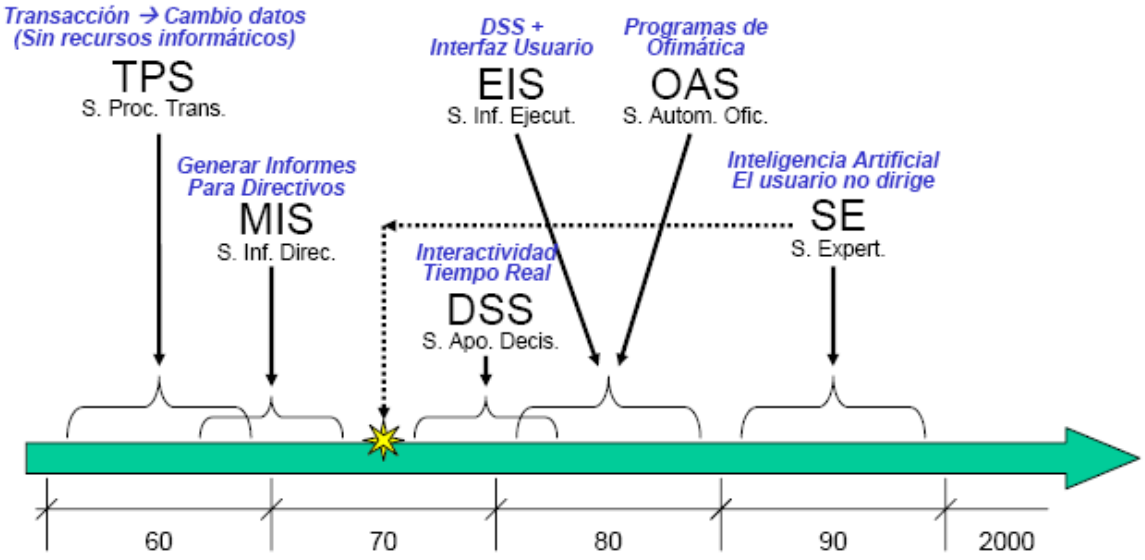



Figura (7)

- *Sistema de procesamiento de transacciones* (**TPS**). - Gestiona la información referente a las transacciones producidas en una empresa u organización.
- *Sistemas de información gerencial* (**MIS**). - Orientados a solucionar problemas empresariales en general.
- *Sistemas de soporte a decisiones* (**DSS**). - Herramienta para realizar el análisis de las diferentes variables de negocio con la finalidad de apoyar el proceso de toma de decisiones.
- *Sistemas de información ejecutiva* (**EIS**). - Herramienta orientada a usuarios de nivel gerencial, que permite monitorizar el estado de las variables de un área o unidad de la empresa a partir de información interna y externa a la misma.
- *Sistemas de automatización de oficinas* (**OAS**). - Aplicaciones destinadas a ayudar al trabajo diario del administrativo de una empresa u organización.
- *Sistema experto* (**SE**). - Estos sistemas, mejor conocidos como sistemas de inteligencia artificial, se basan en utilizar la experiencia y el conocimiento para analizar variables y hallar una solución a un problema dado. Ejemplo de ellos, los simuladores y las computadoras de ajedrez.

*"Los sistemas expertos conforman una clase muy especial de sistemas de información que se ha puesto a disposición de usuarios de negocio gracias a la amplia disponibilidad de hardware y software como computadoras personales y generadores de sistemas expertos." (Kendall y Kendall;2005)*

- *Sistema Planificación de Recursos* (**ERP**). - Integran la información y los procesos de una organización en un solo sistema.

Los sistemas de información mencionados, los podemos ubicar de acuerdo con la clasificación de la información como se muestra a continuación. (Figura 8)

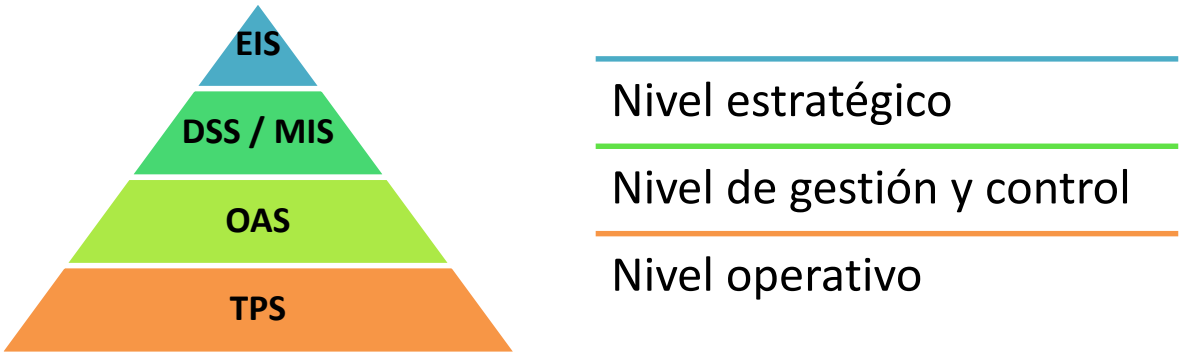



Figura (8)

## Servicios IT (Tecnología de la información)

Es un nuevo modelo de trabajo utilizado por muchas empresas, ya que los servicios IT son herramientas que se utilizan para el almacenamiento, gestión o visionado de la información que vamos a utilizar tanto a nivel personal como en los procesos de negocio. Por ejemplo: Correo electrónico, motores de búsqueda, servicios en la nube, audio o música, almacenamiento de archivos e incluso compras electrónicas.

La evolución de los servicios IT surge del crecimiento de las comunicaciones y del aumento en la demanda de recursos como terminales, redes y servicios. Los servicios IT hacen referencia al almacenamiento y gestión de la información como un recurso de valor para el cliente o negocio. Un servicio IT es la entrega de valor al cliente. (Figura 9)

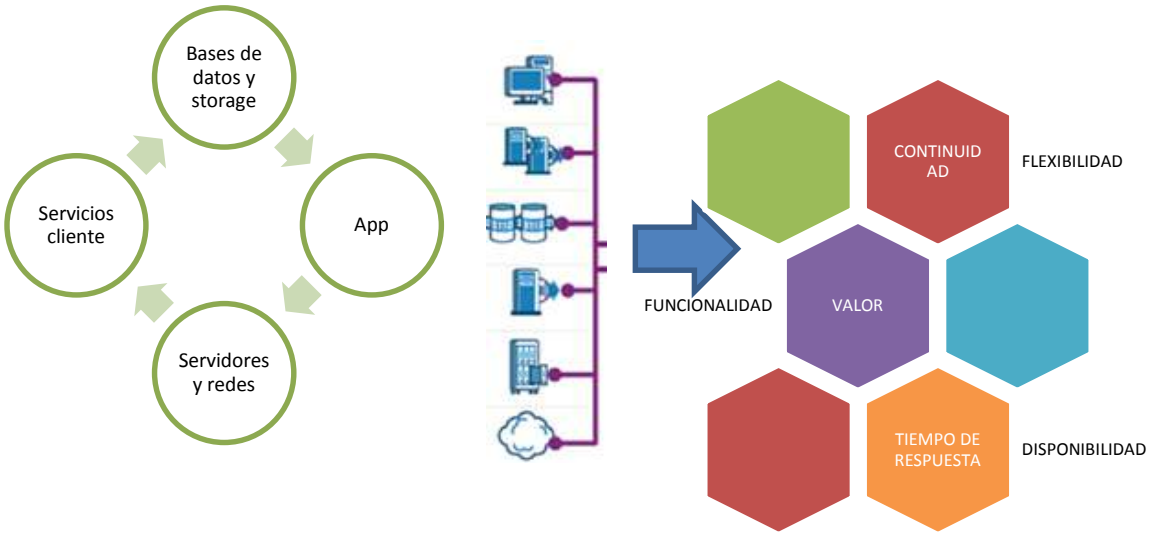



Figura (9)

La gestión de servicios IT se enfoca en el soporte y la entrega de servicios de tecnologías de la información. Es un factor importante para el logro de ventajas competitivas.

"El enfoque a través de un sistema de gestión de tecnologías de la información (SGTI) anima a las organizaciones a analizar los requisitos del cliente, definir los procesos que contribuyen al logro de loa servicios aceptables para el cliente y a mantener estos procesos bajo control."

Para apoyar la gestión IT existen muchos modelos de referencia. El primer modelo llamado ITIL (IT Infrastructure library) fue formulado por el gobierno británico a finales de los 80 y fue adoptado por grandes empresas como HP que desarrollo su modelo HPITSM, IBM con TI modelo de proceso y Microsoft con (MOF)entre muchos otros.

En general los modelos proveen un conjunto de mejores prácticas, agrupadas por disciplinas; por ejemplo: Soporte de servicios, entrega del servicio, gestión de seguridad, gestión de la infraestructura, gestión de aplicaciones, planificación de la gestión de servicios IT y perspectiva del negocio. (Figura 10)

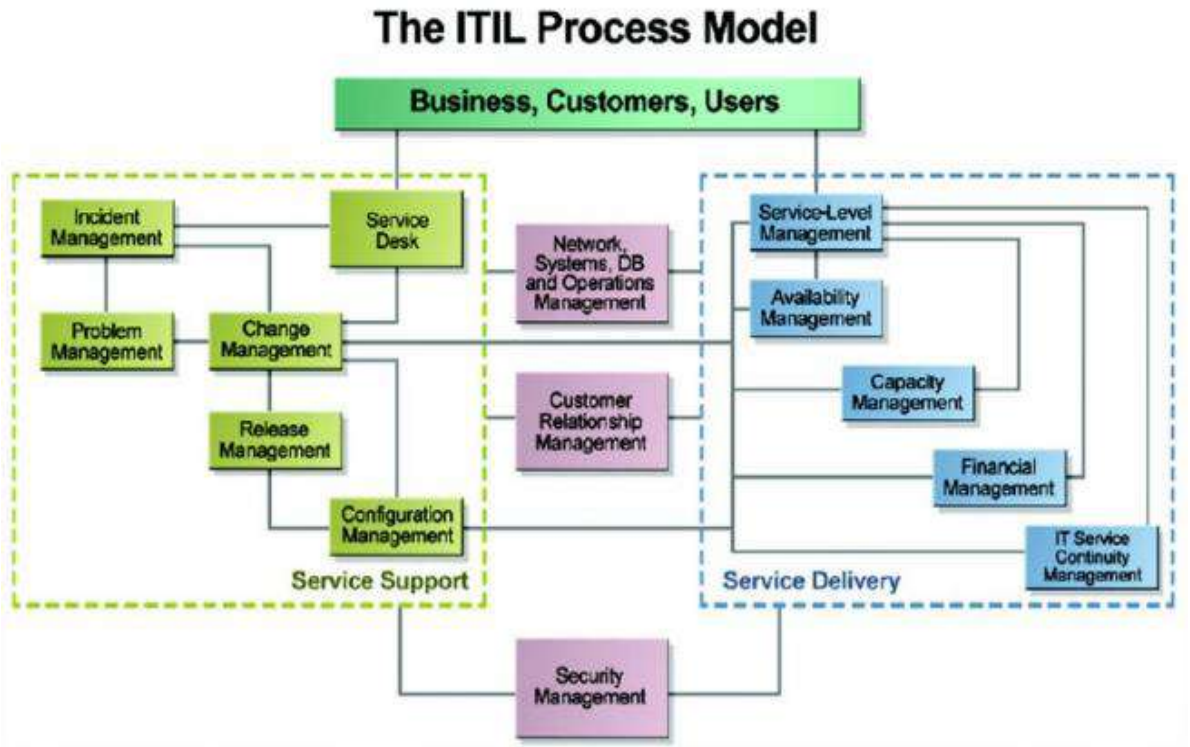



Figura (10)

ITIL comprende 5 fases diferenciadas en el ciclo de vida del servicio (Figura 11):


1.Estrategia de servicio
2.Diseño del servicio
3.Transición del servicio
4.Operación del servicio
5.Mejora continua del servicio

Figura (11)

En cuanto a las herramientas para la gestión IT, existen guías de buenas prácticas como ISO 20000, COBIT, ITL y otros estándares que permiten dar criterios para las buenas prácticas y la correcta prestación de servicios de TI. En resumen; las Normas ITIL garantizan el cumplimiento de TI al incluir un amplio rango de protocolos para monitorear y mejorar el ciclo de vida útil del servicio; y los cuales anticipan la inclusión de nuevos software y herramientas especializadas. Si estructuramos todas las herramientas y normas dentro de una organización, podemos desarrollar una pirámide como se muestra a continuación (Figura 12)

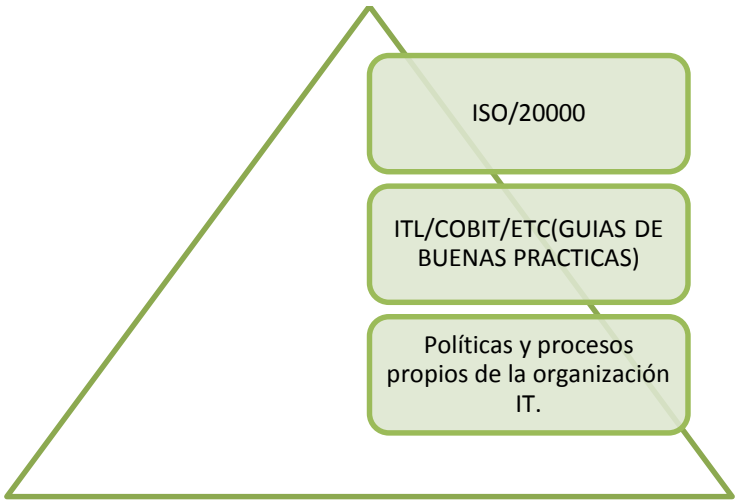



Figura (12)

Como conclusión, podemos afirmar que existen múltiples servicios IT muy útiles para la mayoría de las empresas, que no solo ayudan a abrir nuevos procesos de negocio y formas de trabajar, sino que además requieren una inversión muy baja lo cual los convierte en una oportunidad de inversión.

### Conclusiones

Cuando estudiamos el concepto de sistemas y todos los conceptos relacionados a él, pudimos determinar que para definir y conocer un sistema es necesario definir su objetivo. El objetivo es la meta o fin que quiere alcanzar un sistema.

Los sistemas de información, como todo sistema, tienen un objetivo principal que es el de brindar información; no obstante; luego de analizar el concepto de tecnología de la información; podemos agregar que un sistema de información también tiene como objetivo brindar servicios a una persona u organización.

Cuando hablamos de sistema de información, resulta obligatorio comprender a fondo de qué se trata este aspecto ya que, en la tarea de análisis y desarrollo de un sistema de información como solución informática, el objetivo del SI está vinculado directamente al estudio de requerimientos.

Para ello, es necesario realizar un estudio de los requerimientos de información que se presentan en cada organización en particular tomando en cuenta, además, el proceso de toma decisiones.

Lectura recomendada para ampliar conceptos: Kinectic Delivery Value "Gestión de servicios IT". Córdoba.

## Metodologías de sistemas

¿Cómo empezamos a desarrollar un sistema? ¿Qué es lo primero que tenemos que hacer? ¿Cómo representar la idea? ¿Cuántas personas son suficientes? Son preguntas que surgen espontáneamente y nos hacen reflexionar acerca del **COMO** hacer las cosas siguiendo un método que nos asegure el éxito.

De eso se trata la metodología; estudiar un sistema mediante el enfoque sistémico y atendiendo al planteo del tipo de problema a resolver en donde se conjugan personas; recursos físicos, materiales y tiempos; se desprende la imperiosa tarea de establecer un método de abordaje para el desarrollo de un Sistemas y no fracasar en el intento.

Es así como la metodología de sistemas es un mecanismo o método que provee las herramientas y procedimientos confiables y repetibles que se adecuan particularmente bien a los problemas que se pretenden resolver y el producto que se quiere desarrollar.

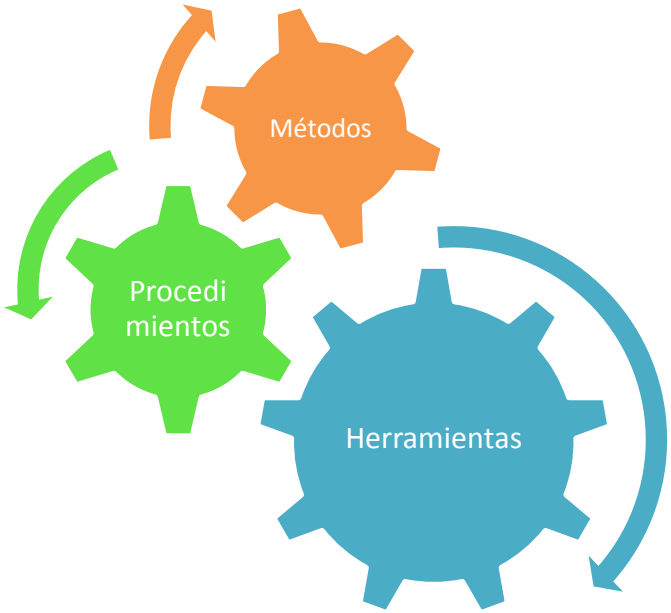



Figura (13)

Vamos a analizar cada uno de los conceptos enunciados en esta definición, cuando hablamos de metodología (Figura 13):

- ***Métodos***: indican cómo construir técnicamente el sistema. Abarcan un amplio espectro de tareas y actividades como: planificación, estimación de esfuerzo, análisis de los requisitos del sistema y el producto de software, diseño de modelos que definan y especifiquen estructuras de datos, arquitectura de programas y procedimientos algorítmicos sujetos a la, codificación, prueba y mantenimiento de los mismos. Introducen frecuentemente una notación especial orientada al lenguaje y un conjunto de criterios para lograr calidad.

- ***Herramientas***: Suministran un soporte automático o semiautomático para los métodos. Combinación de hardware, y software: herramientas de gestión como Trello; Asana y herramientas de modelado como Star UML, Rationla Rose, entre otras.

- ***Procedimientos***: Relaciona métodos y herramientas. Definen la secuencia en la que se aplican los métodos, las entregas que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios, y directrices que ayudan a los gestores del proyecto a evaluar el progreso.

Al igual que la evolución de los paradigmas de programación, las metodologías de sistemas se adaptaron a este cambio y surgieron 2 grandes grupos de referencia para encuadrar las metodologías: entorno tradicional y entorno ágil. A pesar de ello, existe un conjunto de características generales que definen el ciclo de vida típico de un sistema que se representa con 4 etapas. (Figura 14)

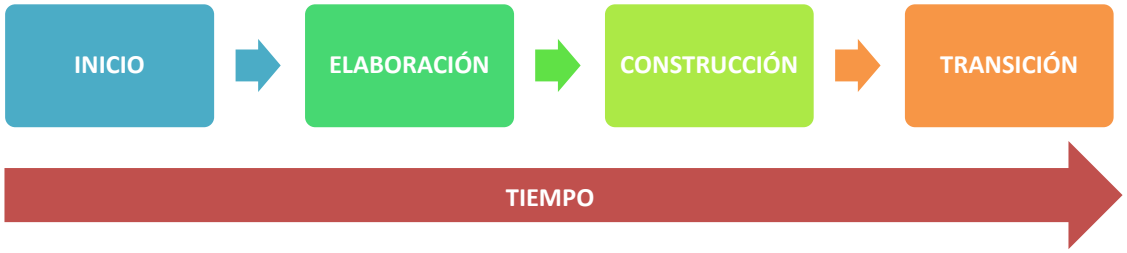



Figura (14)

1. ***Inicio***: se define la "visión del sistema", se establece el alcance del proyecto y se toma la decisión de comenzar con el mismo, es decir, se decide realizar la inversión de dinero y esfuerzo para analizar en detalle el sistema a construir. En el caso de tratarse de la modificación o del agregado de funcionalidad a un sistema existente, esta fase puede ser corta y sencilla, basada en los pedidos de los usuarios, en reportes de problemas o en la necesidad de incorporar avances tecnológicos.

2. ***Elaboración***: su propósito, en primer lugar, es analizar el sistema de negocio para el cual se busca una solución. En segundo término, definir la estructura preliminar del sistema y en tercera instancia identificar los factores de riesgo del proyecto y, por último, elaborar un plan detallado del mismo.

3. ***Construcción***: consiste en la fabricación del sistema y de los productos de apoyo necesarios, tales como documentación y los casos de prueba del sistema. En esta fase también se expanden y revisan los productos o resultados obtenidos en las fases anteriores.

4. ***Transición***: es aquella en la cual el sistema se entrega a los usuarios. Esta fase incluye actividades de instalación, configuración, soporte a los usuarios, correcciones, etc. Y finaliza cuando los usuarios están satisfechos con el sistema, lo cual suele implicar una aceptación formal por parte de estos.

