20

**Ingeniería De Software**

**Modelos de Procesos**

**Páez, Daniel M.**

Desde las primeras décadas en las que el software ha sido empleado para el desarrollo de soluciones para la industria, hasta las épocas actuales que brindan soluciones a la mayoría de los problemas que se presenta las necesidades, el proceso de desarrollo de software se tuvo que adaptar a las grandes demandas que a medida surgían. Siendo este cambio a pensar que el grupo que desarrolla el software no solo lo integran los desarrolladores y representantes de los interesados, sino que aquellos que presten interés sobre el ámbito donde el software se desenvolverá forman parte importante al momento de recaudar las necesidades a satisfacer por el software.

El siguiente documento presenta los diferentes modelos que fueron surgiendo a lo largo del tiempo en la industria de desarrollo de software.

Tabla de contenido

[Procesos del Software 3](#_Toc40712115)

[Modelos del Proceso Del Software 3](#_Toc40712116)

[Modelo En Cascada 3](#_Toc40712117)

[Desarrollo Evolutivo 4](#_Toc40712118)

[Ingeniería de Software Basada En Componentes 5](#_Toc40712119)

[Iteración de Procesos 6](#_Toc40712120)

[Entrega Incremental 7](#_Toc40712121)

[Desarrollo En Espiral 7](#_Toc40712122)

[Proceso Unificado 8](#_Toc40712123)

[Metodologías Agiles 10](#_Toc40712124)

[SCRUM 12](#_Toc40712125)

[Componentes de Scrum 13](#_Toc40712126)

[Elementos de Scrum 15](#_Toc40712127)

[Cuadro Comparativo de Modelos de Procesos de Software 16](#_Toc40712128)

[Conclusiones 18](#_Toc40712129)

[Bibliografía 18](#_Toc40712130)

# Procesos del Software

El Proceso de software es un conjunto de actividades que conduce a la creación y desarrollo de un proyecto software. Este proceso es complejo y dependen de las personas que toman decisiones para desarrollarlo.

No existe un único proceso ideal, y varias organizaciones han desarrollado su propio enfoque para desarrollarlos. Pero para procesos más complicados se utiliza un proceso estructurado, para sistemas de negocio, cuyos requerimientos varían, es probable que un proceso flexible y ágil fuera más efectivo.

A pesar de ser varios, existen actividades fundamentales que son comunes a todos:

* **Especificación de Software:** definir la funcionalidad y las restricciones del software.
* **Diseño e Implementación del Software:** se produce el software que cumpla con la especificación.
* **Validación del Software:** se válida para asegurar que el software cumpla con lo que el cliente desea.
* **Evolución del Software:** el producto va cambiando para cubrir las necesidades cambiantes del cliente.

Los procesos del Software se pueden mejorar con la estandarización del proceso donde la diversidad de los procesos del software en una organización sea reducida. Esto lleva a que se mejore la comunicación y a la reducción del tiempo de formación, y ayuda al proceso de automatización más económica.

# Modelos del Proceso Del Software

Un modelo del Proceso del Software es una representación abstracta de un proceso del software. Cada modelo representa un proceso de desarrollo desde una perspectiva particular. Estos modelos no son descripciones definitivas de los procesos del software, se usan para explicar diferentes enfoques para el desarrollo de software.

Algunos modelos de proceso conocidos son:

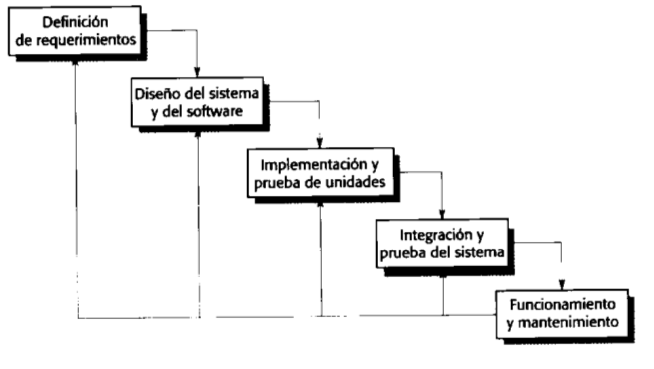
* **Modelo de Cascada.**
* **Desarrollo Evolutivo.**
* **Ingeniería de Software Basada en Componentes.**

## Modelo En Cascada

Fue el primer modelo en ser publicado para el desarrollo de software. Debido a la cascada entre una fase y otra, se lo conoce como ciclo de vida del software.