## Metodología tradicional

Las metodologías tradicionales o PREDICTIVAS o PESADAS buscan siempre una fuerte planificación y documentación durante todo el desarrollo, y las metodologías ágiles, en las que se enfoca al desarrollo de software el cual es incremental, cooperativo, sencillo y adaptado.

Una de las principales características de las metodologías tradicionales es que se sustentan en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto estricto y cerrado ante alguna eventualidad. Por ende, resultan de implementación costosa frente a los cambios en un proyecto; lo que actualmente no acompaña al entorno cambiante en donde nos encontramos inmersos.

Las metodologías tradicionales además de la exhaustiva documentación, centran su atención en la planificación por adelantado y la gestión de los procesos; mayormente lineales.

Aunque parecería que la metodología tradicional no tiene beneficios, debemos reconocerle que se adapta perfectamente cuando se presentan objetivos claramente definidos desde el principio del proyecto, permite controlar exhaustivamente los procesos y sirve para proyectos a largo plazo, es decir; aquellos casos en donde no tenemos restricciones de tiempo. En este escenario el enfoque tradicional nos permite desarrollar una documentación clara; como también; permite disponer de recursos humanos sin alto grado de experiencia.

Existen muchos diferentes procesos de software, pero todos deben incluir cuatro actividades que son fundamentales para la ingeniería de software:

1. Especificación de requerimientos del software: tienen que definirse tanto la funcionalidad del software como las restricciones de su operación.

2. Diseño e implementación del software: en algunos casos se representa mediante un modelo tal que el software debe cumplir con las especificaciones del modelo y su refinamiento.

3. Validación del software: Hay que validar el software para asegurarse de que cumple lo que el cliente quiere o bien que está de acuerdo con la especificación de requerimientos.

4. Evolución del software El software tiene que evolucionar para satisfacer las necesidades cambiantes del cliente y del negocio.

### Modelo clásico en cascada

Es un modelo de base para el estudio de otros modelos. Fue modelado a partir de un enfoque sistemático y secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. A continuación, se describen las actividades que comprende (Figura 15):

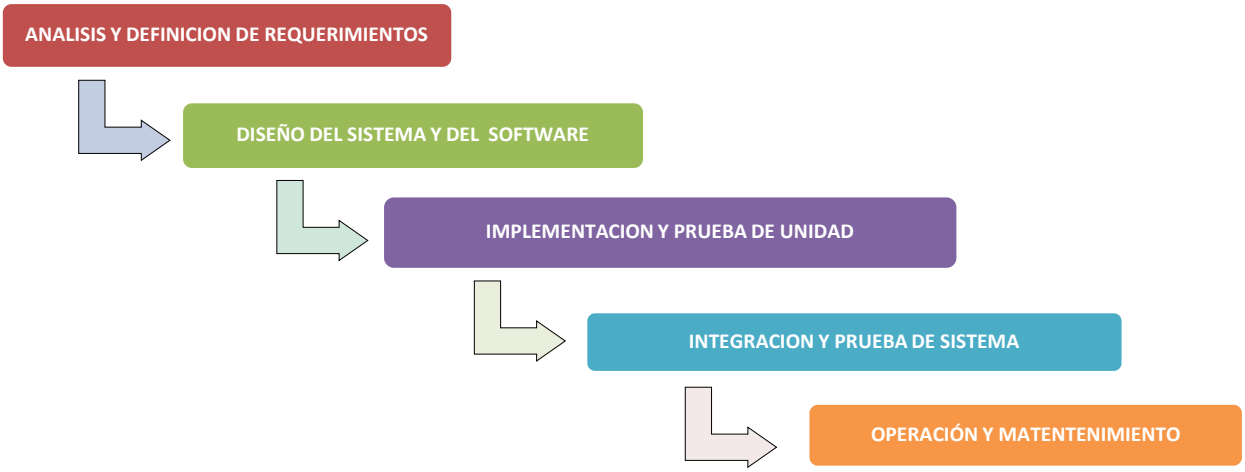



Figura (15)

1. ***Análisis y definición de requerimientos***: Los servicios, las restricciones y el propósito del sistema se establecen y sirven como especificación del sistema.
2. ***Diseño del sistema y del software***: se establece una arquitectura de sistema global. El diseño del software implica identificar y describir las abstracciones fundamentales del sistema de software.
3. ***Implementación y prueba de unidad***: durante esta etapa, el diseño de software se realiza como un conjunto de programas o unidades del programa o componentes. La prueba de unidad consiste en verificar que cada unidad cumpla con su especificación.
4. ***Integración y prueba de sistema***: Las unidades del programa o los programas individuales se integran y prueban como un sistema completo para asegurarse de que se cumplan los requerimientos y restricciones del software. Después de probarlo, se libera el sistema de software al cliente.
5. ***Operación y mantenimiento***: Es la fase más larga del ciclo de vida, donde el sistema se instala y se pone en práctica. El mantenimiento incluye corregir los errores que no se detectaron en etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema e incrementar los servicios del sistema conforme se descubren nuevos requerimientos.

En principio, el resultado de cada fase consiste en uno o más documentos que resultan de un plan de trabajo. La siguiente fase no debe comenzar sino hasta que termine la fase previa y se nutren mutuamente de información muy controlada.

Durante el diseño se identifican los problemas con los requerimientos. En la codificación se descubren problemas de diseño, y así sucesivamente. El proceso de software implica retroalimentación de una fase a otra. Entonces, es posible que los documentos generados en cada fase deban modificarse para reflejar los cambios que se realizan.

*"Debido a los costos de producción y aprobación de documentos, las iteraciones suelen ser onerosas e implicar un rediseño significativo. Por lo tanto, después de un pequeño número de iteraciones, es normal detener partes del desarrollo, como la especificación, y continuar con etapas de desarrollo posteriores." (Soomerville,2009)*

Durante la fase final del ciclo de vida (operación y mantenimiento), el software se pone en servicio. Se descubren los errores y las omisiones en los requerimientos originales del software. Surgen los errores de programa y diseño, y se detecta la necesidad de nueva funcionalidad. Por lo tanto, el sistema debe evolucionar para mantenerse útil.

Las desventajas del modelo en cascada pura se centran en la dificultad para especificar claramente los requerimientos al comienzo del proyecto, antes de que se realice ningún trabajo de diseño y antes de escribir ningún código.

Una gran cantidad de productos software son complicados, y a menudo las personas que se encargan de especificar el software tampoco son expertos. Pueden olvidarse de cosas que parecen muy sencillas cuando se ve el producto funcionando. Cuando se utiliza un modelo en cascada, olvidar algo puede suponer un error costoso.

Por tanto, el principal problema del modelo en cascada es no permitir flexibilidad en los cambios. Se tienen que especificar completamente todos los requerimientos al comienzo del proyecto, lo que puede suponer meses o incluso años antes de tener el software funcionando. Esto no resulta acorde con las necesidades del comercio actual, donde los desarrolladores necesitan implementar la máxima funcionalidad en las últimas etapas del proyecto.

Además, presupone que el producto está perfectamente definido antes de iniciar el desarrollo tal que cuando se descubren problemas en la fase de mantenimiento sólo cabe adaptar el problema a la aplicación, y no al revés.

Por lo tanto, es un modelo rígido y poco flexible donde cometer algún error en alguna de sus etapas hace difícil volver hacia atrás con un efecto negativo en el costo y en el tiempo.

### Modelo de Prototipo

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica información detallada; esto hace que el programador no tenga certeza de la eficiencia de un algoritmo o de las necesidades reales del cliente. En estas y muchas otras situaciones puede ser el mejor método la construcción de un prototipo.

Un prototipo facilita al programador, la tarea de creación de un modelo del software, aunque no sea tan funcional como el producto final. (Figura 16)

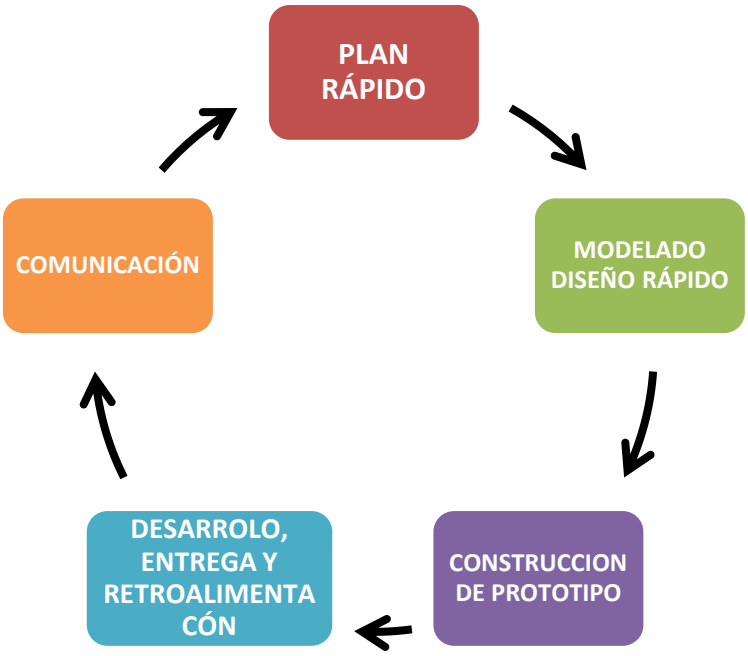



Figura (16) - Modelado de construcción de prototipos

Las etapas en la construcción de prototipos son: un plan rápido para la recolección y refinación de los requisitos / Diseño rápido / Construcción del prototipo / Evaluación del prototipo por el cliente: desarrollo, entrega y retroalimentación del prototipo / Comunicación.

La construcción de prototipos puede ser problemática cunado el cliente ve funcionando lo que parece ser una primera versión del software que no reúne todos los aspectos de calidad o de mantenimiento de software a largo plazo.

Es necesario definir al comienzo las reglas de juego, el cliente y el técnico deben estar de acuerdo en que el prototipo se construyó para servir sólo como un mecanismo de definición de los requisitos. Posteriormente debe construirse un sistema real con calidad y mantenimiento.

Este modelo permite la modificación del sistema en etapas tempranas de su desarrollo; no obstante, el éxito del uso del prototipo depende de qué tan pronto y con qué frecuencia se reciba la retroalimentación del usuario para hacer cambios y adecuarlos a las necesidades actuales. Y fundamentalmente, permite al desarrollador verificar los requerimientos del cliente con mayor comprensión.

Los cambios iníciales durante el desarrollo de un proyecto son menos costosos que si se realizan en etapas tardías, como el prototipo puede cambiar varias veces la flexibilidad y adaptabilidad son su esencia, la pauta del cambio permite que exista retroalimentación de la opinión del usuario sobre cambios a la entrada o salida

de un proceso, que al evaluarla nos permite obtener los requerimientos y mejorar el sistema.

Como desventaja, presenta una funcionalidad limitada, poca fiabilidad y características pobres a nivel de operación; en algunos casos se gestiona el prototipo como un "proyecto" en sí mismo perdiendo de vista su verdadero propósito.

Al adoptar el PROTOTIPO como el sistema final, los usuarios y profesionales de sistemas pueden considerar al prototipo como el sistema final cuando aún está incompleto.

### Modelo en espiral

Cubre las mejores características tanto del ciclo de vida Clásico como la construcción de prototipos agregando el análisis de riesgo, que falta en esos ciclos de vida. Las etapas se muestran en la figura a continuación. (Figura 17):

- ***Planificación***: determinación de objetivos, alternativas y restricciones.
- ***Análisis de riesgo***: análisis de las alternativas e identificación/resolución de riesgos.
- ***Implementación***: se desarrolla el producto de "siguiente nivel".
- ***Evaluación por el cliente***: valoración de los resultados de la ingeniería por parte del cliente.

Con cada iteración alrededor de la espiral se construyen sucesivas versiones del software cada vez más completas. El objetivo final del modelo espiral es que en cada vuelta se construirá una versión más nueva y completa del programa.

El modelo en espiral es actualmente el enfoque más realista para el desarrollo de software y sistemas a gran escala. El modelo en sí es relativamente nuevo y no se ha usado tanto como el ciclo de vida clásico o la creación de prototipos.

El modelo espiral es un modelo de ciclo de vida orientado a riesgos que divide un proyecto software en mini proyectos. Cada mini proyecto se centra en uno o más riesgos importantes hasta que todos éstos estén controlados. El concepto de riesgo se define ampliamente en este contexto, y puede referirse a requerimientos poco comprensibles, arquitecturas poco comprensibles, problemas de ejecución importantes, problemas con la tecnología subyacente, y demás.

Después de controlar todos los riesgos más importantes, el modelo en espiral finaliza del mismo modo que el modelo de ciclo de vida en cascada.

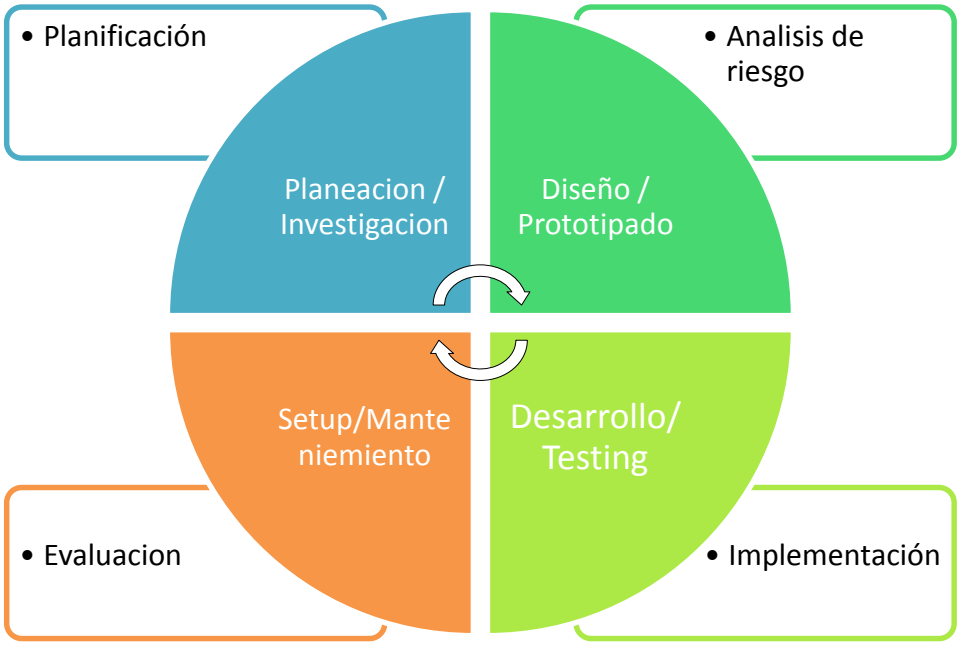



Figura (17)

La idea básica que subyace en el diagrama que representa a este ciclo de vida es que se parte de una escala pequeña en medio de la espiral, se localizan los riesgos, se genera un plan para manejar los riesgos, y a continuación se establece una aproximación a la siguiente iteración. Cada iteración supone que el proyecto pasa a una escala superior, se comprueba que se tiene lo que se desea, y después se comienza a trabajar en el siguiente nivel.

En el modelo en espiral, las primeras iteraciones son las menos costosas. Supone menos gasto desarrollar el concepto de operación que realizar el desarrollo de los requerimientos, y también es menos costosos desarrollar los requerimientos que llevar a cabo el desarrollo del diseño, la implementación del producto y la prueba de este.

El significado del diagrama no tiene por qué seguirse de forma literal. **No es importante que la espiral tenga exactamente cuatro ciclos**, y no es importante tampoco que se realicen exactamente los seis pasos como se indica, aunque se trata de un orden apropiado a utilizar. Se puede adaptar cada iteración de la espiral a las necesidades que demanda el proyecto.

El modelo se puede combinar con otros modelos de ciclo de vida de dos formas distintas. Se puede comenzar un proyecto con una serie de iteraciones para reducir los riesgos, después de que se haya reducido los riesgos a un nivel aceptable, se puede finalizar el esfuerzo de desarrollo con un ciclo de vida en cascada u otro modelo de ciclo de vida no basado en riesgos. Usted puede incorporar otros modelos de ciclo de vida como iteraciones dentro del modelo en espiral. Por ejemplo, si uno

de los riesgos consiste en que no se tiene seguridad de alcanzar los objetivos de rendimiento, puede incluir una iteración de prototipado para investigar si es posible la consecución de estos objetivos.

Una de las ventajas más importantes del modelo en espiral es que mientras los costos suben, los riesgos disminuyen. Cuando más tiempo y dinero emplee, menores serán los riesgos, que es exactamente lo que se quiere en un proyecto de desarrollo rápido.

El modelo en espiral proporciona al menos tanto control de gestión como el modelo en cascada tradicional. Se tienen los puntos de verificación al final de cada iteración. Como el modelo está orientado a riesgo insuperable, descubrirá si el proyecto no se puede realizar por razones técnicas u otras razones, y esto no supondrá un costo excesivo.

La ventaja principal que ofrece este modelo es que intenta eliminar errores en las fases tempranas, es decir al comienzo del proyecto donde hay mucha incertidumbre con su posible efecto en los costos. Además, el riesgo de sufrir algún retraso es muy bajo ya que se puede solucionar en la próxima espiral. También provee mecanismos para asegurar la calidad del software frente a proyectos complejos, dinámicos e innovadores donde la evaluación después de cada fase permite cambios en las percepciones de los usuarios; avances tecnológicos y amplia las perspectivas económicas. Pone foco en los objetivos y restricciones lo que ayuda a asegurar la calidad.

Entre algunas de sus desventajas, aparece en primer lugar: la dificultad para evaluar el riesgo; por otro lado, el costo de implementar cada espiral; finalmente; podemos agregar que requiere la presencia o comunicación continua con el cliente.

Se trata de un modelo complicado que requiere de una gestión atenta y que exige conocimientos profundos. Puede ser difícil definir hitos/objetivos de comprobación que indiquen si está preparado para pasar al siguiente nivel de la espiral.

## Metodología Ágil

**Agile** o metodologías ágiles es un enfoque general utilizado para el desarrollo de software en sus inicios, aunque adaptado a muchos otros sectores, se basa en gran medida en el trabajo en equipo, la colaboración, las tareas y la flexibilidad para responder al cambio lo más rápido posible. **Además**, sigue un proceso iterativo en el que los proyectos se dividen en Sprints de menor duración.

A diferencia del enfoque tradicional, se gasta menos tiempo en la planificación y la priorización por adelantado, ya que el enfoque ágil es más flexible en cuanto a cambios respecto a los requerimientos iniciales. Un requerimiento nuevo representa una oportunidad y no un problema.

Las metodologías ágiles nacen como respuesta a los problemas que se presentan en las metodologías tradicionales y se basa en dos aspectos fundamentales: retrasar las decisiones y la planificación adaptativa.

- Las metodologías están basadas en adaptabilidad de los procesos de desarrollo.
- Las metodologías ágiles ven más importante la capacidad de adaptarse a los cambios que el seguir un plan estricto de desarrollo.

La sociedad está cambiando y estos cambios afectan a todos los ámbitos de la vida y la gestión de proyectos. Si bien históricamente se han utilizado metodologías tradicionales para gestionar el desarrollo de proyectos software, este tipo de metodologías siguen estando muy presentes cuando nos enfrentamos a proyectos de gran envergadura.

## Valores de la metodología ágil

Cabe destacar que la metodología Agile se centra más en personas que en procesos, además de en la adaptación, mutación y cambio para satisfacer al cliente.

1. Valoración de las personas y las relaciones sociales por encima de herramientas y procesos.
2. Colaboración directa con el cliente para mantener una relación más participativa y cercana.
3. Priorización funcional del producto por encima de la acumulación y exceso de documentación.
4. Responder de forma ágil y efectiva a los imprevistos y cambios que puedan haberse desarrollado en el plan inicial.

## Principios de la metodología ágil

Para poder aplicar este método es imprescindible ceñirse a estos 12 principios fundamentales:

1. Perseguir la satisfacción del cliente e informarle periódicamente del estado del proyecto.
2. Los nuevos cambios y requisitos son bienvenidos y se valoran como modificaciones positivas.
3. La división del trabajo se realiza en fases temporales productivas divididas en semanas, quincenas, etc.
4. Posibilidad de medir el progreso.
5. La forma de ejecutar los proyectos debe garantizar en sí misma la continuidad del proyecto (desarrollo sostenible).
6. El equipo debe trabajar de forma coordinada y en conjunto, utilizando el método Scrum, como una práctica efectiva y esencial para la correcta organización y desarrollo del trabajo.
7. Las conversaciones entre los integrantes del equipo y/o el cliente deben llevarse a cabo en persona, para comunicar de forma eficaz los mensajes.
8. Es necesario infundir motivación y confianza a los miembros que forman parte del proyecto para obtener procesos exitosos.
9. Excelencia técnica y buen diseño. En la metodología Agile, la calidad del trabajo y la presentación forman parte del conjunto.
10. Se impone la ley de la simplicidad. Las tareas deben ser lo más sencillas posible. En caso de no poder simplificarlas se tendrá que dividir en iteraciones para reducir su nivel de complejidad.
11. Equipos autogestionados. Aunque es necesario que exista una figura que monitorice los equipos de trabajo, éstos deben ser capaces de organizarse por sí mismos.
12. Adaptación a las circunstancias cambiantes. Es imprescindible que los profesionales que ejecuten los proyectos puedan adaptarse a las distintas circunstancias y modificaciones que puedan surgir durante el proceso.