Posee las siguientes etapas:

1. **Análisis y Definición de Requerimientos**: serán definidos los servicios, restricciones y metas del sistema a través de las consultas con los usuarios. Estos se definen en detalle y sirven como especificación del sistema.
2. **Diseño del Sistema y del Software**: se dividen los requerimientos en sistema hardware y software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
3. **Implementación y Prueba de Unidades**: el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
4. **Integración y Prueba del Sistema**: los programas se integran y prueban como un sistema completo para asegurar que cumplen los requerimientos del software. Después, se lo entrega al cliente.
5. **Funcionamiento y Mantenimiento**: es la fase más larga, el sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.



El resultado de cada fase es uno o más documentos aprobados. La siguiente fase no empezara hasta que la anterior haya finalizado.

El proceso de Software no es lineal, sino que implica una serie de iteraciones de las actividades de desarrollo.

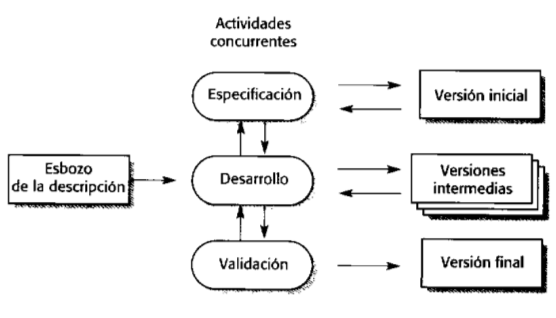
En el modelo de cascada las iteraciones son costosas e implican rehacer el trabajo en cada etapa. Se producen congelamientos de partes del desarrollo, como las especificaciones, y se avanzan con las siguientes etapas. Esto puede implicar que el sistema no haga lo que los usuarios deseen. Se descubren errores y omisiones en los requerimientos originales en la fase final del ciclo de vida (errores de programación, de diseño), por lo que se identifica la necesidad de una nueva funcionalidad, por ende, el sistema debe evolucionar para mantenerse útil. Realizar estos cambios implica repetir etapas previas del proceso.

* **Ventajas del modelo de cascada:** la documentación se produce en cada fase y este cuadra con otros modelos de procesos de ingeniería.
* **Desventajas:** inflexibilidad al dividir el proyecto en etapas distintas, se deben hacer compromisos en las etapas iniciales, lo que hace difícil responder a los cambios en los requerimientos del cliente.

## Desarrollo Evolutivo

Se basa en desarrollar una implementación inicial, exponiéndola a los comentarios de los usuarios y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado.

Las etapas de especificación, desarrollo y validación en vez de separarse se entrelazan, con una rápida retroalimentación entre ellas.



Hay dos tipos de desarrollos evolutivos:

* **Desarrollo Exploratorio:** el objetivo es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. Se empieza con las partes del sistema que mejor se comprendieron, luego evoluciona agregando atributos nuevos propuestos por el cliente.
* **Prototipos Desechables:** el objetivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

Este enfoque suele ser más efectivo que el modelo en cascada para la producción de sistemas, satisfaciendo las necesidades inmediatas de los clientes.

Ventajas:

* La especificación se puede desarrollar de forma creciente. Si los usuarios desarrollen un mejor entendimiento de su problema, este se puede reflejar en un sistema software.

Desventajas:

* El proceso no es visible: los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, no es rentable producir documentos que reflejen cada versión del sistema
* Los sistemas tienen una estructura deficiente: los cambios continuos tienden a corromper la estructura del software, lo cual implica incorporar cambios en el sería una tarea más difícil y costosa.

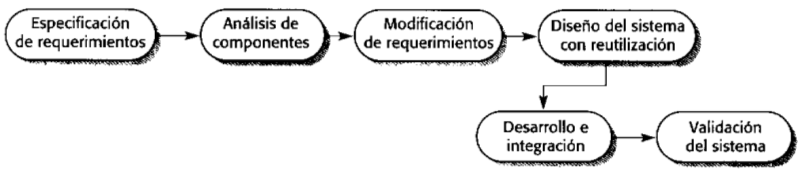
## Ingeniería de Software Basada En Componentes

Se basa en la reutilización de software (proceso informal de conocer diseños o códigos similares a los que se requieren. Se buscan, modifican según lo necesario y son incorporados al sistema), esta técnica es indispensable para el desarrollo rápido de sistemas.

Se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para estos. Estos componentes pueden ser sistemas por si mismos (sistemas comerciales, COTS) que se utilizan para proporcionar funcionalidad específica.

Las etapas de especificación de Requerimientos y la de Validación del sistema son similares a los de otros procesos, las etapas intermedias difieren:

* **Análisis de Componentes**: una vez obtenida la especificación de requerimientos, se buscan los componentes para su implementación. No existe una concordancia exacta y los componentes que se utilizan solo proporcionan parte de la funcionalidad requerida.
* **Modificación de Requerimientos:** los requerimientos se analizan utilizando la información de los componentes. Entonces, los componentes se modifican para reflejar los componentes disponibles. Si no se pueden hacer las modificaciones, se vuelve a la actividad de análisis de componentes para buscar soluciones alternativas.
* **Diseño del Sistema con Reutilización**: se diseña o se reutiliza un marco de trabajo para el sistema. Los diseñadores tienen en cuenta los componentes que se reutilizan y organizan los marcos de trabajo para satisfacerlos. Si los componentes reutilizables no están disponibles, se puede tener que diseñar un nuevo software.
* **Desarrollo e Integración**: para crear el sistema, el software que no se puede adquirir externamente se desarrolla, y los componentes y sistemas se integran. La integración es parte del proceso de desarrollo, más que una actividad separada.



Ventajas:

* Reduce la cantidad de software a desarrollarse, disminuyendo los costos y riesgos.
* Entrega más rápida del software

Desventajas:

* Inevitables compromisos en los requerimientos, dando lugar a un sistema que no cumpla las necesidades reales de los usuarios.
* Si las nuevas versiones de los componentes reutilizables no están bajo el control de la organización que los utiliza, se pierda parte de control sobre la evolución del sistema.

# Iteración de Procesos

En todo tipo de proyectos de software grande, es inevitable que haya cambios en los requerimientos del sistema, debido a las presiones del exterior. Las prioridades de gestión cambian, así como cambian los diseños y la implementación con las tecnologías nuevas. El proceso de software no es único, las actividades se repiten conforme el sistema se rehace en respuesta a cambios.

Existen dos modelos de la iteración de Procesos: **Entrega Incremental y Desarrollo en Espiral.**

## Entrega Incremental

Es un enfoque que combina las ventajas del modelo en cascada y el enfoque de desarrollo evolutivo. En este los clientes identifican los servicios que proporcionara el sistema a grandes rasgos, además de que servicios son más importantes que otros. Entonces se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema, esta asignación de servicios al incremento depende de la prioridad del servicio con aquellos de prioridad más alta.

Cuando se definieron los incrementos del sistema, los requerimientos para los servicios que se van a entregar en el primero se definen en detalle, y se desarrollan. Durante esta etapa de desarrollo, podría llevarse a cabo un análisis adicional de requerimientos para los incrementos posteriores, sin modificar en los requerimientos del incremento actual.

Cuando se completa un incremento, se entrega a los clientes y los ponen en servicio, obteniendo una entrega temprana de parte de la funcionalidad del sistema. Experimentando con el sistema les ayudara a aclarar sus requerimientos para los siguientes incrementos y las últimas versiones del actual. Cuando se completan los nuevos incrementos, se integran a los existentes para que la funcionalidad del sistema sea mejorada con cada incremento entregado.

Ventajas:

* Los clientes no esperan a que el sistema completo se entregue para sacar provecho de él.
* Los clientes usan los primeros incrementos como prototipos y obtienen experiencia sobre los posteriores incrementos del sistema.
* Existe un bajo riesgo de fallo total del proyecto.
* Los servicios de más alta prioridad se entregan primero, y los incrementos se integran a ellos.

Desventajas:

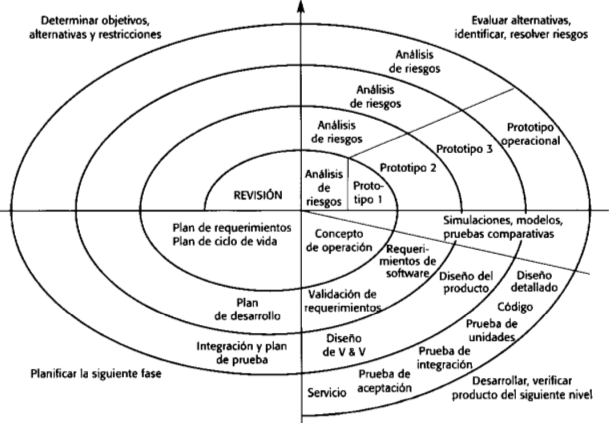
* Los incrementos deben de ser pequeños, cada uno entregando alguna funcionalidad del sistema.
* Puede ser difícil adaptar los requerimientos del cliente a incrementos de tamaño apropiados.
* Difícil identificación de recursos comunes que requieran los incrementos.