Cada vez es mayor el uso de las metodologías agiles, entre las más conocidas figuran: Scrum, Kanban,Lean y XP.(Figura 18)

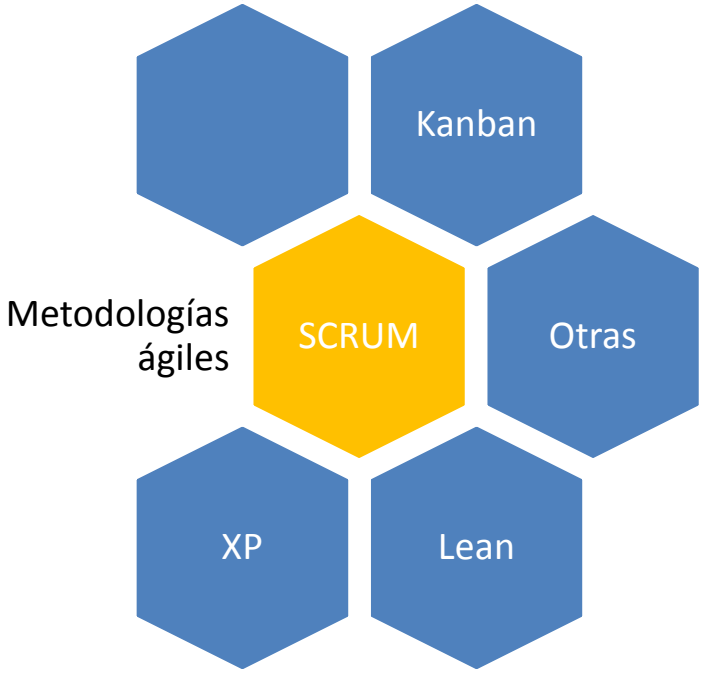



Figura (18)

### XP

La programación extrema (en adelante XP) puede que marque un antes y un después en la ingeniería del software. En este segundo punto del módulo, intentaremos mostrar todas sus ventajas y desventajas, así como su ambicioso punto de partida: el software como solución ágil y no como proyectos arquitectónicos.

Creada por Kent Beck, Ward Cunninghamn y Ron Jeffries a finales de los noventa, la programación extrema ha pasado de ser una simple idea para un único proyecto a inundar todas las "factorías de software". Algunos la definen como un movimiento "social" de los analistas del software hacia los hombres y mujeres de negocios, de lo que debería ser el desarrollo de soluciones en contraposición a los legalismos de los contratos de desarrollo.

Lectura recomendada para ampliar conceptos: Fernandez Gonzales,J (2009) “Introducción a las metodologías agiles y otras formas de analizar y desarrollar” España: FUOC.

### Lean

La premisa básica de lean startup es la de que un startup no es una empresa sino una organización temporal cuyo objetivo es encontrar un modelo de negocio viable y escalable mediante una serie de experimentos que sirven para aprender, y todo esto rodeado de una gran incertidumbre.

*"El movimiento Lean Startup, describe un producto mínimo viable como: la versión de un nuevo producto que permite a un equipo recolectar la cantidad máxima de aprendizaje validado sobre los clientes con el menor esfuerzo." (Eric Ries,2008)*

Esta metodología perfeccionada por Eric Ries en su libro El método Lean Startup, nace de la nueva realidad y las nuevas necesidades de las nuevas empresas en los últimos años. El método Lean Startup es compatible tanto con el Lean Canvas como el Business Model Canvas para diseñar modelos de negocio.

**Primera clave de Lean Startup: Ciclo de desarrollo vs ciclo de aprendizaje (construir-medir-aprender)**

Como otras metodologías modernas, Lean Startup se basa en un enfoque centrado en el cliente en vez de en el producto, de manera que se busca aprender de cada iteración de nuestro producto para poner a prueba nuestras hipótesis y poder de esta manera saber hacia dónde avanzar. (Figura 19)

Para conseguir validar nuestro aprendizaje es fundamental actuar con rapidez y no esperar a tener un producto perfectamente acabado, mientras antes podamos testear nuestras hipótesis con clientes de verdad, antes sabremos si estamos acertando o conviene rectificar. De esta manera nace el concepto de MVP, producto viable mínimo, que no es más que trabajar con una versión de nuestro producto con las características fundamentales y que funciona adecuadamente, con el objetivo de maximizar nuestro aprendizaje del negocio, producto y mercado.

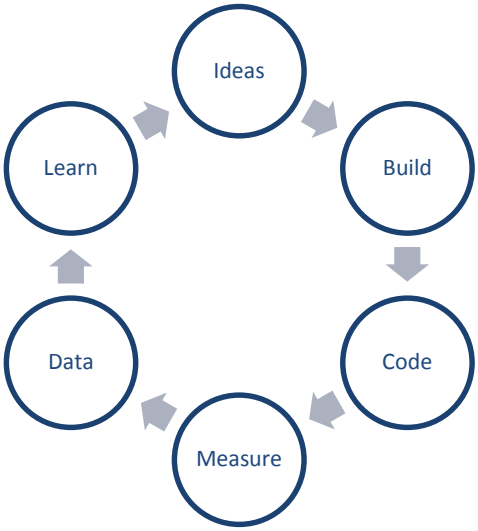



Figura (19)

Esto contrasta con el funcionamiento clásico del ciclo de desarrollo de productos tradicional, en el que el orden correcto pasaba por todas las fases de creación del producto hasta finalizar su desarrollo para finalmente lanzar el producto y esperar que hubiéramos acertado en nuestros planteamientos. La nueva forma de trabajar reduce este tremendo gasto ya que desde el primer minuto estamos trabajando con clientes de verdad obteniendo un conocimiento del negocio que de otra forma sería imposible.

Para esto Lean Startup cambia el clásico ciclo de desarrollo por el novedoso ciclo de aprendizaje, que se basa en 3 fases:

***Construir***: desarrollamos nuestro MVP centrado en las hipótesis que queremos comprobar.

***Medir***: establecemos una serie de métricas con las que valorar nuestro experimento.

***Aprender***: gracias a las métricas obtenemos información con la que aprenderemos nuevos detalles de nuestro negocio para seguir mejorando.

Este ciclo es iterativo, es decir, para cada hipótesis que queremos comprobar debemos crear un nuevo MVP, o una modificación, y lanzarlo para seguir aprendiendo. Se trata de una filosofía basada en la experimentación con ciclos de desarrollo muy cortos.

Segunda clave de Lean Startup: Perseverar o Pivotar tu modelo de negocio

Siguiendo la metodología Lean Startup de Eric Ries, otro de los objetivos de nuestros experimentos es de saber cuándo perseverar en la línea que llevamos o cuando pivotar el modelo de negocio cambiando alguna de sus premisas básicas. Esta información la podremos obtener con el aprendizaje que obtenemos de nuestros MVPs, pero la decisión última siempre será nuestra.

Con un grado de incertidumbre tan grande como en el que se mueve cualquier startup, es imprescindible ser muy flexible para saber adaptar nuestro negocio a las realidades del mercado, de nada sirve perseverar en nuestra magnífica idea de negocio sin no somos capaces de encontrar clientes que estén dispuestos a pagar.

Un ejemplo de la vida real de un producto mínimo viable es la forma en que Dropbox probó la viabilidad de su concepto de intercambio de archivos y sincronización en sus inicios. Antes de invertir millones en servidores basados en la nube, el equipo de Dropbox optó por crear una página de aterrizaje de video simple de tres minutos, diseñada para explicar la funcionalidad y características del servicio.

Solo 24 horas después del lanzamiento del video, las suscripciones al servicio de Dropbox aumentaron de 5,000 a 75,000 personas; todo esto sin desarrollar físicamente un producto.

El video sirvió como justificación suficiente para que el concepto tuviera el potencial de alcanzar un nivel funcional en el mundo real, un fantástico ejemplo de aprendizaje validado.

Lectura recomendada para ampliar conceptos: Moreno Martin, M.A “Filosofía de Lean aplicada a la ingeniería de software” España: Universidad de Sevilla

## Kanban

El sistema Kanban ideado por Ohno es relativamente sencillo. “Kan” significa visible o visual, y “ban”, tarjeta o tablón.

“Imaginemos una fábrica de bicicletas y que somos los encargados de las pastillas de freno. Contamos junto a nuestro puesto de trabajo con un stock de diez piezas que vamos ensamblando, y cuando nos queda la mitad, utilizamos un tablón para avisar de que vamos a necesitar otras diez pastillas de freno. Estas llegarán cuando nos quedemos sin las cinco que teníamos cuando colocamos el aviso.”

La gestión del inventario es justo la adecuada: nunca hay piezas ocupando espacio necesario para otras tareas y nunca se frena la producción por falta de material. El sistema se ajusta al menor o mayor número de pedidos entrantes. La clave del éxito del método es su capacidad de adaptación a un volumen de trabajo cambiante.

Si tomamos como referencia el conjunto de una fábrica, o también cualquier otro tipo de organización, como por ejemplo un equipo de desarrollo, el sistema Kanban se organiza con un gran tablón dividido en columnas, normalmente siete, aunque muchas veces se utilizan solo 3 columnas:

Objetivos: se marcan a largo plazo, con la idea de que todos los miembros del equipo los tengan en mente. Es una columna opcional, no siempre está presente.

Pendiente: esta columna engloba las tareas pendientes que se pueden afrontar de forma inmediata. En el lugar más alto de esa columna colocaremos la tarea pendiente que tiene la máxima prioridad, y en cuanto empecemos, la pasaremos a las siguientes columnas.

Preparación: también es opcional. Aquí incluimos aquellas tareas que necesitan cierta discusión interna antes de ser afrontadas. Cuando lo tengamos claro, pasamos a la siguiente columna.

Desarrollo: en este espacio situamos la tarea hasta que la terminemos. Si algo falla, regresa a la columna anterior.

Prueba: comprobamos que todo funciona bien. En función de ese examen, la tarea avanza en el tablón o retrocede.

Aplicación: la existencia de esta columna depende de las características de cada tarea. Hablamos, por ejemplo, de tareas como colocar una nueva versión de una aplicación en un servidor.

Hecho: cuando ya no tenemos que preocuparnos más de algo porque hemos terminado la tarea.

Este no es un sistema cerrado, ya que en casos de tareas urgentes se puede crear una fila específica -horizontal, por encima de todas las columnas- de 'urgente' donde solo puede estar una tarea, que irá recorriendo, en su propio carril especial superior, todas las columnas.

Otra forma de mejorar el procedimiento de gestión es establecer un número máximo de tareas para cada una de las columnas, en función del equipo de profesionales que tengamos.

Lectura recomendada para ampliar conceptos: Daniel Vacanti, y Yuval Yeret (2019) "La Guía Kanban para Equipos Scrum". Ed.Scrum.org

## Scrum

El primer equipo de Scrum lo creó Jeff Sutherland en Easel Corporation en 1993 y el marco de trabajo Scrum lo formalizó Ken Schwaber en 1995. Hoy en día Scrum es usado por empresas de todos los tamaños tales como Yahoo!, Microsoft, Google, Lockheed Martin, Motorola, SAP, Cisco, GE, CapitalOne y otros.

Scrum es un marco de trabajo compuesto de procesos que se ha utilizado para gestionar el trabajo de productos complejos mediante la mejora continua del producto, del equipo y del entorno de trabajo.

*"Scrum no es un proceso, una técnica, o método definitivo. Todo lo contrario, es un marco de trabajo donde se pueden emplear un conjunto de diferentes procesos y técnicas. "Ken Schwaber y Jeff Sutherland (2017)*

El marco de trabajo Scrum se compone por los Equipos Scrum, sus Roles, Eventos, Artefactos y Reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso.

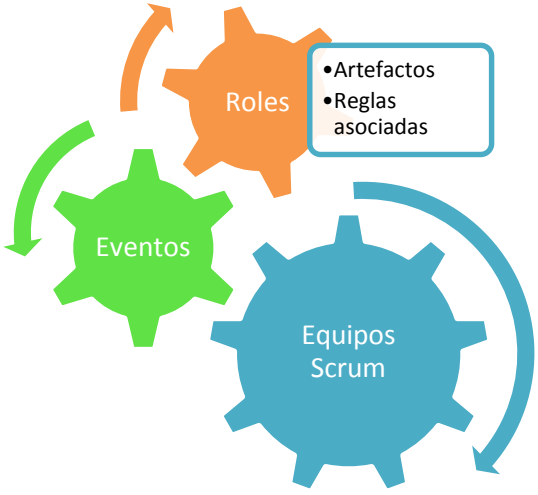



Figura (19)

Scrum fue desarrollado para gestionar y desarrollar productos, pero luego se hizo extensivo el uso de Scrum para investigar mercados, desarrollo y lanzamiento de productos; gestión en la nube e incluso para tareas de mantenimiento.

Todo este amplio marco de oportunidades que ofrece Scrum se debe a que ha demostrado ser un modelo muy efectivo en la transferencia de conocimiento y el trabajo en equipo como mecanismo para incrementar la productividad en forma incremental y continua.

La esencia de Scrum se sostiene en pequeños equipos de personas tal que el equipo individualmente es muy flexible y adaptivo.

## Marco teórico de Scrum

Scrum se basa en la teoría de control de procesos empírica que asegura que el conocimiento procede de la experiencia y en poder tomar decisiones basándose en lo conocido para ello utiliza un enfoque incremental en periodos cortos de tiempo que optimizan la predictibilidad y controlan el riesgo.

Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación.


Figura (20)

**Transparencia**

Las ceremonias son el mecanismo fundamental para una clara visibilidad sobre el trabajo en equipo ya que permiten que los aspectos importantes del proceso sean visibles y comprensibles; mediante un mensaje común para todos los que participan de ellas. ES así que, los observadores y responsables del resultado compartan la misma idea.

Por ejemplo, al utilizar un taskboard con las tareas pendientes, en proceso y terminadas estamos mostrando a todos los miembros del equipo lo que está sucediendo día a día. Otro ejemplo es la definición de “Done”; tal que aquellos que desempeñan el trabajo y quienes inspeccionan el incremento resultante deben compartir una definición común del estado de terminado o Done.

**Inspección**

Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su

inspección no debe ser tan frecuente como para que pueda interferir en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo.

**Adaptación**

Sucede cuando el Equipo Principal de Scrum aprende a través de la transparencia y la inspección y luego se adaptan al hacer mejoras en el trabajo ya en progreso, por ejemplo, la implementación de buenas prácticas. El último día de la iteración se realiza una reunión de revisión en la que el equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado, con el mínimo esfuerzo. En esa misma reunión, el equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad.

Scrum propone cuatro eventos formales, contenidos dentro del Sprint, para la inspección y adaptación: Planificación del Sprint (Sprint Planning), Scrum Diario (Daily Scrum), Revisión del Sprint (Sprint Review) y Retrospectiva del Sprint (Sprint Retrospective).

## Valores de Scrum


Figura (21)

1. ***Foco***: Nos centramos en trabajar en el mínimo número de cosas posible y evitamos distracciones de manera que consigamos entregar antes, trabajar mejor juntos y obtener resultados de mayor calidad.
2. ***Coraje***: Dado que el equipo es uno y por tanto no afrontamos los retos y objetivos solos, podemos aceptar retos mayores confiando en poder alcanzarlo todos juntos como un equipo.
3. ***Franqueza***: Contamos de manera clara cómo avanzamos, los problemas que tenemos y manifestamos nuestras preocupaciones.
4. ***Compromiso***: Nosotros somos los que controlamos la cantidad de trabajo que hacemos y cómo lo hacemos, por tanto, nos comprometemos con el resultado esperado.
5. ***Respeto***: Al trabajar juntos, compartiendo éxitos y fracasos, nos iremos conociendo y respetando cada uno como es, asumiendo que todos aportamos lo mejor de nosotros mismos en cada momento.

A continuación, presentamos algunas situaciones donde estos valores nos pueden guiar:

| Situación | Comportamiento siguiendo valores | Anti-patrón | Valores involucrados |
|---|---|---|---|
| Otros equipos nos piden ayuda frecuentemente | Indicar dónde pueden encontrar la solución para que puedan ser autónomos en el futuro asistiéndoles sólo en momentos de especial dificultad.<br>Si fuera demasiado recurrente facilitarles el contacto de algún arquitecto del área de la compañía y educadamente indicar que necesitas focalizarte en el trabajo de tu equipo. | Ir cada vez a ayudarles dejando nuestro trabajo. | Foco y compromiso |
| El Propietario del Producto quiere introducir nuevas funcionalidades al sprint | Si está dentro del objetivo y él ve que es más relevante que lo que estamos haciendo aceptarlo indicando los riesgos posibles y si fuera necesario avisar que algo comprometido puede no acabarse.<br>Si fuera recurrente sacar el problema en la retrospectiva. | Aceptar siempre lo que el Propietario del Producto nos agregue. | Compromiso, coraje y franqueza |

| | | | |
|---|---|---|---|
| El Propietario del Producto pide una funcionalidad que no parece posible realizar | Se acepta el reto solicitando una ventana de tiempo para poder experimentar antes de decir que no se puede hacer. | Indicar que no es posible hacer lo que pide. | Coraje y compromiso |
| Un miembro del equipo parece distraído | Le preguntas si le pasa algo a lo que te da las gracias por preocuparte pero que es algo personal.<br>Le dejas su espacio, hoy lo necesita. | Le recriminas que qué le pasa hoy y que así no va a salir el trabajo. | Respeto |

Tabla (2)

El uso exitoso de Scrum depende que las personas lleguen a desarrollar unas habilidades extraordinarias en alcanzar las metas del Equipo Scrum (Scrum Team). Los miembros del Equipo Scrum (Scrum Team). tienen coraje para hacer bien las cosas y para trabajar en los problemas difíciles. Todos se enfocan en el trabajo del Sprint y en las metas del Equipo Scrum (Scrum Team).

## Sus principios

Los principios de Scrum son las pautas básicas para aplicar el marco de Scrum y obligatoriamente deben usarse en todos los proyectos de Scrum. Los seis principios de Scrum son:

1. ***Control del proceso empírico***: se trata de la filosofía central de las ideas de Scrum a través de la transparencia, inspección y adaptación.
2. ***Auto-organización***: este principio se centra en la productividad del equipo que entrega un valor significativamente mayor cuando está auto-organizado.
3. ***Colaboración***: se centra en el trabajo colaborativo mediante la idea de crear conciencia de entrega de valor compartido como mayor valor.
4. ***Priorización basado en valor***: este principio pone de relieve el enfoque de Scrum para ofrecer el máximo valor de negocio, desde el principio del proyecto hasta su conclusión.
5. ***Tiempo asignado***: en Scrum se utiliza el concepto de time box para guiar y organizar todas las ceremonias.
6. ***Desarrollo Iterativo***: Este principio define el desarrollo iterativo y enfatiza cómo adaptarse a los cambios y crear productos que mantengan el valor para el cliente.

## Roles

El Equipo Scrum (Scrum Team) consiste en un Propietario del Producto (Product Owner), el Equipo de Desarrollo (Development Team) y un Scrum Master.

**Development Team** (El equipo de desarrollo)

El Equipo de Desarrollo (Development Team) se compone de profesionales que realizan el trabajo de entregar un Incremento de producto "Terminado" ("Done") que potencialmente se pueda poner en producción al final de cada Sprint. Solo los miembros del Equipo de Desarrollo (Development Team) participan en la creación del Incremento.

La organización es la encargada de estructurar y empoderar a los Equipos de Desarrollo para que estos organicen y gestionen su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo (Development Team).

Los Equipos de Desarrollo (Development Teams) tienen las siguientes características:

- Son auto-organizados. Nadie (ni siquiera el Scrum Master) indica al Equipo de Desarrollo (Development Team) cómo convertir elementos de la Pila del Producto (Product Backlog) en Incrementos de funcionalidad potencialmente desplegables;
- Los Equipos de Desarrollo (Development Teams) son multifuncionales, con todas las habilidades necesarias para crear un Incremento de producto;
- Scrum no reconoce títulos para los miembros de un Equipo de Desarrollo (Development Team), independientemente del trabajo que realice cada persona;
- Scrum no reconoce sub-equipos en los Equipos de Desarrollo, no importan los dominios particulares que requieran tenerse en cuenta, como pruebas, arquitectura, operaciones, o análisis de negocio;
- Los miembros individuales del Equipo de Desarrollo (Development Team) pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo (Development Team) como un todo.