## Desarrollo En Espiral

Desarrollado por Boehm en 1988, representa el proceso de software como una secuencia de actividades con retrospectiva de una actividad a otra, representados como una espiral. Cada ciclo en la espiral representa una fase del proceso de software.

Cada ciclo de la espiral se divide en cuatro sectores:

1. **Definición de Objetivos:** se definen os objetivos específicos, se identifican restricciones del proceso y del producto, y se traza un plan detallado de gestión. Se identifican los riesgos del proyecto, los cuales dependiendo de cada uno se planean estrategias alternativas.
2. **Evaluación y Reducción de Riesgos:** se realiza un análisis detallado para cada uno de los riesgos identificados del proyecto. Definir los pasos para reducir dichos riesgos.
3. **Desarrollo y Validación:** luego de evaluar los riesgos, se elige un modelo de desarrollo.
4. **Planificación:** el proyecto se revisa y se toma la decisión de si debe continuar con un ciclo posterior de la espiral, en caso de continuar, se desarrollan los planes para la siguiente fase del proyecto.



Un ciclo de la espiral empieza con la elaboración de objetivos, entonces se enumeran formas alternativas de alcanzarlos y las restricciones impuestas en cada una. Cada alternativa se evalúa contra cada objetivo, identificando las fuentes de riesgo del proyecto. Luego, se resuelven los riesgos encontrados mediante actividades de recopilación de información. Terminada la evaluación de los riesgos, se lleva a cabo cierto desarrollo, seguido de una actividad de planificación para la fase siguiente del proyecto.

Ventajas:

* Este modelo difiere de otros, debido a la consideración explicita del riesgo, los cuales podrían originar problemas en el proyecto (como confección de agendas y excesos en los costos).

# Proceso Unificado

Es un ejemplo de modelo de proceso moderno, proveniente del trabajo de UML y el Proceso Unificado de Desarrollo de Software. Reúne elementos del todos los modelos de procesos genéricos, iteraciones de apoyo e ilustra buenas prácticas en la especificación y el diseño.

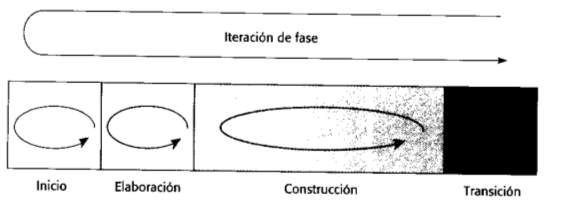
Este se puede describir desde 3 perspectivas:

* Una perspectiva dinámica, mostrando las fases del modelo sobre el tiempo.
* Una perspectiva estática, que muestra las actividades del proceso que se representan.
* Una perspectiva práctica, que sugiere buenas practicas a utilizar durante el proceso.

Es un modelo en fases, que identifican las cuatro fases diferentes en el proceso del software. A diferencia de las fases del modelo en cascada, estas fases se equiparan con las actividades del proceso, y están mucho más relacionadas con asuntos de negocio más que técnicos.

Estas son:

1. **Inicio:** establece un caso de negocio para el sistema. Se identifican todas las entidades externas (personas y sistemas) que interactuaran con el sistema y definir estas interacciones. Esta información se usa para evaluar la aportación que el sistema hace al negocio, si esta aportación es de poca importancia, se puede cancelar el proyecto después de esta fase
2. **Elaboración:** desarrolla una comprensión del dominio del problema, establece un marco de trabajo arquitectónico para el sistema, desarrollar el plan del proyecto e identificar los riesgos claves del proyecto. Terminando esta fase, se debe tener un modelo de los requerimientos del sistema (especificación de casos de uso), una descripción arquitectónica y un plan de desarrollo para el software.
3. **Construcción:** comprende el diseño del sistema, la programación y las pruebas. Aquí se desarrollan e integran las partes del sistema. Terminando con un sistema software operativo y la documentación correspondiente lista para ser entregada a los usuarios.
4. **Transición:** fase final del Proceso Unificado, se ocupa de trasladar el sistema desde la comunidad de desarrollo a la de los usuarios y hacerlo trabajar en un entorno real. Terminando esta fase debe existir un sistema software documentado que funciona correctamente en su entorno operativo.



La iteración dentro del Proceso Unificado se representa de dos formas, cada fase se representa de un modo iterativo con los resultados desarrollados incrementalmente, y el conjunto entero de fases puede también representarse de forma incremental en forma de bucle desde la transición hasta el inicio.