## Tamaño del Equipo de Desarrollo (Development Team)

El tamaño óptimo del Equipo de Desarrollo (Development Team) es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para poder completar una cantidad de trabajo significativa. Tener menos de

tres miembros en el Equipo de Desarrollo (Development Team) reduce la interacción y resulta en ganancias de productividad más pequeñas. Los Equipos de Desarrollo más pequeños podrían encontrar limitaciones en cuanto a las habilidades necesarias durante un Sprint, haciendo que el Equipo de Desarrollo (Development Team) no pudiese entregar un Incremento que potencialmente se pueda poner en producción. Tener más de nueve miembros en el equipo requiere demasiada coordinación.

Los Equipos de Desarrollo grandes generan demasiada complejidad como para que un proceso empírico pueda ser de utilidad. Los roles de Propietario del Producto (Product Owner) y Scrum Master no se contabilizan en el cálculo del tamaño del equipo a menos que también estén contribuyendo a trabajar en la Pila del Sprint (Sprint Backlog).

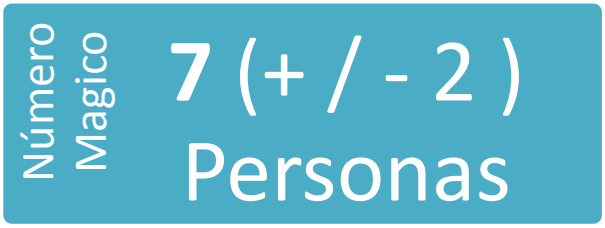



Figura (22)

## ¿Qué significa ser un equipo auto-organizado?

Los principales objetivos de los equipos auto-organizados son los siguientes:

- Entender la visión del proyecto y el valor para la organización.
- Estimar y asignar historias de usuarios y asignar tareas a sí mismos durante el proceso de Elaboración de la lista de pendientes del Sprint.
- Crear tareas de forma independiente durante el proceso de Elaboración de tareas.
- Aplicar y aprovechar la experiencia de ser un equipo multi-funcional al trabajar en las tareas durante el proceso de Crear entregables.
- Lograr resultados tangibles, que son aceptados por el cliente durante el proceso de Demostración y validación del Sprint.
- Resolver problemas individuales al discutirlos durante las reuniones diarias.
- Aclarar cualquier discrepancia o duda y estar abierto a aprender cosas nuevas.
- Actualizar los conocimientos y habilidades de manera continua a través de interacciones regulares dentro del equipo.
- Mantener la estabilidad de los miembros del equipo durante toda la duración del proyecto al no cambiar los miembros, a menos que sea inevitable.

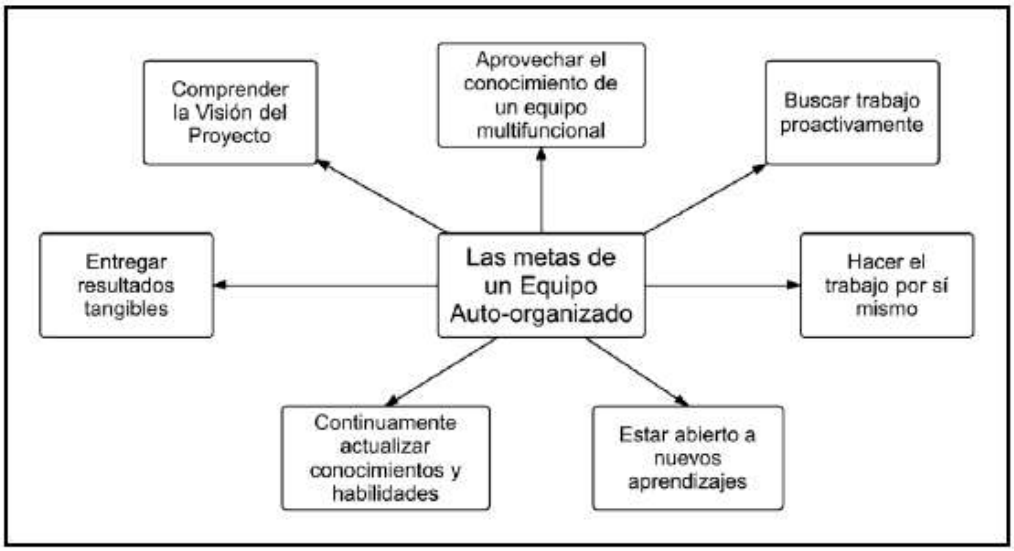



Figura (23)

**Product Owner** (El Propietario del producto)

Es la persona responsable es el responsable de maximizar el valor del producto del trabajo del Equipo de Desarrollo (Development Team) y es responsable de expresar y ordenar claramente los elementos de la Pila del Producto (Product Backlog).

El P.O tiene la responsabilidad de:

- Ordenar los elementos en la Pila del Producto (Product Backlog) para alcanzar los objetivos y las misiones de la mejor manera posible;
- Optimizar el valor del trabajo que realiza el Equipo de Desarrollo (Development Team); que la Pila del Producto (Product Backlog) sea visible, transparente y clara para todos y que muestre, lo que el equipo trabajará a continuación; y,
- Asegurar que el Equipo de Desarrollo (Development Team) entiende los elementos de la Pila del Producto (Product Backlog) a nivel necesario.
- El Propietario del Producto (Product Owner) es una única persona. Para que el Propietario del Producto (Product Owner) pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Propietario del Producto (Product Owner) se reflejan en el contenido y en la priorización de la Pila del Producto (Product Backlog).

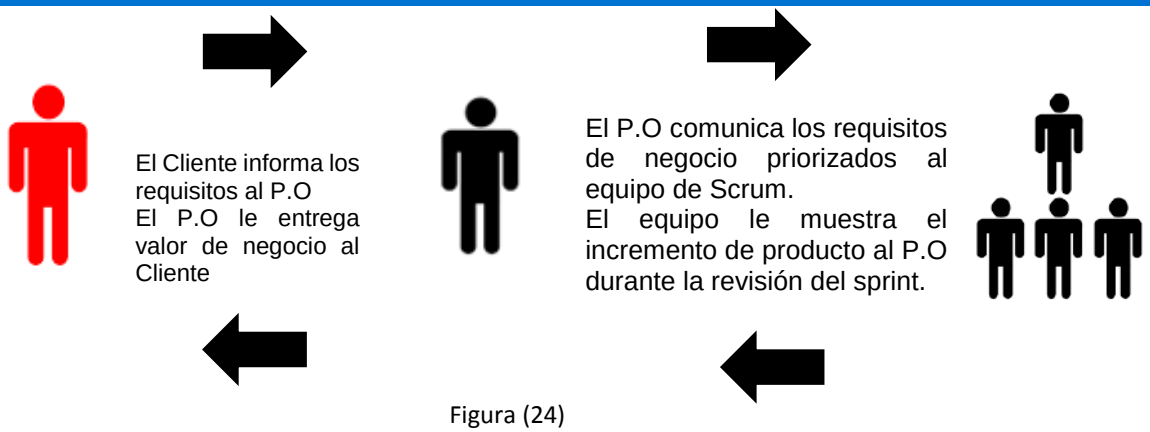



Figura (24)

**El Scrum Master:**

El Scrum Master es un facilitador que asegura que el Equipo Scrum se encuentre en un ambiente propicio para completar el proyecto con éxito. El Scrum Master guía, facilita y les enseña las prácticas de Scrum a todos los involucrados en el proyecto y asegura que se estén siguiendo los procesos de Scrum.

El Scrum Master es un colaborador líder que está al servicio del Equipo Scrum (Scrum Team), al Product Owner y la organización.

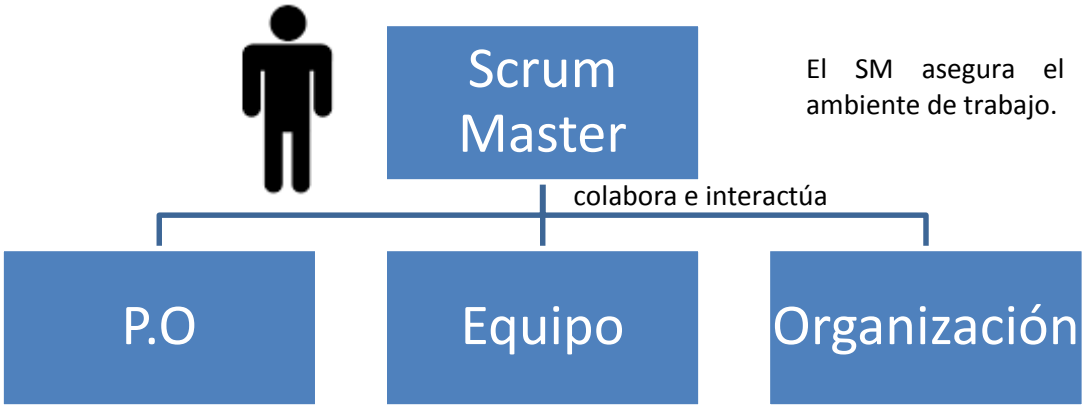



Figura (25)

**SM colabora con el Product Owner tal que:**

El SM asegura que los objetivos, el alcance y el dominio del producto sean entendidos por todos en el equipo Scrum (Scrum Team) y ayuda al Equipo Scrum a entender la necesidad de contar con elementos de Pila de Producto claros y concisos.

También asegurar que el Propietario del Producto (Product Owner) conozca cómo ordenar la Pila del Producto (Product Backlog).

**El Scrum Master sirve al Equipo de Desarrollo (Development Team) de varias formas, incluyendo:**

- Guiar al Equipo de Desarrollo (Development Team) en ser auto-organizado y multifuncional;
- Ayudar al Equipo de Desarrollo (Development Team) a crear productos de alto valor;
- Eliminar impedimentos para el progreso del Equipo de Desarrollo (Development Team);
- Facilitar los eventos de Scrum según se requiera o necesite;
- Guiar al Equipo de Desarrollo (Development Team) en entornos organizacionales en los que Scrum aún no haya sido adoptado y entendido por completo.

**El Scrum Master sirve a la organización de varias formas, incluyendo:**

- Liderar y guiar a la organización en la adopción de Scrum;
- Planificar las implementaciones de Scrum en la organización;
- Ayudar a los empleados e interesados a entender y llevar a cabo Scrum y el desarrollo empírico de producto;
- Motivar cambios que incrementen la productividad del Equipo Scrum; y, con otros Scrum Masters para incrementar la efectividad de la aplicación de Scrum en la organización.

## Eventos de Scrum y time-boxes

En Scrum existen diferentes eventos predefinidos con el fin de crear un ritmo de trabajo regular y minimizar la necesidad de reuniones que no sean del marco teórico que propone Scrum.

Todos los eventos son compartimientos o periodos de tiempo limitado (time-boxes), de tal modo que todos tienen una duración máxima.

Además del propio Sprint, que es un contenedor del resto de eventos, cada uno de los eventos de Scrum constituye una oportunidad formal para la inspección y adaptación en algún aspecto. Algo muy importante es que una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse. Los otros eventos pueden terminar siempre que se alcance el objetivo del evento.

Estos eventos se diseñaron específicamente para habilitar los pilares vitales de transparencia e inspección. La falta de alguno de estos eventos da como resultado una reducción de la transparencia y constituye una oportunidad perdida de inspección y adaptación.

## Sprint

El aspecto clave de Scrum es el Sprint***, es un período de tiempo (time-box)*** de un mes o menos durante el cual se crea un incremento de producto "Terminado" utilizable y potencialmente desplegable y será consistente a lo largo de todo el esfuerzo de desarrollo.

Cada ***sprint tiene un objetivo*** (Sprint Goal) a ser alcanzado por el Scrum Team. Siempre cada nuevo Sprint comienza inmediatamente después de la finalización del Sprint anterior.

Los Sprints contienen y consisten en la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), el trabajo de desarrollo, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective).

Durante el Sprint no se realizan cambios que puedan afectar al objetivo del Sprint (Sprint Goal);

Los objetivos de calidad no disminuyen; y el alcance puede clarificarse y renegociarse entre el Product Owner (Propietario del Producto) y el Equipo de Desarrollo a medida que se va aprendiendo más.

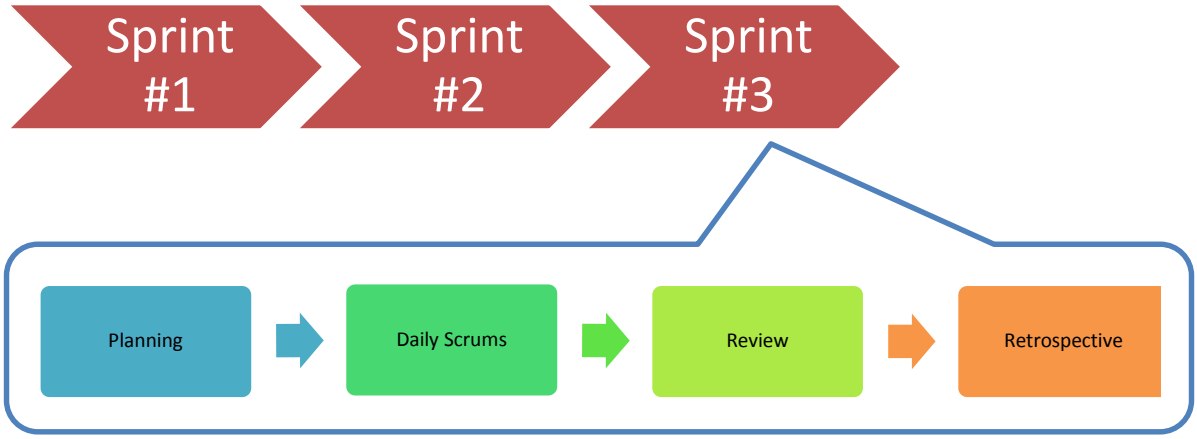



Figura (26)

**Sprint Planning Meeting**: La planificación de las tareas a realizar en la iteración se divide en dos partes:

- QUE: Primera parte de la reunión. Se realiza en un timebox de alrededor de 4 horas (si la iteración es de 2 semanas):
  - El PO presenta al equipo la lista de requisitos priorizada del producto o proyecto, pone nombre a la meta de la iteración (de manera que ayude a tomar decisiones durante su ejecución) y propone los requisitos más prioritarios a desarrollar en ella.

- El equipo examina la lista, pregunta al PO las dudas que le surgen, añade más condiciones de satisfacción y selecciona los objetivos/requisitos más prioritarios que prevé completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

- COMO: Segunda parte de la reunión. Se realiza en un timebox de alrededor de 4 horas (si la iteración es de 2 semanas). El equipo planifica la iteración, elabora la **táctica** que le permitirá conseguir el mejor resultado posible con el mínimo esfuerzo. Esta actividad la realiza el equipo dado que es el responsable de organizar su trabajo y es quien mejor conoce cómo realizarlo.
  - Define las tareas necesarias para poder completar cada objetivo/requisito, creando la lista de tareas de la iteración (Sprint Backlog) basándose en la definición de hecho (DoD).
  - Realiza una estimación conjunta del esfuerzo necesario para realizar cada tarea.
  - Los miembros del equipo se autoasignan las tareas que pueden realizar, se autoorganizan para trabajar incluso en parejas (o grupos mayores) con el fin de compartir conocimiento (creando un equipo más resiliente) o para resolver juntos objetivos especialmente complejos.

**Sprint Daily Meeting**: Es la ***reunión diaria de sincronización del equipo***, además es una reunión corta con una duración de 15 minutos (dependiendo del equipo se calcula por miembro 3 minutos) donde los miembros del equipo se reúnen para informar sobre cómo marcha el proyecto, respondiendo a las siguientes tres preguntas:

1. *¿Qué terminé desde ayer?*
2. *¿Qué voy a terminar hoy?*
3. *¿Qué obstáculos existen en el proyecto?*

Esta reunión se lleva a cabo por el equipo como parte del proceso de llevar a cabo el Standup diario. Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para al finalizar la reunión poder hacer las adaptaciones necesarias que permitan cumplir con la previsión que hizo el equipo de objetivos a entregar al final de la iteración (en la reunión de planning).

**Sprint Review Meeting**: Durante el Sprint Review Meeting que se lleva a cabo en el proceso de Demostración y validación del Sprint, el Equipo Scrum le presenta los entregables del Sprint actual al Propietario del producto. El Propietario del producto revisa el product para compararlos con los Acceptance Criteria acordados. Finalmente, el PO acepta o rechaza los User Stories concluidos.

Se puede decir que es ***un incremento del producto y la adaptación de la Pila del Producto (Product Backlog)***. Esta es una reunión de cuatro horas como máximo para Sprints de un mes.

***Retrospective Sprint Meeting***: Es una reunión que ocurre después el a revisión review. Es una oportunidad para que el Equipo Scrum se inspeccione a sí mismo y cree un plan para implementar mejoras durante el próximo sprint. por lo que brinda una oportunidad formal para enfocarse en la inspección y adaptación (realiza un crecimiento en el equipo). El equipo dialoga sobre lo que salió bien durante el Sprint anterior y lo que no salió bien, con el objetivo de aprender y mejorar los Sprints futuros a partir de buenas prácticas y cursos de acción que se derivan de las retrospectivas. Para un Sprint de un mes, dura 4 horas.

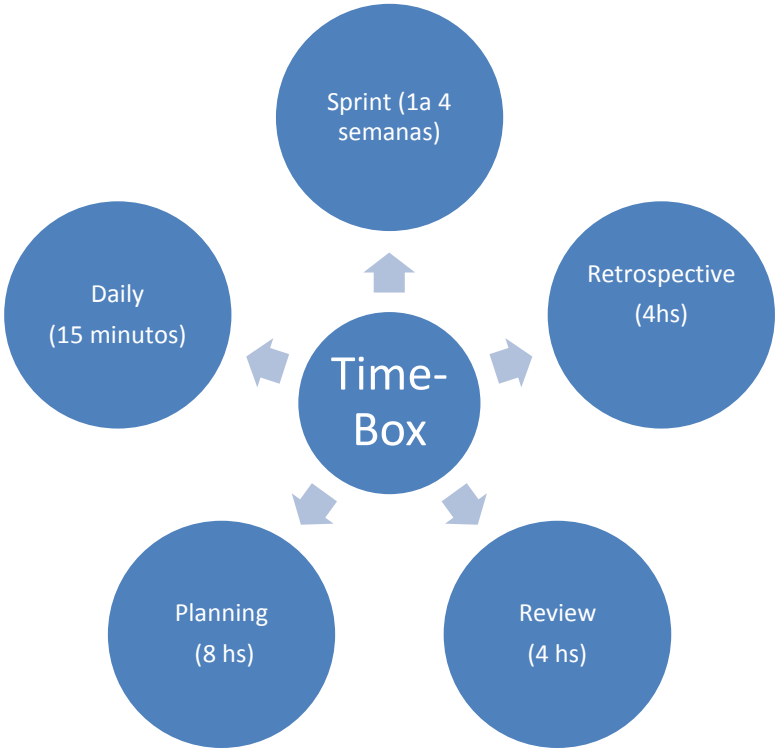



Figura (27)

## Artefactos de Scrum

Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto.

**Pila del Producto (Product Backlog)**

Es una lista ordenada de todo lo conocido que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Propietario del Producto (Product Owner) es el responsable de la Pila del Producto (Product Backlog), incluyendo su contenido, disponibilidad y ordenación.

Una Pila del Producto (Product Backlog) nunca está completa y es un artefacto dinámico ya que los requisitos nunca dejan de cambiar así que la Pila del Producto (Product Backlog) también cambia. (El desarrollo más temprano de la misma solo refleja los requisitos conocidos y mejor entendidos al principio.)

La Pila del Producto (Product Backlog) enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras.

**¿Qué elementos tiene la Pila de producto?**

Los elementos de la Pila del Producto (Product Backlog) tienen como atributos la descripción, el orden, la estimación y el valor. Ejemplo:

| ID | NOMBRE | IMPORTANCIA (0-10) | ESTIMACIÓN (Dias) | CÓMO PROBARLO |
|---|---|---|---|---|
| 1 | Spring Rest Web Service | 10 | 30 | Resultado final de la aplicación. |
| 2 | Análisis de la aplicación | 10 | 4 | Resultado final de la aplicación. |
| 3 | Plantilla del proyecto: Arquitectura básica incluyendo Acceso a BD MySql y patrón Web Service Implementado mediante Spring. | 9 | 3 | Funcionalidad simple que devuelve un resultado del servidor. |
| 4 | Adaptación de las funcionalidades de parte del servidor a la plantilla del proyecto. | 8 | 20 | Lógica de la aplicación |
| 5 | Adaptación de las funcionalidades de parte del cliente a la plantilla del proyecto. | 8 | 20 | Lógica de la aplicación |
| 6 | Pruebas funcionales | 9 | 5 | Casos de prueba |
| 7 | Pruebas de integración | 9 | 5 | Navegador Web |
| 8 | Gestión de documentación | 10 | 30 | |

Figura (28)

A medida que un producto es utilizado y se incrementa su valor y el mercado proporciona retroalimentación, la Pila del Producto (Product Backlog) se convierte en una lista más larga y exhaustiva.