Los flujos de trabajo son las actividades que tienen lugar durante el proceso de desarrollo. Hay seis principales flujos de trabajo del proceso identificados en el proceso:

* **Modelo de Negocios**: se modelan los procesos de negocios con casos de uso de negocio.
* **Requerimientos:** definir los actores que interactúan con el sistema, y se desarrollan casos de uso para modelar los requerimientos del sistema
* **Análisis y Diseño:** se crea y documenta un modelo de diseño utilizando modelos arquitectónicos, modelos de componentes, modelo de objetos y de secuencias
* **Implementación:** se implementan y estructuran en subsistemas los componentes del sistema. La generación automática de código de los modelos de diseño ayuda a acelerar este proceso.
* **Pruebas:** son un proceso iterativo que se llevan a cabo conjuntamente con la implementación. A la finalización de la implementación tienen lugar las pruebas del sistema.
* **Despliegue:** se distribuye a los usuarios y se instala en su lugar de trabajo

Y tres principales flujos de trabajo de soporte:

* Configuración y Cambios de Gestión: gestiona los cambios del sistema
* Gestión de Proyecto: gestiona el desarrollo del sistema.
* Entorno: hace herramientas software apropiadas disponibles para los equipos de desarrollo de software.

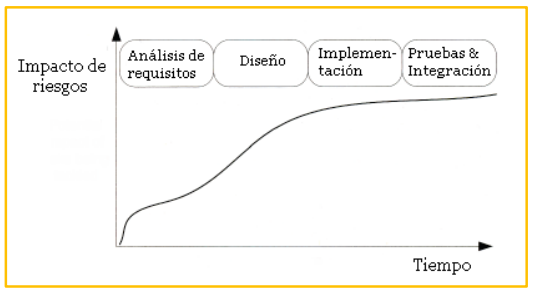
# Metodologías Agiles

Surgen como alternativa a las metodologías tradicionales, las cuales son demasiado burocráticas y rígidas para las características actuales del mercado.

Hoy en día, el entorno en el cual se mueve el software es bastante inestable y cambiante, buscando la reducción del tiempo de creación pero sin dejar de lado la calidad del software, hasta el punto de que las metodologías anteriores no son capaces de adaptarse, mostrando los siguientes inconvenientes:

* Es necesario conocer desde el inicio que es lo que el cliente desea.
* No se deben de cambiar los requerimientos iniciales. A medida que se sigue construyendo el proyecto, modificaciones y correcciones de errores que puedan surgir elevarían sus costos.
* Los mecanismos de control establecidos pueden llegar a dar inflexibilidad a posibles cambios y realizarlo incrementaría su costo.
* Demasiada documentación que podría resultar inútil.
* Lentitud del desarrollo a la hora de la creación de productos.
* Encuentran dificultades casi al final del proyecto, retrasando las entregas





Estos inconveniente hicieron que las metodologías tradicionales no sean capaces de eliminar fallos ni adaptarse a software orientados a Web, estos últimos requieren cambios constantes y tiempos cortos de desarrollo.

Las Metodologías Agiles surgen como una alternativa atractiva a estos problemas. El desarrollo ágil está centrado en:

* **Iteraciones:** realizados en porciones pequeñas de tiempo denominadas ***timeboxes***, desarrolladas por un equipo multidisciplinar auto organizado, los cuales deciden como realizar las tareas en cada iteración.
* **Comunicación** no solo en el ambiente del grupo de trabajo, sino también con alguna figura representativa del cliente. El que represente al cliente es necesario como elemento de apoyo para los desarrolladores, ya que será la persona al cual se podrán realizas las cuestiones necesarias, con el fin de comprobar si se cumplen los objetivos propuestos. Por lo tanto, trabajar con una buena comunicación permite tomar decisiones de forma rápida y aplicarla.

Las metodologías agiles son más adecuadas cuando el entorno presenta cierta incertidumbre o es cambiante, a comparación de las metodologías tradicionales.

Consta de 5 fases que definen el ciclo de desarrollo ágil:

1. **Concepto:** Se define de forma general las características del producto y se asigna el equipo encargado de su desarrollo.
2. **Especulación:** se hacen disposiciones con la información obtenida y se establecen los límites del desarrollo del producto, tales como costes y agendas. Se construirá el producto a partir de las ideas principales y se comprueban las partes realizadas y como impactan en el entorno. Esta fase se repite en cada iteración y consiste en:
   * **Desarrollar y revisar los requisitos generales**
   * **Mantener la lista de las funcionalidades que se esperan.**
   * **Plan de entrega. Se establecen las fechas de las versiones, hitos e iteraciones. Medirá el esfuerzo realizado en el proyecto.**
3. **Exploración:** Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
4. **Revisión:** el equipo revisara todo lo que se ha construido y se contrasta con el objetivo deseado.
5. **Cierre:** Se entregara en las fechas acordadas una versión del producto deseado. Siendo este una versión, el cierre no indica el fin del proyecto, sino que seguirá habiendo cambios denominado “mantenimiento”, acercado al producto final lo más cerca del producto final deseado.