Pila del sprint (Sprint Backlog)

Pila del Sprint (Sprint Backlog) La Pila del Sprint (Sprint Backlog) es el conjunto de los elementos de la Pila del Producto (Product Backlog) seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el objetivo del Sprint.

La Pila del Sprint (Sprint Backlog) es una predicción hecha por el Equipo de Desarrollo (Development Team) acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”.

La Pila del Sprint (Sprint Backlog) es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Scrum Diario (Daily Scrum).

El Equipo de Desarrollo (Development Team) modifica la Pila del Sprint (Sprint Backlog) durante el Sprint y esta Pila del Sprint (Sprint Backlog) emerge a lo largo del Sprint. Esto ocurre a medida que el Equipo de Desarrollo (Development Team) trabaja en lo planeado y aprende más acerca del trabajo necesario para conseguir el objetivo del Sprint.

Cuando se requiere nuevo trabajo, el Equipo de Desarrollo (Development Team) lo adiciona a la Pila del Sprint (Sprint Backlog). A medida que el trabajo se ejecuta o se completa se va actualizando la estimación de trabajo restante. Cuando algún elemento del plan se considera innecesario, es eliminado. Solo el Equipo de Desarrollo (Development Team) puede cambiar su Pila del Sprint (Sprint Backlog) durante un Sprint. La Pila del Sprint (Sprint Backlog) es una imagen visible en tiempo real del trabajo que el Equipo de Desarrollo (Development Team) planea llevar a cabo durante el Sprint y pertenece únicamente al Equipo de Desarrollo.

Ejemplo:

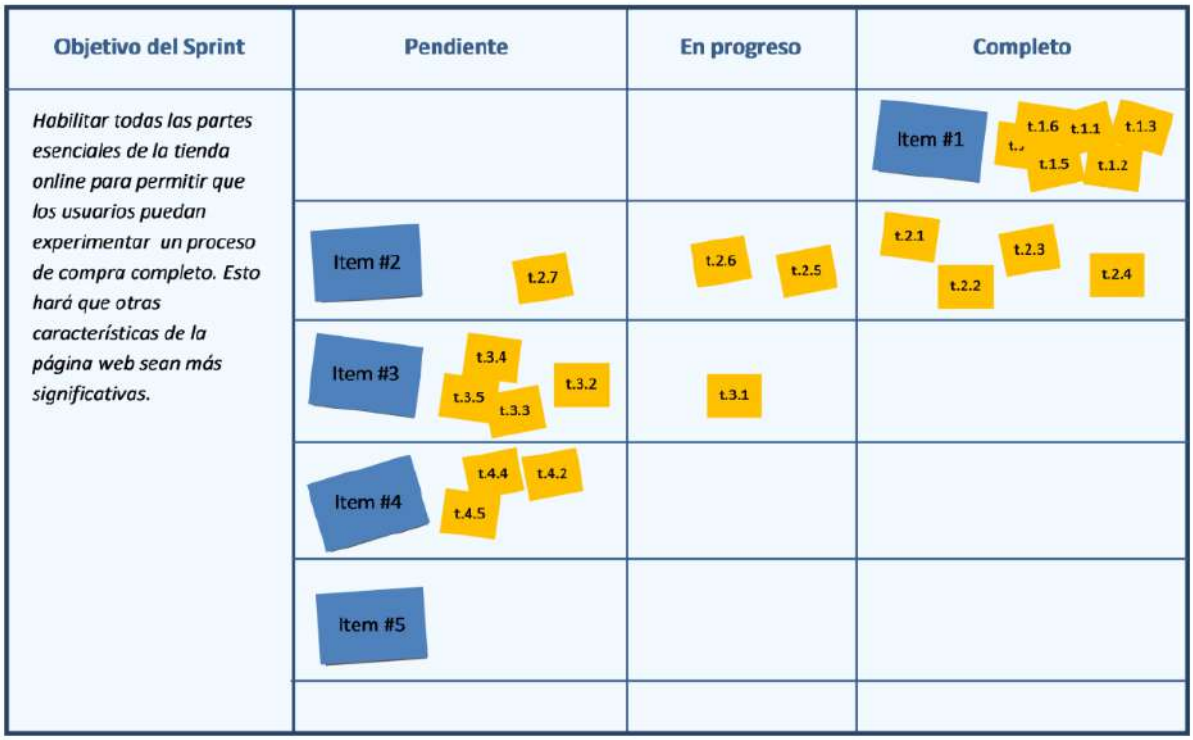



Figura (29)

**Incremento (Increment)**

El Incremento es la suma de todos los elementos de la Pila del Producto (Product Backlog) completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint el nuevo Incremento debe estar "Terminado", lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de "Terminado" del Equipo Scrum. Un incremento es un cuerpo de trabajo inspeccionable y terminado que respalda el empirismo al final del Sprint. El incremento es un paso hacia una visión o meta. El incremento debe estar en condiciones de utilizarse sin importar si el Propietario del Producto (Product Owner), decide liberarlo o no.

### Definición de terminado

Cuando un elemento de la Pila del Producto (Product Backlog) o un Incremento se describe como "Terminado", todo el mundo debe entender lo que significa "Terminado" ("Done").

Esta misma definición guía al Equipo de Desarrollo (Development Team) en saber cuántos elementos de la Pila del Producto (Product Backlog) puede seleccionar durante la Planificación del Sprint (Sprint Planning). El propósito de cada Sprint es entregar Incrementos de funcionalidad que potencialmente se puedan poner en producción y que se ajustan a la Definición de "Terminado" ("Done") actual del Equipo Scrum (Scrum Team).

Cada Incremento se integra con todos los Incrementos anteriores y es probado de manera exhaustiva, asegurando que todos los Incrementos funcionan en conjunto. A medida que los Equipos Scrum maduran, se espera que su definición de "Terminado" ("Done") amplíe para incluir criterios más rigurosos para una mayor calidad. El uso de las nuevas definiciones puede descubrir trabajo por hacer en los incrementos previamente "Terminados" ("Done"). Cualquier producto o sistema debería tener una definición de "Terminado" ("Done") que es un estándar para cualquier trabajo realizado sobre él.

### El proceso en Scrum

El marco de Scrum es impulsado por el objetivo de ofrecer el máximo valor empresarial en un período de tiempo mínimo. Para lograr esto de forma práctica, Scrum cree en Desarrollo iterativo of Deliverables (entregas de desarrollo iterativas).

En la mayoría de los proyectos complejos, el Customer puede que no sea capaz de definir unos requisitos muy concretos o puede no estar seguro de cómo debería de ser el product final. El modelo iterativo es más flexible para asegurar que cualquier cambio solicitado por el customer pueda ser incluido como parte del

proyecto. Es posible que los User Stories tengan que ser escritos constantemente a lo largo de la duración del proyecto. En las etapas iniciales de la escritura, la mayoría de los User Stories son las funcionalidades de alto nivel. Estos User Stories son conocidos como Epic(s). Epics, por lo general son muy grandes para que los equipos los completen en un sólo Sprint y por lo tanto se dividen en pequeños User Stories.

Cada aspecto complejo del proyecto se divide mediante la elaboración progresiva durante el proceso de Mantenimiento de la lista priorizada de pendientes del producto. Los procesos de Elaborar historias de usuario y Estimate, Approve, and Commit User Stories se utilizan para agregar nuevos requisitos para el Prioritized Product Backlog. La tarea del Propietario del producto es asegurar un mayor retorno de la inversión (ROI), centrándose en el valor y la entrega continua con cada Sprint. El Propietario del producto debería tener una buena comprensión del Justificación del negocio y el valor que el proyecto se supone debe entregar cuando redacta el Prioritized Product Backlog, y por lo tanto decidir qué entregables contractuales y valores se han de entregar en cada Sprint. Luego, los procesos de Elaboración de tareas, Estimar tareas, y Elaboración de la lista de pendientes del Sprint producen el Sprint Backlog lo cual el equipo utiliza para crear los entregables.

En cada Sprint, el proceso de Crear entregables se utiliza para desarrollar las salidas del Sprint. El Scrum Master tiene que garantizar que se sigan los procesos de Scrum y facilitar al equipo el trabajo de la manera más productiva. El Equipo Scrum se auto-organiza y tiene como objetivo crear Sprint Deliverables de los User Stories en el Sprint Backlog. En grandes projects, varios equipos multifuncionales funcionan en paralelo a través de Sprints, entregando soluciones potencialmente entregables al final de cada Sprint.

Después de que cada Sprint se ha completado, el Propietario del producto acepta o rechaza los entregables basados en los Acceptance Criteria del proceso Demostración y validación del Sprint. La gente obtiene comentarios y observaciones que se puede incorporar al siguiente Sprint.

Scrum pone el énfasis en productos que funcionen correctamente al final del Sprint y potencialmente para entregar. (Figura 30)

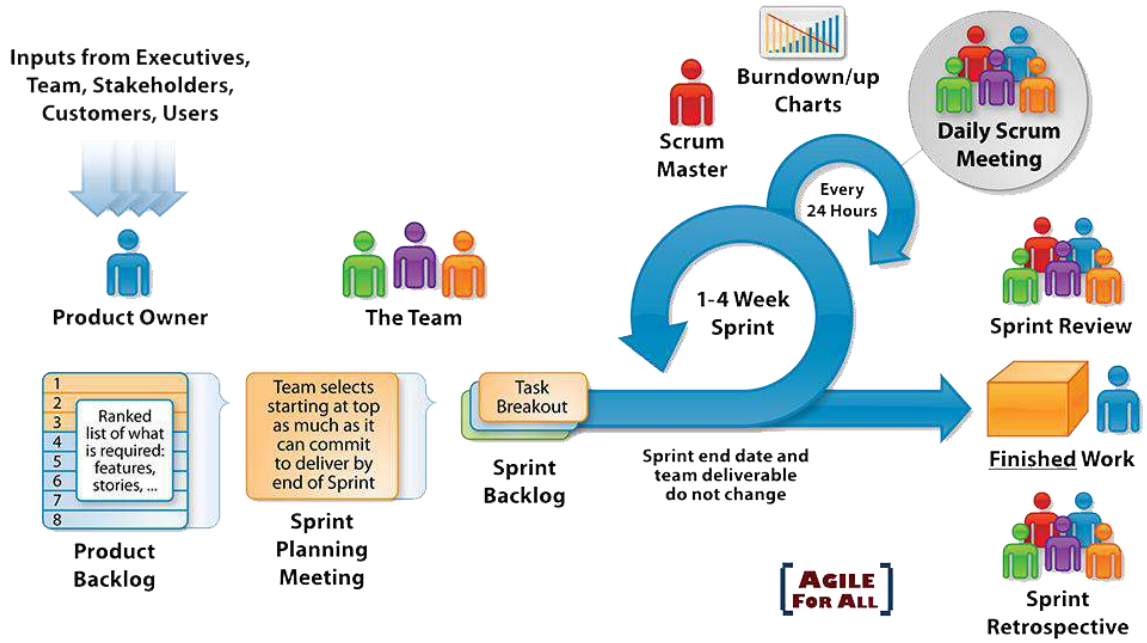



Figura (30)

## Conclusiones

El énfasis en la gestión de proyectos tradicional es llevar a cabo la planificación detallada por adelantada del proyecto con énfasis en gestionar y solucionar el alcance, costo, horarios y gestión de esos parámetros.

El Marco de Scrum se basa en la creencia de que los equipos pueden ofrecer mucho más que sus conocimientos técnicos, y que tratar de asignar y planear en un entorno de constante cambio no es eficiente. Por lo tanto, Scrum anima la toma de decisiones iterativa basada en datos. En Scrum, el enfoque principal es la entrega de productos que satisfagan los requisitos del customer en pequeños incrementos iterativos que sean funcionales y aporten valor al negocio.

Para entregar la mayor cantidad de valor en el menor tiempo posible, Scrum promueve priorizar y trabajar con marcos de tiempo corto sobre la fijación del alcance, costo y cronograma de un proyecto. Una característica importante de Scrum es Auto-organización.

Lectura recomendada para ampliar conceptos: Ken Schwaber y Jeff Sutherland (2017) “La Guía de Scrum”.Ed.Scrum.org

Lectura recomendada para ampliar conceptos: SCRUMStudy (2016) “A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™GUIDE) 2016 Edition”. Estados Unidos: SCRUMstudy™

## Principales diferencias entre la metodología tradicional y ágil

Las metodologías de desarrollo tradicionales imponen una disciplina de trabajo fundamentada en la documentación sobre el proceso de desarrollo de software, se hace hincapié en la planificación global y total de todo el trabajo a realizar, y una vez que esté detallado, comienza el ciclo de desarrollo de software; caso contrario a lo que respecta a las metodologías de desarrollo ágiles que muchas veces obvia la documentación y se centra en el trabajo buscando el equilibrio entre proceso y esfuerzo.

| METODOLOGÍA ÁGILES | METODOLOGÍA TRADICIONALES |
|---|---|
| Están preparadas para el cambio durante el proyecto | Son poco flexibles a los cambios |
| Proceso menos controlado, con pocos principios | Proceso muchos más controlados, con numerosas normas |
| No existe tradicional o al menos es bastante flexible | Existe un contrato prefijado |
| El cliente es parte activa en el proceso de desarrollo | El cliente interactúa con el equipo solo mediante reuniones de entregas |
| Grupos pequeños, 9 integrantes o menos y trabajando en el mismo sitio en el cual todos tiene conocimiento sobre todo el proceso de desarrollo | Grupos grandes y posiblemente distribuidos donde a cada integrante se le asigna tareas especificas |
| Menos énfasis en la arquitectura del software | La arquitectura del software es esencial y se expresa mediante modelos |

Tabla (3)

En resumen, la figura 31, muestra y compara las etapas principales de ambas metodologías.


Figura (31)

## ¿Qué es un requerimiento?

Un requerimiento de software puede ser definido como una capacidad del SW necesaria por el usuario para poder resolver un problema o alcanzar un objetivo.

Una capacidad del SW debe ser reunida o poseída por un sistema o componente del sistema para satisfacer un contrato, especificación u otra documentación formal del proyecto.

*"Si una compañía desea otorgar un contrato para un gran proyecto de desarrollo de software, tiene que definir sus necesidades de una forma suficientemente abstracta para que una solución no esté predefinida. Los requerimientos deben redactarse de tal forma que muchos proveedores liciten en pos del contrato, ofreciendo, tal vez, diferentes maneras de cubrir las necesidades de organización del cliente. Una vez otorgado el contrato, el proveedor tiene que escribir con más detalle una definición del sistema para el cliente, de modo que éste comprenda y valide lo que hará el software. Estos documentos suelen nombrarse documentos de requerimientos para el sistema". Somerville (2011)" Ingeniería de software Novena edición (pp 82-118)" México: Pearson*

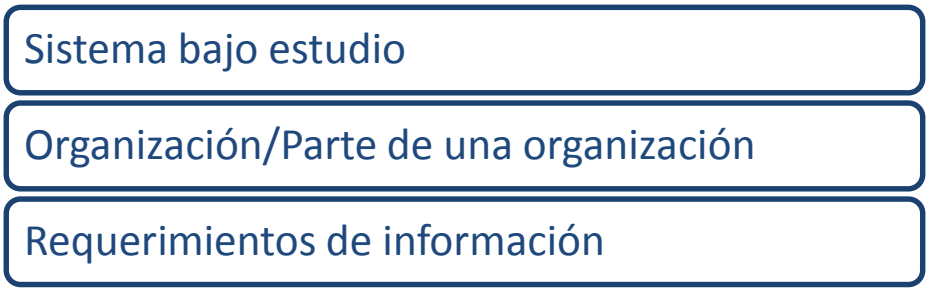



Figura (32)

Para poder determinar a qué actividades de la organización realizará apoyo el sistema de información y cuál será la información

de salida que debería producir, es necesario realizar un estudio de los requerimientos de información que se presentan en cada organización en particular.

## Análisis de los requerimientos

La determinación de los requerimientos sugiere estudiar el sistema actual con la finalidad de entender cómo trabaja y dónde se debe mejorar.

Un requerimiento es una característica que debe incluirse en un nuevo sistema y puede consistir en una forma de captar o procesar datos, producir información, controlar una actividad o dar apoyo a la gerencia.

La determinación de requerimientos significa estudiar el sistema existente y recopilar los datos en relación con éste para encontrar cuáles son las necesidades que se presentan. Para poder identificar requerimientos debemos adquirir conocimiento de la organización y captar todos aquellos problemas causados por errores producidos en la información, como así también la falta de información que lleva a las personas que trabajan en las mismas a realizar sus actividades con cierto grado de incertidumbre.

Por ejemplo, algunos requerimientos que pueden presentarse al entrevistarnos con personas que trabajan en una organización y donde es necesario realizar un proyecto de sistemas:

- Necesidad de contar con información actualizada de ventas.
- Informes estadísticos de producción.
- Consultas sobre las cuentas corrientes y saldos de los clientes.
- Disminuir errores en los cálculos de las operaciones.
- Obtener rápidamente información sobre precios de los proveedores.

## Importancia de los requerimientos

El nivel de importancia de los requerimientos ha dado lugar a una especialización de la ingeniería.

La ingeniería de requerimientos es una etapa particularmente crítica del proceso de software, ya que los errores en esta etapa conducen de manera inevitable a problemas posteriores tanto en el diseño como en la implementación del sistema.

El proceso de ingeniería de requerimientos se enfoca en evitar el fracaso de los proyectos de desarrollo de software y producir un documento que describa las necesidades del cliente y futuros usuario del nuevo sistema que brindará apoyo al sistema bajo estudio.

Dentro del proceso de ingeniería de requerimientos, existen 3 pasos fundamentales: Obtener, especificar y validar.


Figura (33)

1. ***Obtención y análisis de requerimientos***: Éste es el proceso de obtener o descubrir los requerimientos del sistema mediante un estudio inicial o relevamiento, discusiones con los usuarios potenciales, etcétera. Esto puede incluir el desarrollo de uno o más modelos de sistemas y prototipos, lo que ayuda a entender el sistema que se va a especificar, por ejemplo: el diagrama de casos de uso provisto por el lenguaje unificado de modelado(UML).
2. ***Especificación de requerimientos***: Consiste en la actividad de transcribir la información recopilada durante la actividad de análisis, en un documento que define un conjunto de requerimientos. Los requerimientos, siempre, fueron especificados en el lenguaje natural del cliente, utilizando enunciados y frases convencionales propias del lenguaje natural y de la visión de cada persona, trayendo algunos problemas:

En primer lugar, todos los participantes que los utilizan deben interpretar su significado de la misma manera. Si el cliente piensa sobre un objeto o característica de una manera y los desarrolladores lo hacen de otra, los requerimientos se vuelven confusos, pero es improbable que clientes y desarrolladores tengan la misma comprensión a cerca de todas las palabras utilizadas por ello en muchos casos, se utilizan normas y estándares para redactar los requerimientos de modo tal que se puedan cumplir con estas características. Por ejemplo, utilizar una lista de verbos específicos para acciones del sistema. Ejemplo: Registrar, consultar, modificar, Generar, actualizar, etc. Es así que la especificación de requerimientos se vuelve un documento útil que define en forma completa precisa y verificable los requisitos, el diseño, el comportamiento u otras características de un sistema o componente de un sistema.

Es importante mencionar que la especificación debe, como hemos dicho, abordar la descripción de lo que hay que desarrollar no el cómo, el cuándo, etc. Esto implica clasificar y describir correctamente todos los requisitos.

3. ***Validación de requerimientos:*** Esta actividad verifica que los requerimientos sean:
   - Verificable: Si un requerimiento no se puede comprobar, entonces ¿cómo se sabe si se cumplió con él o no?
   - Conciso: Deber ser fácil de leer y entender. Su redacción debe ser simple y clara para todos.
   - Completo: Es decir, que proporciona la información suficiente para su comprensión.
   - Consistente: Un requerimiento es consistente si no es contradictorio con otro requerimiento.
   - No ambiguo: Un requerimiento no es ambiguo cuando tiene una sola interpretación. El lenguaje usado en su definición, no debe causar confusiones al lector.

## Problemas asociados a los Requerimientos

A lo largo de la historia del desarrollo del software, sucedieron hechos que demostraron que las mayores causas de fracaso de los proyectos de desarrollo se debían a fallas en la definición de los requerimientos. Alcanzar un acuerdo respecto de los requerimientos para un producto de software es el mayor desafío al que se enfrenta el desarrollo de software y es un problema fuertemente vinculado con la comunicación y los vínculos sociales que se establecen entre todos los involucrados en el proyecto de desarrollo.

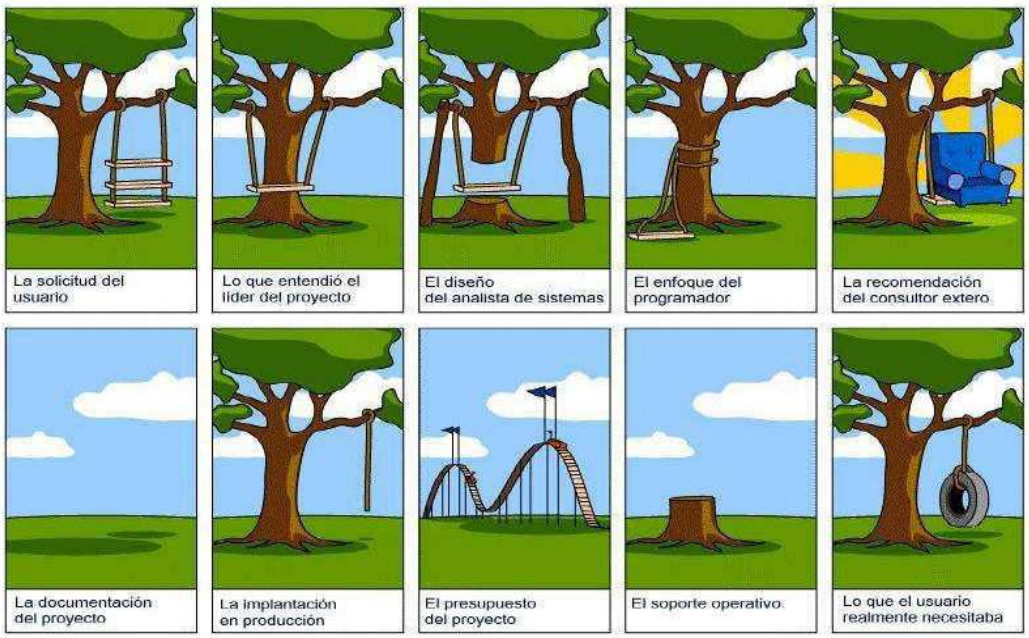



Figura (34)

Entre todos los problemas relacionados con los requerimientos, vamos a destacar los más comunes según nuestra experiencia:

- Los requerimientos no son siempre fáciles de expresar de manera clara mediante palabras.
- El número de requerimientos puede volverse inmanejable si no se controla.
- Los requerimientos cambian.

Con el uso de los frameworks para desarrollo de software se logra mejorar algunos de estos casos y disminuir el riesgo; no obstante, ninguna técnica o herramienta resulta perfecta y ha logrado evitar ambigüedades y malentendidos derivados de la interacción entre todas las personas que participan en el desarrollo de un producto de software.

En síntesis, la problemática de los requerimientos puede resumirse en:

- Dificultad de establecer un esquema de comprensión común entre el equipo de desarrollo y el grupo de clientes/usuarios.
- El cambio de los requerimientos como consecuencia de cambios en el dominio del problema.
- La imposibilidad de identificar la totalidad de los requerimientos de un producto al inicio del desarrollo.

De todo ello, podemos concluir y aconsejar que los requerimientos deben cumplir con dos características: ser objetivos y verificables. Tal que exista una interpretación única por parte de todos los involucrados (objetivo) y que exista un proceso finito en tiempo y conveniente en costo para determinar que el requerimiento se ha implementado adecuadamente y es verificable

Más adelante, retomaremos este tema luego de analizar las 2 metodologías para el análisis de sistemas: tradicionales y ágiles.

## Requerimientos según la metodología.

Hemos visto que hay dos paradigmas que abordan el uso de modelos encuadrados dentro de la metodología tradicional y otros modelos que se ubican dentro de las metodologías ágiles. En cualquier caso y dependiendo de las características del problema a resolver, el abordaje de los requerimientos es la fase más importante al principio del desarrollo de un SI. Por ello es necesario definirlos en forma completa precisa y verificable. De este modo, según se trate del entorno ágil vamos a referirnos a historias de usuario; mientras que para la metodología tradicional nos referimos como requerimientos de acuerdo a su clasificación y definición.

## Requerimientos en la metodología tradicional.

Es importante mencionar que la especificación de requerimientos describe lo que hay que desarrollar y no describe el cómo, el cuándo, etc. Esto implica clasificar y describir correctamente todos los requisitos.

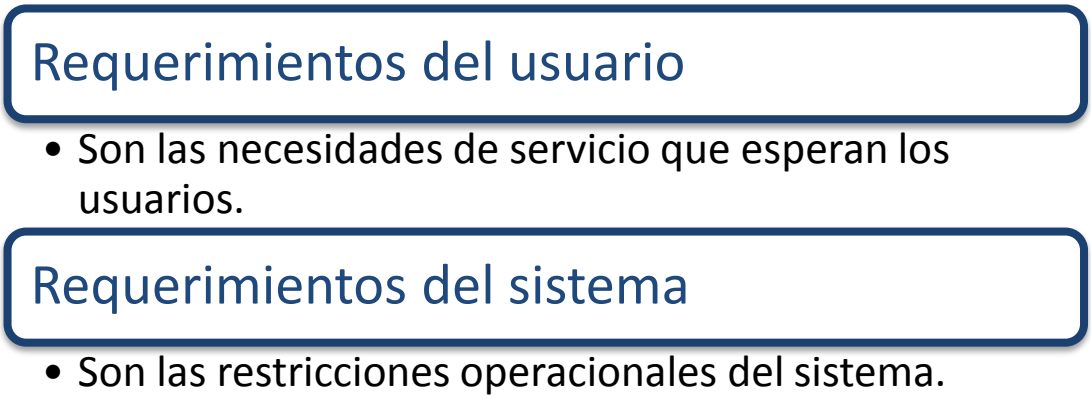



Figura (35)

*"Los requerimientos del usuario y los requerimientos del sistema se definen del siguiente modo:*

- *Los requerimientos del usuario son enunciados, en un lenguaje natural junto con diagramas, acerca de qué servicios esperan los usuarios del sistema, y de las restricciones con las cuales éste debe operar.*
- *Los requerimientos del sistema son descripciones más detalladas de las funciones, los servicios y las restricciones operacionales del sistema de software.*

*"(Somerville,2009)*

Se debe tener en cuenta que no siempre pueden especificarse detalles al inicio o que puede haber modificaciones entonces el requisito deber ser especificado de la forma más completa posible en la ERS de forma que sirvan de base para el diseño posterior. En caso de cambios debe hacerse de manera formal, para identificar, controlar, seguir e informar los mismos. Los cambios aprobados en los requisitos deben ser incluidos en la ERS. (Figura extraída de libro de Sommerville,2009)


Figura (36)

## Estructura de la ERS:

Se trata de un modelo propuesto por el estándar IEEE. No necesariamente se debe seguir este esquema o notación, pero cualquier ERS debe incluir toda esta información:

**1-INTRODUCCION**
1.1-Objetivo
1.2-Ambito
1.3-Definiciones, siglas y abreviaturas
1.5-Visión global
**2-DESCRIPCIÓN GENERAL**
2.1-Perspectiva de producto
2.2-Funciones del producto
2.3-Características del usuario
2.4-Limitaciones generales
2.5-Supuestos y dependencias
**3-REQUISITOS ESPECIFICOS**
3.1-Requisitos Funcionales
3.1.1-Requisito Funcional 1
3.1.1.1-Introducción
3.1.1.2-Entradas
3.1.1.3-Procesamiento
3.1.1.4-Salidas
Otros aspectos.
3.2- Requisitos de interfaz externa
3.2.1-Interfaces del usuario
3.2.2-Interfaces hardware
3.2.3-Interfaces software
3.2.4-Interfaces de comunicación
3.3-Requisitos de ejecución
3.4-Restricciones de diseño
3.4.1-Acatamiento de estándares
3.4.2-limitaciones hardware
.....
3.5-Atributos de calidad
3.5.1-Seguridad
3.5.2-Mantenimiento
.....
3.6-Otros requisitos
3.6.1-Base de datos
3.6.2-Operaciones
3.6.3-Adaptación de situación
.....
**Apéndices**

Tabla (4)

Quien la **puede** usar:

- Un cliente/usuario que vaya a definir requerimientos (características) de un software que necesite.
- Un desarrollador (interno/externo) que haga software “a la medida mediante proyecto.
- Un desarrollador que haga software “de paquete” que se venda masivamente.

## Clasificación de los requerimientos según el entorno tradicional

Además de la clasificación que mencionamos respecto a requerimientos del usuario y del sistema, existe otra clasificación que utilizaremos a nuestros fines de definir y analizar requerimientos, a saber:

**Requerimientos funcionales (RF)**

Describen la funcionalidad o los servicios que se espera que el sistema provea. Éstos dependen del tipo de software y del sistema que se desarrolle y de los posibles usuarios del software. Son los que definen las funciones que el sistema será capaz de llevar a cabo, sin tener en cuenta restricciones físicas. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Por ejemplo: Imprimir una entrada para el cine, o registrar una reserva de una entrada para una función del cine.

**Requerimientos no funcionales (RNF)**

Son aquellos que no se refieren a las funciones específicas del sistema, sino a las propiedades de éste que aseguran la calidad del software como: fiabilidad, respuesta en el tiempo, capacidad de almacenamiento, etcétera. Definen restricciones del sistema tales como la capacidad de los dispositivos y la representación de los datos utilizados en la interfaz del sistema.

Veamos un ejemplo de RNF del producto.

| Propiedad | Medida |
|---|---|
| **Rapidez** | Transacciones procesadas por segundo. |
| **Tamaño** | Mb de archivos. |
| **Facilidad de uso** | Tiempo de capacitación. Curva de aprendizaje. |
| **Fiabilidad** | Tiempo medio por falla. Tasa de ocurrencia de falla. Disponibilidad. |
| **Robustez** | Tiempo de reinicio luego de una falla. Probabilidad de corrupción de datos. |
| **Portabilidad** | Numero de sistemas objetivo. |

Tabla (5)

Otro ejemplo de RNF: el acceso al sistema debería ser con clave encriptada.

Existe una amplia clasificación que se deriva de los RNF, sin embargo, solo aplicaremos los RNF relacionados directamente con el producto ya que pueden ser utilizados para genera métricas de calidad del sistema construido.

Los RNF se clasifican en:

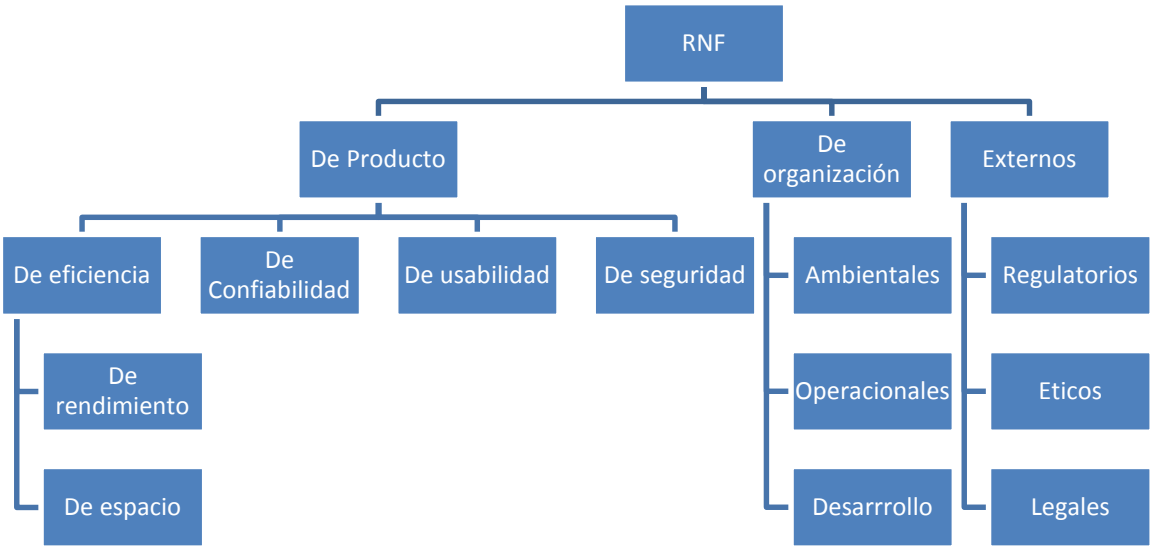
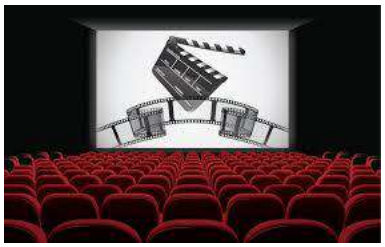



Figura (37)

## Análisis de un caso práctico "Cine"



### Objetivo del Producto

Administrar las programaciones de funciones de los cines del complejo, que serán la base para realizar reservas y ventas entradas. Brindar información resultante de las gestiones realizadas.

### Requerimientos Funcionales (Alcances)

- Administración de Cines y Salas (RG)
- Administración de Películas (RG)
- Administración de información relacionada a la película. (RG)
- Administración de Promociones para Ventas de entradas (RG)
- Administrar tarjetas de crédito. (RG)
- Administración de Usuarios y perfiles (RG)
- Administración de Permisos de Acceso al sistema (RG)
- Administración de precios y horarios de funciones (RG)
- Gestión de Programación de Funciones. (RG)
- Gestión de Reserva de lugares para funciones. (RG)
- Gestión de Venta de Entradas (RG)
- Gestión de Cobro de entradas con tarjeta de crédito. (RG)

**Requerimientos No Funcionales (Arquitectura)**

- Se debe plantear una interfaz que permita intercambiar información bidireccional con la máquina expendedora. (producto)
- Es necesario controlar concurrencia en la venta de entradas para cada uno de los puntos de venta. (producto)
- El producto permitirá la reserva de entradas vía web. (producto)
- El producto permitirá la consulta de horarios de funciones vía web. (producto)
- El tiempo de respuesta para la venta de entradas no debe superar los 5 minutos. (producto)
- Se debe mostrar la calificación de la película de acuerdo a la ley de cine regulada por el INCAA(ne
- gocio/externa)

## ¿Requerimientos y reglas de negocios son iguales?

Las reglas de negocio son el conjunto de acciones predefinidas por el negocio.

Una regla de negocio, por ejemplo, para una mutual, sería: si el cliente forma parte del staff de una empresa, entonces es un cliente preferencial. Por el contrario, esta no sería una regla de negocio: si el cliente tiene el ID número 1251251, entonces es preferencial.

Para el Cine, tenemos las siguientes reglas de negocio.

| Nro. | Nombre | Descripción |
|---|---|---|
| 1 | Anulación de entradas | Se acepta la anulación de venta de entradas siempre y cuando sea exactamente en el momento inmediato posterior a la ocurrencia de la venta de entradas. |
| 2 | Promociones | Las promociones pueden aplicarse a una película en particular o a uno o más tipos de función. En cualquier caso las promociones no son acumulables (no aplica más de una a la vez). |
| 3 | Forma de Cobro con expendedoras | La venta automática de entradas reservadas, utilizando máquinas expendedoras, se cobra únicamente con tarjeta de crédito habilitadas. |
| 4 | Retiro de entradas reservadas | Al momento de comprar entradas reservadas, no se permite selección parcial de los lugares reservados para las funciones. Se toma la totalidad de los lugares reservados vigentes, es decir que no hayan sido cancelados previamente. |
| 5 | Cancelación de Reservas | La cancelación de reservas sólo podrá realizarse telefónicamente. |

Tabla (6)

| Nº | Nombre | Regla de Negocio – Descripción |
|---|---|---|
| 1. | Capacidad de las salas | Los cines del complejo tienen varias salas, las cuales no necesariamente tienen la misma capacidad (cantidad de butacas). |
| 2. | Programación de funciones | La programación tiene un período de vigencia y es la que determina las películas a proyectarse durante ese período y los horarios para cada función de cada sala, para todos los días que forman el período de la programación, para cada uno de los cines que integran el complejo. |
| 3. | Habilitación de funciones | La programación diseña las funciones y sus horarios, quedando las mismas inhabilitadas. Es necesario habilitar las funciones para permitir la reserva y/o venta de entradas para las mismas. |
| 4. | Entradas para venta | El complejo vende entradas para las funciones que están exhibiéndose en ese momento o para funciones posteriores. |
| 5. | Anulación de entradas | Se acepta la anulación de venta de entradas siempre y cuando sea exactamente en el momento inmediato posterior a la ocurrencia de la venta de entradas. |
| 6. | Determinación del Precio de la entrada | Los tipos de entradas y los días y horarios de proyección son los que determinan el precio de la entrada. Las funciones admiten ciertos tipos de entradas y otros no, dependiendo de factores como: horarios, calificación de las películas, etc. Por ejemplo: si una película está calificada como para mayores de 16 años, para esa función no se pueden vender entradas de TIPO = MENOR.<br>Además pueden variar para cada cine, es decir una entrada del mismo tipo de entrada y para el mismo tipo de función puede tener un precio en un cine y otro precio en otro cine, del complejo. |
| 7. | Promociones sobre las entradas para funciones | Las promociones pueden aplicarse a una película en particular o a uno o más tipos de función. En cualquier caso las promociones no son acumulables (no aplica más de una a la vez). |

Figura (38)

## Requerimientos en la metodología ágil

Como hemos visto, las metodologías agiles se basan en gestionar el proceso y esfuerzo para realizar un proyecto. Por lo que se deduce que resulta necesario, mantener una comunicación sólida, completa, y mejor aún, se torna necesario el contacto cara-a-cara entre los interlocutores.

En un esfuerzo orientado a que esas conversaciones existan y por resolver de la mejor forma posible los problemas asociados a la identificación de los requerimientos, es que Mike Cohn creó la técnica de Historias de Usuario o User Stories.

## Historias de Usuario

Una **historia de usuario** es una descripción corta de una funcionalidad, valuada por un usuario o cliente de un sistema. Las historias de usuario suelen expresarse en una sencilla frase, y estructurarse de la siguiente manera:

**COMO <rol del usuario>**
**QUIERO < funcionalidad>**
**PARA <beneficio esperado, valor para el negocio>**

Veamos cada ítem:

"**Como [perfil/rol]**": ¿para quién lo estamos creando? No solo buscamos un cargo, sino que buscamos el perfil de una persona. Por ejemplo: Pablo. Todos los miembros del equipo deben comprender quién es Pablo. Es muy probable que hayamos entrevistado a muchos Pablo pero realmente debemos entender cómo trabaja esa persona, cómo piensa y cómo se siente. Tenemos empatía hacia Pablo.

"**Quiere**": aquí describimos cuál es su intención, y no las funciones que utiliza. ¿Qué es lo que intenta conseguir realmente? Esta afirmación no debería estar condicionada por ningún aspecto relativo a la implementación; si estás describiendo cualquier parte de la interfaz de usuario y no el objetivo del usuario, es que no has entendido de lo que se trata.

"**Para que**": ¿cómo se adapta su deseo inmediato de hacer algo al conjunto global? ¿cuál es la ventaja general que está intentando conseguir? ¿cuál es el gran problema que necesita resolverse?

Ejemplo:

Como Pablo, quiero invitar a mis amigos para que podamos disfrutar de este evento juntos.

Como gestor, quiero poder comprender el progreso de mis compañeros para poder informar mejor de nuestros éxitos y fracasos.

| | |
|---|---|
| **Loguear delivery**<br>Como delivery quiero loguearme para poder visualizar los pedidos de clientes.<br>***Notas***:<br>*Datos para login: nombre, apellido, teléfono, celular, dominio, número de moto, central* | **1** |
| ***Pruebas de usuario***:<br>• Probar ingresar los datos completos cuando el delivery se encuentra asociado a una central.<br>• Probar ingresar los datos cuando el delivery no se encuentra asociado a una central. | |

Las historias encajan perfectamente en marcos ágiles como Scrum y Kanban. En Scrum, las historias de los usuarios se añaden a los Sprints y se "queman" a lo largo del sprint. Los equipos Kanban introducen las historias de usuario en su backlog y las hacen avanzar a través de su flujo de trabajo. Es este trabajo sobre las historias de usuario lo que ayuda a los equipos de Scrum a mejorar en la estimación y planificación del sprint, lo que conduce a un pronóstico más preciso y a una mayor agilidad. Gracias a las historias, los equipos de Kanban aprenden a gestionar el trabajo en curso (WIP, del inglés "work in progress") y pueden perfeccionar aún más sus flujos de trabajo.

## ¿Cómo escribir historias de usuario?

Podemos tomar en cuenta los siguientes aspectos:

- Definición de "Hecho": La historia se suele considerar "hecha" cuando el usuario puede completar la tarea designada, pero asegúrate de definirla.
- Designa las subtareas o tareas: decide cuáles son los pasos específicos que deben completarse y quién es el responsable de cada uno de ellos.
- Perfiles de usuario: ¿para quién? Si hay varios usuarios finales, piensa en crear varias historias.
- Pasos en orden: escribe una historia para cada paso de un proceso más amplio.
- Ten en cuenta los comentarios: habla con tus usuarios y detecta cuál es el problema o la necesidad según su perspectiva. No hay necesidad de adivinar las historias cuando puedes obtenerlas de tus clientes.

## Elementos de una historia de usuario

Dependiendo del nivel de avance en el tema, vamos a completar algunos o todos los campos de las HU.

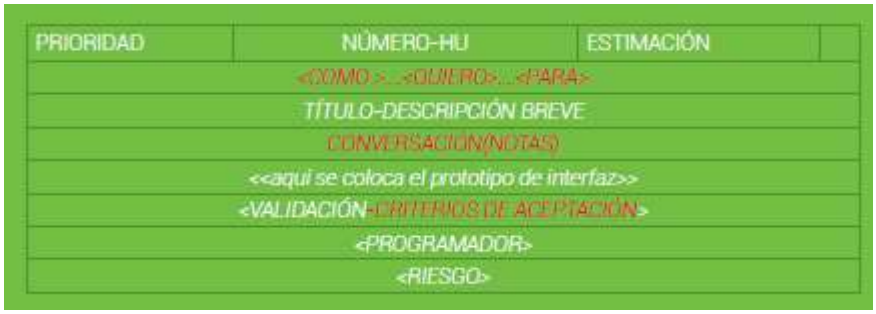



Figura (39)

- ***Numero de HU***: identifica una historia de usuario.
- ***Prioridad***: define la prioridad de al HU según una técnica, por ejemplo Moscow. La define el product owner. Puede ser: alta/media/baja.
- ***Titulo/Descripción breve***: sigue el formato <como> <quiero> <Para>.
- ***Estimación***: es un valor que indica el esfuerzo requerido para completar la HU.
- ***Conversación***: la conversación surge de aclarar dudas con el Product Owner en una reunión personal. SE puede agregar un prototipo de interfaz.
- ***Validación***: es una lista de criterios de aceptación que debe cumplir la HU para pasar al estado de “Done” o terminada.
- ***Programador***: persona a cargo de al HU.
- ***Riesgo***: SE usa una escala según lo indica la gestión de riesgo. Ejemplo:Alto/medio/bajo.

## Características de una historia de usuario

El método INVEST descrito por Bill Wake nos ayuda a escribir historias de usuario y consiste en cumplir las siguientes características:

1. ***Independiente***: una historia debería ser independiente de otras. La dependencia entre las historias hace que sea más difícil planificar, priorizar y estimar. A menudo, se puede reducir las dependencias haciendo una combinación de historias, o partiendo historias de forma diferente.

2. ***Negociable***: una historia de usuario es negociable. La "tarjeta" de la historia es tan sólo una descripción corta que no incluye detalles. Los detalles se trabajan durante la etapa de "Conversación". Una tarjeta con demasiados detalles limita la conversación con el cliente.

3. ***Valiosa***: cada historia tiene que tener valor para el cliente (para el usuario o para el comprador). Una forma muy buena de generar historias valiosas es hacer que el cliente las escriba. Una vez que el cliente se dé cuenta que la historia no es un contrato y es negociable, van a sentirse mucho más cómodos para escribir historias.

4. ***Estimable***: los desarrolladores necesitan poder estimar una historia de usuario para permitir que se pueda priorizar y planificar la historia. Los problemas que pueden impedirle a los desarrolladores estimar una historia son: falta de conocimiento del dominio en cuyo caso se necesita más Negociación / Conversación; o si la historia es muy grande en cuyo caso se necesita descomponer la historia en historias más pequeñas.

5. ***Pequeña***: una buena historia debe ser pequeña en esfuerzo, generalmente representando no más de 2-3 personas/semana de trabajo. Una historia que es más grande va a tener más errores asociados a la estimación y alcance.

6. ***Testeable***: una historia necesita poder probarse para que ocurra la etapa de "Confirmación". Recordemos que desarrollamos aquello que no podemos probar. Si no podemos probarlo, nunca vamos a saber si lo terminamos. Un ejemplo de historia no testeable sería: "el software tiene que ser fácil de usar".


Figura (40)

## Niveles de abstracción: estructura jerárquica de las historias de Usuario

Al final, la historia de usuario puede ser tan pequeña que no genere valor para el usuario o tan grande que no pueda ser concluida en un sprint o ciclo de desarrollo.

Al desarrollar una historia de Usuario, esta no debe tardar más de tres o cuatro días, de forma que pueda ser parte de un sprint. Incluso es bueno que una historia de Usuario pueda ser controlada dentro de una semana de trabajo; de esta forma al realizar las reuniones diarias se podrá identificar el avance e identificar posibles riesgos de no terminarla y corregirla durante la semana.

Las historias de usuario son también los componentes básicos de los marcos ágiles más grandes, como las épicas y las iniciativas. Las épicas son grandes cuerpos de trabajo divididos en un conjunto de historias, y varias épicas constituyen una iniciativa. Estas estructuras más grandes garantizan que el trabajo diario del equipo de desarrollo (las historias) contribuya a los objetivos organizacionales incorporados en las épicas y las iniciativas.

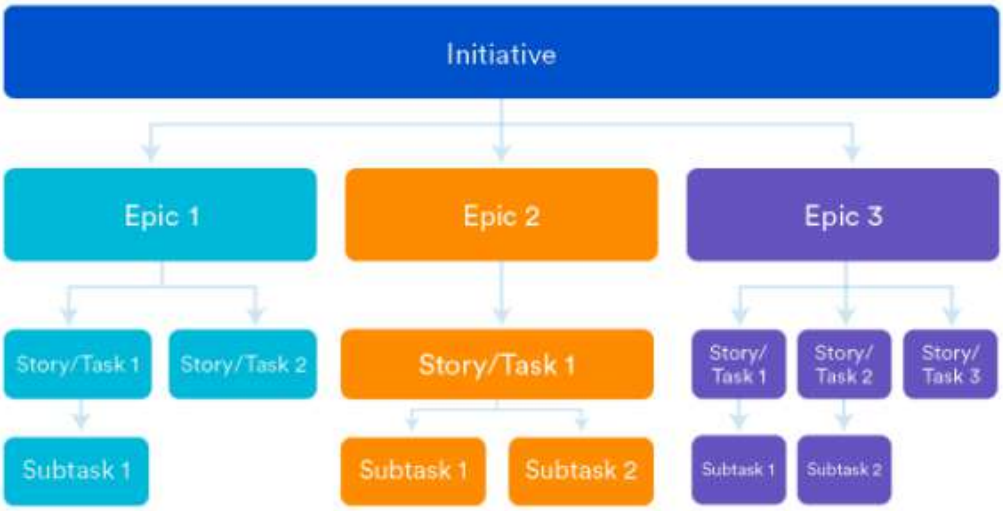



Figura (41)

Desde el punto de vista de la relación con el tiempo, podemos decir que los Temas se desarrollan en periodos largos como: años; mientras que las Épicas requieren meses; las feature semanas y finalmente las HU días.

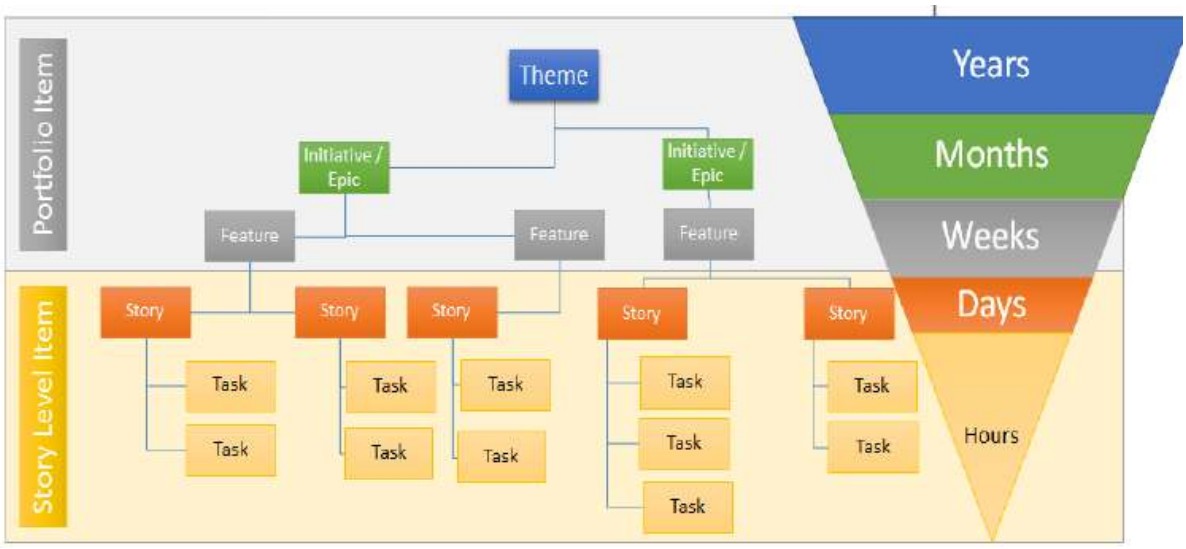



Figura (42)

Cabe mencionar que casi siempre es posible dividir una historia grande en historias más pequeñas. Simplemente hay que asegurarse de que las historias pequeñas siguen representando entregables con valor de negocio. En Scrum, y en las metodologías ágiles en general, una pila del producto puede contener tanto historias de usuario como épicas tal que, de la mitad de la pila hacia el final de la lista, donde está lo menos prioritario, es el lugar de las épicas. A medida que nos acercamos a los elementos más prioritarios, el detalle debe aumentar, por tanto, las historias de usuario son los elementos que deben de encabezar la lista.

Así, podemos establecer una jerarquía de HU como se muestra en la figura (43).

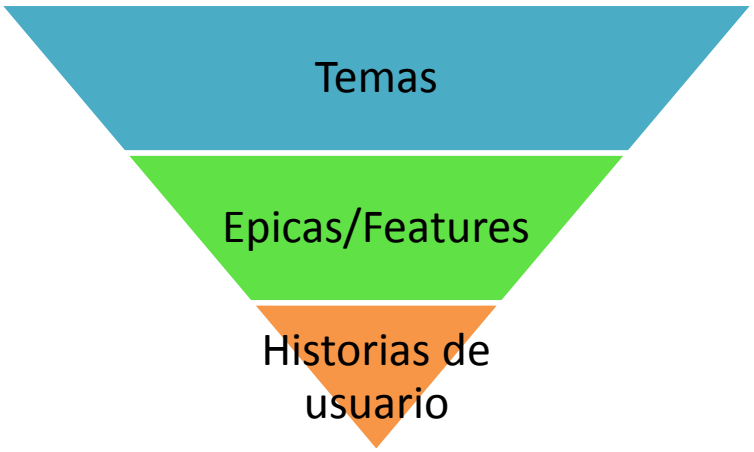



Figura (43)

Una Épica puede ser vista como una historia de usuario de alta complejidad (es decir, como petición de negocio de alto nivel y complejidad, que generalmente no es clara). Para elaborarla se puede requerir un esfuerzo muy grande y no puede ser cubierta en un sprint.

Una épica debido a su gran tamaño es difícil de estimar y de acometer, así que deben descomponerse en historias de usuario. Además, implementar una épica suele llevar dos o más Sprints.

Ejemplos de Épicas.

- "Como usuario quiero poder registrarme y tener una cuenta de usuario para poder acceder a mi información de usuario y de los servicios que uso".
- "Como usuario quiero un servicio de 'lista de deseos' para poder guardar los artículos que me gustan".

El punto importante a tener en cuenta es que cuando se depuran las historias de usuarios, el equipo debe involucrar al Product Owner y negociar las funcionalidades con el él si estas se vuelven muy complejas. La negociación lleva al equipo a dividir una historia de usuario en otras de menor tamaño que de igual forma brindan valor al usuario.

Veamos un ejemplo:

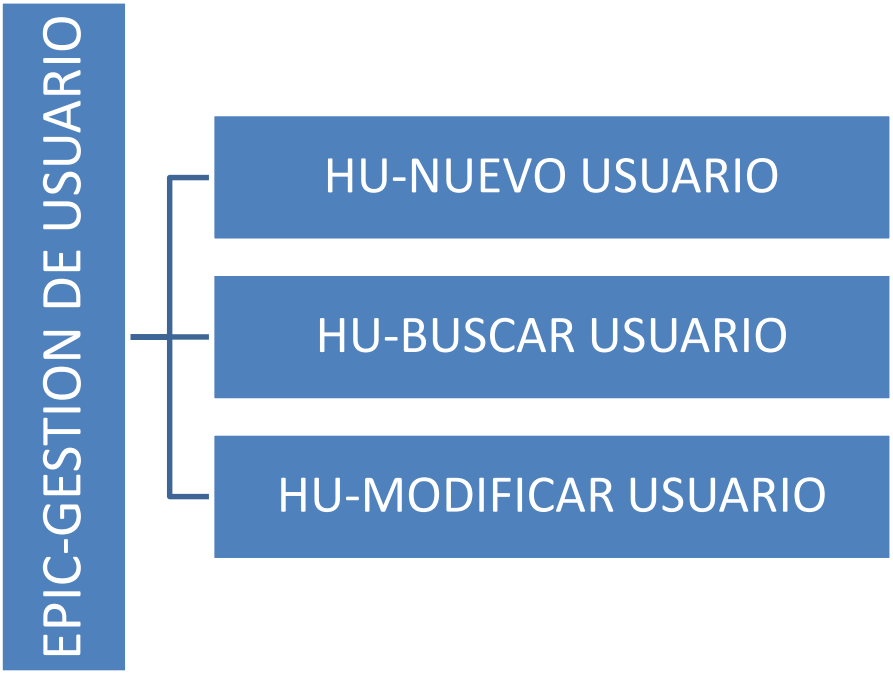



Figura (44)

Una pregunta muy válida es: ¿cuál es la diferencia entre historias y tareas?

La diferencia es muy simple. Las historias son entregables de los que el Dueño de Producto se preocupa. Las tareas son no-entregables o aspectos de los que el Dueño de Producto no se preocupa.

Veamos un ejemplo de división de una historia en tareas.

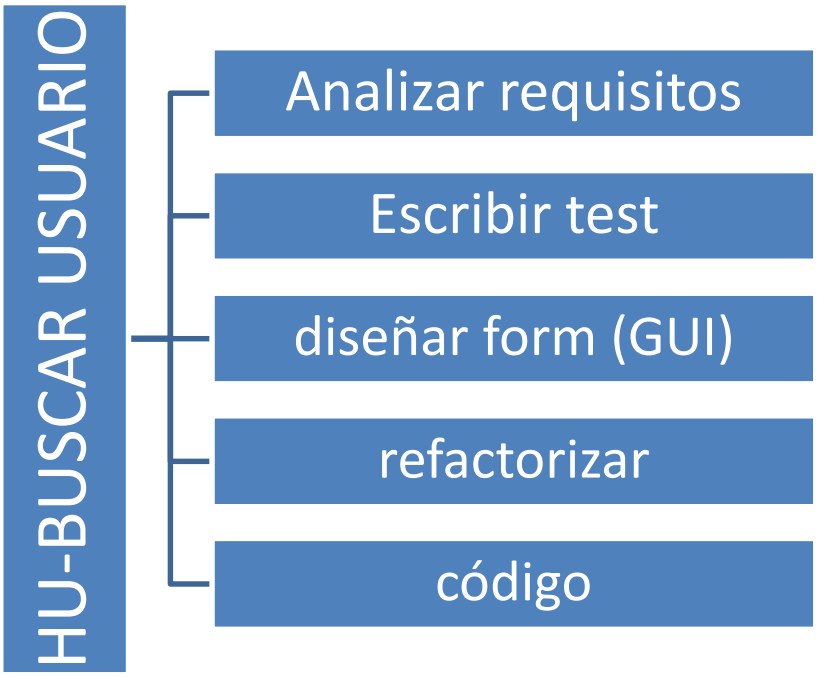



Figura (45)

Para historias que se entienden bien, es tan fácil hacer esto desde el principio como hacerlo más tarde.

Este tipo de división frecuentemente revela trabajo adicional que hace que las estimaciones suban, con lo que se consigue un plan de Sprint más realista. Este tipo de división desde el principio hace que los SCRUM diarios sean notablemente más eficientes.

Incluso si la división es inexacta y cambia una vez que empezamos, todas las ventajas anteriormente mencionadas siguen siendo válidas.

## Historias de usuario: técnicas, canónicas y spikes

Historias de usuario técnicas: también llamadas elementos no-funcionales para algunos autores o Sprint (0) para otros- son un asunto complejo. Se trata de cosas que deben hacerse pero que no son un entregable ni están directamente relacionadas con ninguna historia específica. Tampoco son de valor inmediato para el Dueño de Producto.

Por ejemplo: Instalar un servidor.

¿Por qué debe hacerse?: porque ahorra cantidades inmensas de tiempo a los desarrolladores y reduce el riesgo de problemas al final de la iteración.

Escribir una descripción general del diseño:

¿Por qué debe hacerse?: porque los desarrolladores olvidan constantemente el diseño general, y entonces escriben código inconsistente. Necesitan una visión global documentada para mantener a todo el mundo en la misma línea de diseño.

Refactorizar la capa de acceso a datos: ¿Por qué debe hacerse?: porque la capa de acceso a datos se ha vuelto realmente desordenada y le está costando tiempo a todo el mundo debido a la confusión y los errores innecesarios.

Las Historias Técnicas ¿son historias en el sentido normal o son tareas que no están conectadas a ninguna historia específica?,¿quién las prioriza?, ¿debería involucrarse el Dueño de Producto en estos asuntos?

Historias de usuario canónicas: son historias de usuario que sirven de referencia, para hacer la estimación de otras historias de usuario por puntos de historia.

Por ejemplo:

| | |
|---|---|
| ***Registrar tipo de programa*** **[use story canónica]**<br>COMO responsable de programación QUIERO poder registrar un tipo de programa así se agregan a la programación vigente para permitir actualizar la programación. | **1** |
| ***Pruebas de usuario***<br>• Probar registrar un nuevo tipo de programa(pasa)<br>• Probar registrar un tipo de programa existente(falla) | |

Historias de usuario de tipo Spikes: es un tipo especial de historia utilizado para quitar riesgo o incertidumbre de una user Story. Se clasifican en: técnicas y funcionales

| Técnicas | Funcionales |
|---|---|
| Utilizadas para investigar enfoques en el dominio de la solución. | Utilizadas cunado hay cierta incertidumbre respecto de COMO el usuario interactuara con el sistema. |
| Cualquier situación en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad en un tiempo fijo | Usualmente son mejor evaluadas con prototipos para obtener retroalimentación de los usuarios involucrados. |

Tabla (7)

Por ejemplo: ***COMO*** <Cliente> ***QUIERO*** <ver mi consumo diario de energía en un histograma para poder comprender y comparar mi consumo pasado, presente y proyectado.

En este caso se puede crear dos spikes:

***Spike Técnica***: investigar cuanto tiempo se requiere para actualizar un display de un cliente al uso actual. Conectividad, ancho de banda y si los datos se actualizan en formato push o pull.

***Spike Funcional***: crear un prototipo de histograma en el portal web y obtener retroalimentación de algunos usuarios respecto al tamaño, diseño y aspecto gráfico.

Salvo que la spike sea sencilla y con una solución rápida, en ese caso puede incluirse en la misma iteración. Caso contrario debe implementarse en una iteración separada. Utilizar spikes como última opción.

## Pruebas y criterios de aceptación en las Historias de usuario

Los criterios de aceptación definen los requisitos del Product Owner sobre cómo debe comportarse la aplicación para que una determinada acción se pueda llevar a cabo. Generalmente ayudan al equipo de desarrollo a responder a las preguntas:

- ¿He construido el producto correcto?
- ¿He construido el producto correctamente?

Los criterios de aceptación deben describir siempre un contexto, un evento y la respuesta o consecuencia esperada del sistema. La forma más utilizada para describir los criterios de aceptación es conocida como **Given-When-Then**.

- Ejemplo 1: Loguin de un cliente.

| **HU - LOGUEAR CLIENTE** |
|---|
| **<Dado>** un usuario que aún no se ha identificado en el sistema<br><br>**<Cuando>** intenta acceder a alguna funcionalidad<br><br>**<Entonces>** se le redirige automáticamente a la página de login para que pueda loguearse. |

Aunque describen comportamiento de la aplicación, se utiliza siempre un lenguaje de negocio, no técnico. En el ejemplo anterior usamos “cuando intenta acceder a alguna funcionalidad” y no “cuando hace clic en alguna opción de menú que requiere sesión”.

- Ejemplo 2:

| | |
|---|---|
| <COMO> cliente <QUIERO> pagar mi deuda del servicio de rentas con tarjeta <PARA> aprovechar las oportunidades de los planes vigentes que ofrecen las tarjetas de crédito. | **5** |
| ***Nota***:<br>Se acepta Visa, Mastercard y Cordobesa en compras superiores a $1500 y se pide código de verificación que figura al dorso de la tarjeta. | |
| ***Criterios de aceptación***:<br>1. Probar con Visa, Mastercard y Cordobesa(pasa)<br>2. Probar con cantidad de números que identifica una tarjeta (pasa)<br>3. Probar con cantidad diferente de números que identifica una tarjeta (falla)<br>4. Probar con tarjetas vencidas (falla)<br>5. Probar con Naranja (falla)<br>6. Probar con montos mayores a $1500 (pasa)<br>7. Probar con montos menores a $1500 (falla) | |

- Ejemplo 3

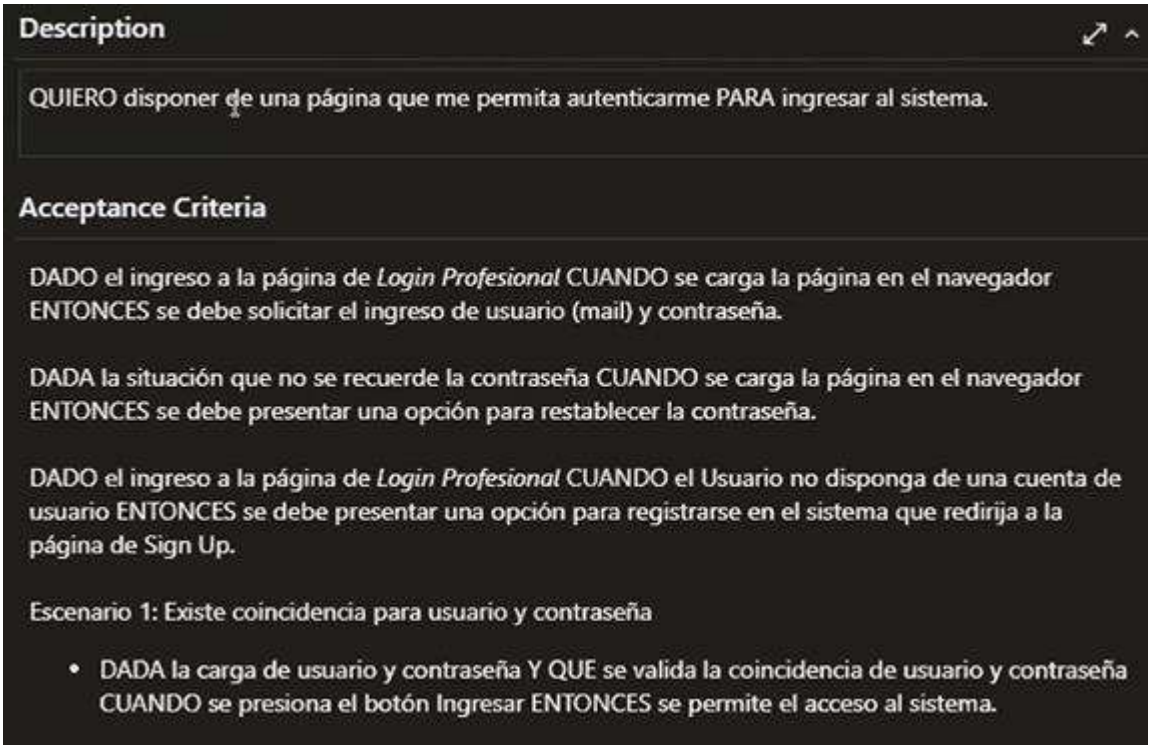



Figura (46)

- Ejemplo 4: el formulario de búsqueda de Google.

1. **Dada** una petición a la página de búsqueda **cuando** se carga la página en el navegador **entonces** el cursor se desplaza al cuadro de búsqueda para que el usuario pueda comenzar a teclear de inmediato
2. **Dado** que el usuario está en el cuadro de búsqueda y el usuario ha hecho login con su cuenta de Google **cuando** el usuario pulsa sobre el mismo cuadro de búsqueda **entonces** se le muestra una lista con sus últimas búsquedas
3. **Dado** que el usuario está en el cuadro de búsqueda **cuando** el usuario teclea algo **entonces** se le muestra una lista con sugerencias de búsqueda relacionadas con lo que ha tecleado
4. **Dada(s)** una(s) palabra(s) en el cuadro de búsqueda **cuando** el usuario acepta **entonces** se le redirige a la página de resultados del texto introducido
5. **Dada(s)** una(s) palabra(s) en el cuadro de búsqueda **cuando** el usuario selecciona la opción “me siento con suerte” **entonces** se le redirige a la página del primer resultado de la búsqueda
6. **Dado** que el usuario está en la página de búsqueda **cuando** el usuario selecciona la opción “Búsqueda por voz” **entonces** se le redirige a la página de reconocimiento por voz y una vez interpretado el texto se le devuelve a la página de resultados del texto interpretado

Si fuera una única historia de usuario, seguramente estaríamos delante de una épica. Podríamos utilizar los criterios de aceptación para dividir ésta en historias de usuario más pequeñas. Podríamos hacer una historia de usuario con los criterios 1 y 4, y el resto constituir cada uno historias de usuario independientes.

## Definición of Done (DoD)

Cada equipo Scrum tiene su propia definición, pero la “definición de hecho” puede ser una simple lista de actividades (comentarios de codificación, pruebas unitarias, documentos de diseño, etc.) o simplemente una serie de acuerdos que agregan valor verificable y demostrable al producto. Es un entendimiento compartido de lo que significa que una tarea está terminada. El Definition of done es crucial para un equipo altamente funcional en Scrum. Que el equipo cumpla los criterios del DoD asegurará que se están entregando tareas que están realmente hechas, no sólo en términos de funcionalidad sino también en términos de calidad.

El criterio de aceptación se define para cada historia de usuario (o item del Product Backlog) y se aplica individualmente a cada una de ellas. La Definición of Done se aplica a todas las historias de usuario.

Mientras que los criterios de aceptación son únicos para los User Stories individuales, el criterio de Done es un conjunto de reglas que se aplican a todas los User Stories en un determinado Sprint.

¿Quién actualiza la Definición of Done?

El DoD puede cambiar con el tiempo y se actualiza siempre que sea necesario, pero nunca en medio de un Sprint. Generalmente se utiliza el Sprint Retrospectiva para actualizar la Definición of Done.

Ejemplos de DoD

- Todas las pruebas unitarias y funcionales son correctas.
- Todos los criterios de aceptación se cumplen.
- OK del equipo: UX, desarrollador, Product Owner, etc.
- Pruebas en dispositivos/navegadores pasada.
- Pruebas de rendimiento pasadas.
- Se han corregido todos los bugs.
- Entorno preparado para la subida a producción.

## ¿Cuáles son los requisitos más comunes definidos en el Definition of Done?

Algunos de los requisitos más comunes son los siguientes:

- La tarea debe cumplir con los estándares de calidad definidos.
- La tarea debe haber sido validada y verificada.
- La tarea debe haber sido puesta en producción o haberse hecho la petición de pase a producción.

## Consideraciones finales acerca de las historias de usuario

Diferir el análisis detallado tan tarde como sea posible, lo que es justo antes de que el trabajo comience.

Las User Stories son descripciones breves de funcionalidad relevante para el cliente.

Las HU no son requerimientos; son marcadores para conversaciones más detalladas y análisis que deberá ocurrir conforme esas historias vayan implementándose.

## La propuesta de sistema y la Práctica Supervisada

Para dar comienzo al proyecto de desarrollo de un software, se necesita presentar una propuesta de sistema como solución a un caso objeto de estudio; ya sea una organización u otro caso práctico que permita un abordaje metodológico mediante algunos de los modelos estudiados en esta unidad.

En el caso de la asignatura de Práctica Supervisada de la TUP, se trabajará con un proyecto de desarrollo de software que será gestionado durante el cursado de dicha asignatura.

Como cualquier sistema de información, lo primero era definir el objetivo del sistema, tal que dicho objetivo será una descripción breve y resumida de las gestiones o suma de todas las funciones o requisitos que se convertirán en las características del producto final.

La propuesta también incluye el alcance del sistema, que será la descripción de todas las funciones del sistema a desarrollar en la práctica supervisada organizados como requerimientos funcionales globales y detallados.

Por último, se deberá tomar en cuenta aspectos relacionados con los requerimientos no funcionales tanto para sistemas basados en una organización como propuestas de inventiva propia del estudiante.

Las técnicas de relevamiento a utilizar para comenzar el proyecto son: entrevistas, tormenta de ideas, concept mapping, sketches y storyboards para aplicaciones web, casos de uso, cuestionarios y chek list entre otras. Estas técnicas mencionadas no forman parte de los contenidos a desarrollar en esta asignatura.

### Formulación de la propuesta con requerimientos funcionales (Formulario 1)

No existe un formato formal para presentar una propuesta de sistema, de hecho, hemos visto a lo largo del estudio de esta unidad que podemos referirnos a los requerimientos como historias de usuario si se trata de un proyecto con fines de desarrollo ágil y se representa como la pila de producto que no tiene un formato particular; son que se ajusta al uso de herramientas para llevar adelante dicha gestión. No obstante, cabe mencionar que atendiendo a los lineamientos de la asignatura Práctica Supervisada, vamos a utilizar la siguiente plantilla.

| Etapa 1: PRESENTACION DEL PROYECTO | |
|---|---|
| **1-Datos del alumno** | |
| Apellido y nombre: Juan Pérez | Legajo:21565/25 |
| Nombre del proyecto: GES (es un nombre de fantasía que personaliza tu proyecto y debe ser único) | |
| Correo institucional: jperez@frc.utn.edu.ar (debe ser el corre de la universidad) | |
| **2-Datos de la PS** | |
| ☐Se desarrolla en una empresa | ☐Es inventiva del propio alumno |
| Objetivo del proyecto (Propósito del sistema) | |
| **Brindar información para [gestionar clientes], [gestionar reclamos] y [gestionar ventas minoristas] que contribuyan al proceso de toma de decisiones de las áreas mencionadas generando información estadística para la trazabilidad de los reclamos de clientes.** | |
| Limite(desde/hasta) | |
| Desde que se registra un cliente Hasta que se generan los informes estadísticos de reclamos | |
| Alcance | |
| Requerimientos funcionales (R.F) | |
| ***Registrar cliente*** | |
| ***Registrar venta*** | |
| ***Registrar reclamo*** | |
| ***Registrar modificación de venta*** | |
| ***Genera reporte estadístico de reclamos*** | |
| ***Etcétera*** | |
| Requerimientos no funcionales (R.N.F) | |
| Las consultas de montos vendidos no deben superar los 60 segundos. | |
| El acceso al sistema se hace mediante clave y perfil de usuario. | |
| Etcétera. | |
| **3-Justificación** | |
| | |
| | |
| **4-Datos a completar por el docente** | |
| Fecha: | Estado:☐ Aprobado ☐ Rechazado |

**Funciones del sistema**

## Formulación de la propuesta con historias de usuario (Formulario 1)

| Etapa 1: PRESENTACION DEL PROYECTO | |
|---|---|
| **1-Datos del alumno** | |
| Apellido y nombre: Juan Pérez | Legajo:21565/25 |
| Nombre del proyecto: GES (es un nombre de fantasía que personaliza tu proyecto y debe ser único) | |
| Correo institucional: jperez@frc.utn.edu.ar (debe ser el corre de la universidad) | |
| **2-Datos de la PS** | |
| ☐Se desarrolla en una empresa | ☐Es inventiva del propio alumno |
| Objetivo del proyecto (Propósito del sistema) | |
| **Brindar información para [gestionar clientes], [gestionar reclamos] y [gestionar ventas minoristas] que contribuyan al proceso de toma de decisiones de las áreas mencionadas generando información estadística para la trazabilidad de los reclamos de clientes.** | |
| | |
| Alcance | |
| Requerimientos funcionales (R.F) | |
| ***Epica: Gestionar Ventas*** | |
| ***HU -Registrar cliente*** | |
| ***HU -Registrar venta*** | |
| ***HU-Registrar reclamo*** | |
| ***HU-Registrar modificación de venta*** | |
| ***HU- Generar reporte estadístico de reclamos*** | |
| ***Etcétera*** | |
| Requerimientos no funcionales (R.N.F) | |
| Las consultas de montos vendidos no deben superar los 60 segundos. | |
| El acceso al sistema se hace mediante clave y perfil de usuario. | |
| Etcétera. | |
| **3-Justificación** | |
| | |
| | |
| **4-Datos a completar por el docente** | |
| Fecha: | Estado:☐ Aprobado ☐ Rechazado |

**Funciones del sistema**

# ANEXO

## Participantes del proceso de desarrollo de software

En los equipos desarrollo de software, dependiendo del proyecto y su complejidad; se requieren distintas habilidades. Por ejemplo:

1- El ***programador***: será emprendedor y creativo, una persona capaz de inventar nuevas soluciones para las necesidades que crean a diario las nuevas tecnologías. Además, debe tener iniciativa empresarial y visión de futuro, alguien con capacidad de crear nuevos proyectos, así como de ejecutarlos correctamente. Tendrá altos conocimientos tecnológicos partiendo de lenguajes como C++, Java, Python, JavaScript,Angular,entre otros.

Además, es importante que sea experto en iOS, Android, FirefoxOS, Windows Phone, HTML5, node.js, MongoDB, Openstack, Linux y en diseño de interfaz de usuario.

También será experto en algoritmos, paradigmas y fundamentos del desarrollo de software. Dominará prácticas de programación, algoritmos, estructuras de datos, programación funcional, programación orientada a objetos y arquitecturas cliente servidor, protocolos TCP/IP y HTTP, sin olvidar bases de datos relacionales y no relacionales.

Un desarrollador, se clasifica en:

- ***Frontend***: se enfoca en el usuario, en todo con lo que podemos interactuar y lo que vemos mientras navegamos. ES todo lo relacionado con nuestra web y que busca causar una buena impresión y agradar al usuario, para lo cual utiliza HTML, CSS y JAVASCRIPT. Buena experiencia de usuario, inmersión y usabilidad, son algunos de los objetivos que busca un buen frontend y hoy en día existen una gran variedad de frameworks, preprocesadores y librerías que nos ayudarán en esta tarea. Para un frontend la creatividad es el recurso más valioso, ya que tendrá que tomar fuentes, colores, imágenes y todos los recursos de los cuales disponga para crear sitios agradables que se vean bien en todos los dispositivos y resoluciones.

- ***Backend***: está enfocado en hacer que todo lo que está detrás de un sitio web funcione correctamente. Toma los datos, los procesa y los envía al usuario, además de encargarse de las consultas o peticiones a la Base de Datos, la conexión con el servidor, entre otras tareas que debe realizar en su día a día. Cuenta con una serie de lenguajes y herramientas que le ayudan a cumplir con su trabajo como PHP, Ruby, Python, JavaScript, SQL, MongoDb, MySQL, etc, estos son usados para crear sitios dinámicos. Los programadores dedicados al backend tienen conocimientos específicos de bases de datos y frameworks para que la página se cargue correctamente en los servidores y los usuarios puedan navegar cómodamente, asegurando así buenos niveles de usabilidad e interacción cuyo objetivo es ofrecer la mejor experiencia de

usuario posible. Algunos de los frameworks que se manejan en el back-end son Vue.js o React.js, entre otros.

- ***FullStack***: En la actualidad encontramos desarrolladores Frontend con conocimientos de Backend y viceversa, conocidos como Full Stack Developers. Estos, tendrán que tener formación sobre hardware, comunicaciones, protocolos y sistemas operativos hasta nivel de aplicación.

Además de todas estas ***softskills***; es importante que tenga buena capacidad de comunicación, autoliderazgo, gestión del tiempo y del trabajo para dirigir y gestionar equipos. Para convertirse en un buen manager de desarrolladores, resulta indispensable que tenga eficiencia en la gestión de personas.

2 - **DBA** - Un administrador de base de datos (DBA) dirige o lleva a cabo todas las actividades relacionadas con el mantenimiento de un entorno de base de datos exitoso. Las responsabilidades incluyen el diseño, implementación y mantenimiento del sistema de base de datos; el establecimiento de políticas y procedimientos relativos a la gestión, la seguridad, el mantenimiento y el uso del sistema de gestión de base de datos; y la capacitación de los empleados en la gestión y el uso de las bases de datos. Se espera que un DBA se mantenga al tanto de las nuevas tecnologías y los nuevos enfoques de diseño.

3- El **diseñador UI/UX**: es el profesional que debe definir los lineamientos visuales, aplicando bases teóricas, resultados del análisis del comportamiento de los usuarios y buenas prácticas de la usabilidad y accesibilidad web. Su principal objetivo es configurar la mejor experiencia posible para el público objetivo de la aplicación. Se encarga de que la percepción y sensaciones que el uso de un producto o servicio deje en la mente de las personas sean las óptimas bajo cualquier punto de vista: ergonomía, facilidad de uso, eficiencia, etc.

**UX Design** (User Experience Design) o “Diseño de Experiencia de Usuario” es una filosofía de diseño que tiene por objetivo la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo. Toma forma como un proceso en el que se utilizan una serie de técnicas multidisciplinares y donde cada decisión tomada debe estar basada en las necesidades, objetivos, expectativas, motivaciones y capacidades de los usuarios.

| Competencias transversales | Competencias técnicas |
|---|---|
| • Adaptabilidad<br>• Análisis<br>• Comunicación<br>• Empatía<br>• Interés por aprender<br>• Mentalidad abierta para cuestionar ideas<br>• Observación<br>• Pensamiento crítico<br>• Saber escuchar | • Administración de proyectos<br>• Arquitectura de información<br>• Design thinking<br>• Habilidades lingüísticas y de escritura<br>• Nociones de diseño gráfico y comunicación visual para poder realizar wireframes y prototipos<br>• Nociones de programación (coding)<br>• Técnicas de investigación de usuarios y contextos |

4 - **Analista de sistemas**: es el profesional que se encarga entre otras cosas de la creación y de las revisiones de las apps, creando soluciones a los problemas que plantea el cliente. También proporciona opiniones sobre las modificaciones que se deben implementar en el software que se está usando dentro de un contexto determinado, planificando los cambios oportunos en un sistema operativo o bien, en las plataformas o aplicaciones que esté utilizando la empresa ola compañía en cuestión para la que trabaje. Es el encargado de encontrar una serie de necesidades o problemas en el ámbito de las Tecnologías de la Información y de la Comunicación eligiendo soluciones que pueden ser programadas o diseñadas.

5 - **Analista funcional**: es el vínculo de unión entre el usuario y el área informática de la empresa. Su misión consiste en elaborar el análisis funcional de nuevas aplicaciones para la organización, así como actualizar y mejorar las ya existentes; es decir, debe controlar, analizar y supervisar el desarrollo funcional de las aplicaciones informáticas, asegurando su correcta explotación y su óptimo rendimiento. Presta apoyo a los distintos usuarios; es decir, realiza una labor de asesoramiento y capacitación, con el fin de evitar cualquier problema que pueda surgir con los programas y obtener así el máximo rendimiento de los mismos.

Otras funciones son evaluar tanto la viabilidad técnica como la económica de los desarrollos de las aplicaciones que se han de ejecutar, y preparar y elaborar toda la documentación técnica y de usuario de cada aplicación.

6 - **Arquitecto**: Es la persona o personas con el suficiente conocimiento técnico del producto o servicio que se va a proporcionar como para dar una solución técnica a las necesidades del cliente. Es el responsable de crear durante el proceso de desarrollo de software la documentación que recoge los requisitos (siempre en comunicación con el grupo de comerciales y el cliente).

Durante la fase de desarrollo y testing será el que centralice las decisiones técnicas sobre los problemas que irán surgiendo en el transcurso del proyecto.

7 - **Tester**: Es la persona que se encargará de asegurar que los requisitos definidos por el arquitecto se cumplen en la implementación del producto o servicio. Deberá informar de todos los defectos encontrados en la fase de pruebas, pero también deberá de informar de todo lo que potencialmente puede ser un problema o mal entendido en la implementación.

## Bibliografía


- Kendall, K y Kendall, J. (2005) *Análisis y diseño de sistemas Sexta edición*. México. Editorial Pearson.
- Sommerville,I(2011). "*Ingeniería de software Novena edición"* México. Editorial Pearson.
- Xavi Sanchez (2020) "Metodología Lean Startup: aprender a crear tu empresa con más éxito ". Disponible en:
- Ken Schwaber y Jeff Sutherland (2017) "La Guía de Scrum". Ed.Scrum.org


**Atribución-NoComercial-SinDerivadas**

Se permite descargar esta obra y compartirla, siempre y cuando no sea modificado y/o alterarse su contenido, ni se comercializarse. Referenciarlo de la siguiente manera:

Universidad Tecnológica Nacional Regional Córdoba (2020). Material para la Tecnicatura en Programación Semipresencial de Córdoba. Argentina.