Ventajas:

* Gran capacidad de respuesta ante cambios, los cuales no son reconocidos como problemas sino como algo necesario para que se mejore el producto y pueda satisfacer al cliente. **Los cambios forman parte del proceso de desarrollo.**
* Las entregas no se hacen al final, sino que se hacen pequeñas entregas, las cuales permiten al cliente valorar el producto además de ir trabajando con algunas funcionalidades.
* Los ciclos cortos de entrega ayudan a disminuir el riesgo, sobre todo al principio de cada proyecto.
* Se trabaja en equipo entre el cliente y los desarrolladores mediante una comunicación casi diaria, evitando errores y documentación innecesaria.
* Elimina el trabajo que no es necesario y que no aporta valor al negocio.
* Busca mejores técnicas y mejor diseño para conseguir productos de calidad.
* Mejora los procesos y al equipo de desarrollo.

## SCRUM

Metodología de Gestión de proyectos, en el cual la **agilidad, flexibilidad y la incertidumbre** son los principales elementos.

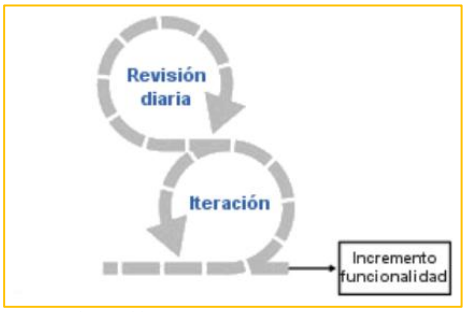
Surge mediante la observación de **Takeuchi y Nonaka** a empresas como Honda, HP, Canon y demás, en donde descubrieron que sus productos no seguían fases de desarrollo con un equipo especializado, sino que partían de unos requisitos muy generales y que un equipo multidisciplinario desarrollaba el producto desde comienzo a fin del proyecto. Compararon esta manera de trabajo en equipo con la colaboración que realizan jugadores de Rugby y la utilización de una formación denominada **Scrum.**

Scrum es adecuado para empresas en las que el desarrollo de productos se realiza en entornos caracterizados por:

1. **Incertidumbre:** se plantea el objetivo que se desea alcanzar sin proporcionar un plan detallado del producto.
2. **Auto organización:** los equipos son capaces de organizarse por sí solos, sin necesidad de roles para la gestión, los cuales deben cumplir las siguientes características:
   * *Autonomía:*son los encargados de encontrar la solución usando la estrategia que encuentren adecuada.
   * *Auto superación:* las soluciones iniciales sufrirán mejoras.
   * *Auto enriquecimiento:* siendo equipos multidisciplinarios se ven enriquecidos de forma mutua, aportando soluciones que puedan complementarse.
3. **Control Moderado:** Se establecerá un control suficiente para evitar descontroles. S basa en crear un escenario de “autocontrol entre iguales” para no impedir la creatividad y espontaneidad de los miembros del equipo.
4. **Transmisión del Conocimiento**: todos aprenden de todos. Las personas pasan de unos proyectos a otros y así comparten sus conocimientos a lo largo de la organización.

Siendo una metodología ágil, tiene como base la idea de creación de ciclos breves para el desarrollo o iteracionesdenominadas ***Sprints*** en Scrum.

Scrum gestiona las iteraciones a través de reuniones diarias, un elemento fundamental de esta ideología.



### Componentes de Scrum

Scrum se puede dividir de forma general en 3 fases, que podemos entender como **reuniones**. Las reuniones forman parte de los artefactos de esta metodología, junto con los **roles** y los elementos que lo forman.

La planificación está compuesta de tres reuniones fundamentales:

1. **Planificación del Backlog:** define un documento en el que se reflejaran los requisitos del sistema por prioridades. Se definen la planificación del **Sprint 0**, donde se deciden cuáles van a ser los objetivos y el trabajo que hay que realizar para la siguiente iteración. De esta reunión se obtendrá además un Sprint Backlog, el cual consistirá en la lista de tareas y establecerá cual será el objetivo más importante del Sprint.
2. **Seguimiento del Sprint:** se hacen reuniones diarias en las que se desarrollaran 3 preguntas importantes para evaluar el avance de las tareas: ***¿Qué trabajo se realizó desde la última reunión?, ¿Qué trabajo se hará hasta la siguiente reunión? y ¿Qué inconvenientes surgieron y que se deberá solucionar para poder avanzar?***
3. **Revisión del Sprint:** en esta fase se finaliza el Sprint y se realizara una revisión del incremento que se ha generado. Se presentaran los resultados finales y una demo o versión, esto ayudara a mejorar el feedback con el cliente.

Los **Roles** se dividen en dos grupos: **cerdos y gallinas,** esto surgio gracias al chiste sobre un cerdo y una gallina que tenían intención de poner un restaurante:



**Los Cerdos** son las personas comprometidas con el proyecto y el proceso de Scrum**:**

* **Product Owner:** persona que toma las decisiones, es la que realmente conoce el negocio del cliente y su visión del producto. Se encarga de escribir las ideas del cliente, las ordena por prioridad y las coloca en el Product Backlog.
* **Scrum Master:** encargado de comprobar que el modelo y la metodología funciona. Elimina todos los inconvenientes que hagan que el proceso no fluya e interactúa con el cliente y los gestores.
* **Equipo de Desarrollo:** suele ser un equipo de unas 5 o 9 personas y tienen la autoridad para organizar y tomar las decisiones para conseguir su objetivo. Está involucrado en la estimación del esfuerzo de las tareas del Backlog.

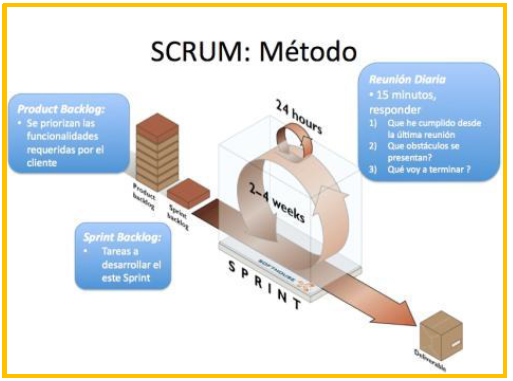
**Las Gallinas** son aquellos que aunque no son parte del proceso de Scrum, es necesario que parte de la retroalimentación de la salida del proceso y así poder revisar y planear cada sprint.

* **Usuarios:** destinatario final del producto.
* **Stakeholders:** personas a las que el proyecto producirá beneficio. Participan durante las revisiones de Sprint.
* **Managers:** Toma las decisiones finales participando en la selección de los objetivos y de los requisitos.

### Elementos de Scrum

Los elementos que forman a Scrum son:

* **Product Backlog:** representa el inventario en el que se almacenan todas las funcionalidades o requisitos en forma de lista priorizada. Estos requisitos serán los que tendrá el producto o los que irán adquiriendo en sucesivas iteraciones. Esta lista es gestionada y creada por el cliente con la ayuda del Scrum Master, quien indicara el coste estimado para completar un requisito, y además contendrá todo lo que aporte un valor al producto.
* **Sprint Backlog:** lista de las tareas que elaborara el equipo durante la planificación de un Sprint. Se asignan tareas a cada persona y el tiempo que queda para terminarlas. De esta manera el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en que tareas no se está avanzando e intentar eliminar el problema.
* **Incremento:** representa los requisitos que se han completado en una iteración y que son perfectamente operativos. Según los resultados que se obtengan, el cliente puede ir haciendo los cambios necesarios y replantear el proyecto.



# Cuadro Comparativo de Modelos de Procesos de Software

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Comprensión de Requerimientos (Cliente) | Comprensión del Dominio (Desarrolladores) | Experiencia en el uso de la Metodología | Manejo de Contingencias en el Desarrollo | Adaptabilidad al cambio de requerimientos | Tamaño del Proyecto | Interacción con el cliente |
| Modelo en Cascada | Alta, el cliente debe tener en claro lo que necesita desde un principio. | Alta, se debe conocer bien el dominio del problema. | Alta, este modelo no es conveniente cuando se es el primer trabajo para los desarrolladores. Por lo tanto, el equipo debe tener experiencia en el uso de este modelo. | Baja, si no se determinan bien los requerimientos desde el principio, aparecerán cuando el proceso finalice, lo cual no es conveniente. | Baja, se debe conocer todos los requerimientos al principio. Algún cambio a mitad del proceso implicaría volver a realizarlo. | Media, si el proyecto es ambicioso requerirá más tiempo, en cambio en proyectos chicos y bien detallados el tiempo es menor. | Baja, el cliente interactúa al principio (indicando sus necesidades) y al final cuando se obtiene un producto terminado |
| Desarrollo Evolutivo | Media, el cliente es participe en todo el proceso de desarrollo. Por lo tanto algún requerimiento adicional puede aparecer durante el proceso. | Media, al tener al cliente brindando información no es necesario conocimientos previos sobre el ámbito. | Media, los desarrolladores deben conocer hasta qué punto se pueden realizar los cambios sin corromper al sistema. | Media, al ser un proceso repetitivo la presencia de alguna contingencia se solucionara mediante la siguiente iteración. Pero demasiadas iteraciones pueden ser costosas. | Media, los nuevos requerimientos que brinde el cliente serán abordados mientras no comprometa al desarrollo. | Media, al ser adaptable puede llevarse adelante en proyectos grandes o chicos. | Medio, el cliente es participe del proceso, pero es posible que no se desarrolle todo lo que pide. |
| Basada en Componentes | Baja, el cliente no tiene interacción con el sistema, así que brinda poca información sobre sus requerimientos. | Baja, al conocer el dominio del problema los desarrolladores buscaran soluciones existentes y las integraran en base a los requerimientos. | Medio, si bien se tienen una gran cantidad de componentes o sistemas similares a los requerimientos, se deberá tener conocimiento para saber cuál utilizar y saber integrarlos. | Baja, si se obtiene algún problema es probable que sea difícil saber en qué componentes ocurre. | Baja | Media, quizás ideal para pequeños y medianos proyectos. | Baja |
| Entrega Incremental | Alta, el cliente no solo interviene en brindar los requisitos, sino en la devolución de cada entrega para los siguientes incrementos. | Baja, el equipo desarrollara lo que entendieron y se retroalimentaran con las respuestas de los clientes en cada entrega parcial. | Baja, no se requiere mucha experiencia por parte del equipo de desarrollo, pueden ir aprendiendo en cada entrega. | Alta, las contingencias se manejan más temprano e incluso el cliente informa sobre estas. | Alta, el cliente brindara siempre requerimientos nuevos para cada incremento. | Medio, ideal para proyectos medianos y grandes. | Alta, el cliente está constantemente brindando su devolución en cada entrega para que el software se acercarse más a lo que este necesite. |
| Desarrollo en Espiral | Alta, el cliente debe brindar detalladamente los requerimientos para el análisis de algún riesgo. | Media | Alta, el grupo de desarrollo debe estar capacitado para poder reconocer los riesgos a través de los requerimientos en una etapa temprana en el desarrollo y desarrollar tácticas para enfrentarlo | Alta, se establece de manera temprana una estimación de los riesgos y planificar una estrategia en cuanto esto ocurran. | Baja, se centra más en la prevención de riesgos, para esto se debe tener un conocimiento completo de los conocimientos. | Medio, para proyectos medianos o cortos. | Baja. |
| Proceso Unificado | Media, al ser un proceso iterativo e incremental, el total de los requerimientos no surgirán al principio. | Media, el equipo de desarrollo se centra más en el dominio que en el desarrollo. | Media, los desarrolladores pueden o no tener experiencia en el proceso. | Medio | Alta, a medida que el cliente informa sobre sus requerimientos, irán existiendo los cambios al proyecto. | Apto para todos los proyectos. | Alta, los clientes están constantemente en colaboración con el equipo de desarrollo. |
| Metodologías Agiles: Scrum | Baja, el cliente continuara informando sobre los requerimientos incluso cuando ya se ha empezado el proceso. | Media, a medida que el cliente interactúa con el equipo estos aprenden con este y no requieren conocimientos previos necesarios para abordar el proyecto | Baja, los desarrolladores que no estén experimentados con el proceso pueden aprender de la mano con sus compañeros. | Alta, al ser un proceso iterativo, cualquier contingencia detectada se la corregirá en la siguiente iteración. | Alta, debido a que el cliente está constantemente brindando retroalimentación, así que los cambios son inevitables. | Media, adaptable para distintos tipos de proyectos. | Alta, el cliente y aquellos implicados están constantemente retroalimentando al equipo para llegar a obtener un producto cercano a sus ideales. |

# Conclusiones

A medida que el software fue avanzando en los años, la constante demanda a adaptabilidad a cambios hizo que algunos modelos nacieran, sean reformulados o incluso reemplacen a otros. Sin embargo, determinados ámbitos que solicitan algún software fácilmente puede ser desarrollado por alguno de los modelos más antiguos antes mencionados. Así que el uso de determinados procesos son eficaces en determinadas circunstancias, será el equipo de desarrollo el encargado de elegir el adecuado y llevarlo a cabo, con el fin de brindar un producto lo más cercano a lo deseado por los clientes.

# Bibliografía

Ingenieria de Software – Ian Summerville

Metodologia Scrum – Gestion de proyectos informáticos – Manuel Trigas Gallego

<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf>