# ÁgilUC

## Proceso de desarrollo de software para pequeñas organizaciones

Elaborado por:

Sandra Victoria Hurtado Gil

Óscar Hernán Franco Bedoya

# Tabla de Contenido

# Proceso ÁgilUC

El proceso ÁgilUC comprende un conjunto de actividades, productos, roles y guías para orientar el desarrollo de software para equipos de desarrollo pequeños. El proceso permite cumplir con los lineamientos para el ciclo de vida del desarrollo de software definidos en la norma ISO/IEC 29110, parte 5-1-; al tiempo que incluye prácticas ampliamente conocidas de métodos ágiles.

**Resumen del proceso:**

La primera actividad es la identificación de un problema u oportunidad que conlleva el desarrollo de un software. A partir de esta problemática u oportunidad se contempla una reunión inicial entre todos los interesados para establecer de manera colaborativa la visión del proyecto, incluyendo un plan preliminar. Posteriormente, se procede con dos actividades en paralelo: definir la arquitectura del software y a configurar el entorno, es decir, establecer toda la infraestructura necesaria para el proyecto.

Seguidamente se procede con el desarrollo propiamente dicho, el cual se realiza mediante iteraciones. Cada iteración es un subproceso, pues está conformadas por otras actividades, tanto de ingeniería como de gestión, que incluyen la planeación y seguimiento y todo lo relacionado con requisitos, modelado (es decir, análisis y diseño), construcción, integración y pruebas. Además, considerando que se trata de un proceso ágil, al final de cada iteración hay una revisión y una retrospectiva.

Al finalizar el proyecto se hace una entrega formal al cliente, lo que constituye la última actividad del proceso.

## Roles

En ÁgilUC se proponen tres roles, que son:

- Representante del cliente: Corresponde a una persona que representa a los clientes y usuarios de la solución de software que se desarrollará. Quien tenga este rol debe conocer el negocio, organización o contexto donde se implementará el software,

para poder definir los requisitos. También debe poder aprobar las entregas y los cambios realizados.

*En caso de que el software corresponda a una idea de negocio del equipo de desarrollo, es importante que contacten algún experto en el dominio para que haga las veces de cliente, por lo menos en algunas de las actividades.*

- Gestor del proyecto: Corresponde a una persona que lidera la parte administrativa del proyecto de desarrollo del software, por lo cual debe tener buenas habilidades de comunicación, para poder entender los requisitos y necesidades tanto de los clientes y usuarios como del equipo de desarrollo. También debe realizar seguimiento al desarrollo del proyecto, para identificar posibles desviaciones de los planes y poder tomar, junto con los demás interesados, acciones correctivas.

  *Se debe seleccionar a un integrante del equipo para que tenga este rol. La selección la debe realizar el mismo equipo, y en caso de no llegar a un consenso, se puede rotar entre los integrantes en cada iteración del proyecto. El gestor del proyecto también debe tener el rol de miembro del equipo de desarrollo.*
- Equipo de desarrollo: Grupo de personas encargados de desarrollar el software. Quienes tengan este rol deben estimar la duración y los recursos necesarios para desarrollar cada uno de los incrementos del software y conocer y aplicar prácticas, técnicas y métodos de ingeniería de software, especialmente los relacionados con requisitos, análisis, diseño, implementación, pruebas, implantación, calidad y gestión de la configuración. También deben recolectar y analizar la información que se requiera para cada proyecto.

Independiente del rol, todos los que participan en un desarrollo con ÁgilUC deben:

- Colaborar para lograr el éxito del proyecto.
- Estar comprometidos con el desarrollo de un software de alta calidad.
- Proponer cambios en el software y en el proceso cuando estos contribuyan al logro de los objetivos estratégicos del cliente.
- Informar a tiempo de cualquier inconveniente, para que se puedan tomar acciones correctivas de forma temprana.

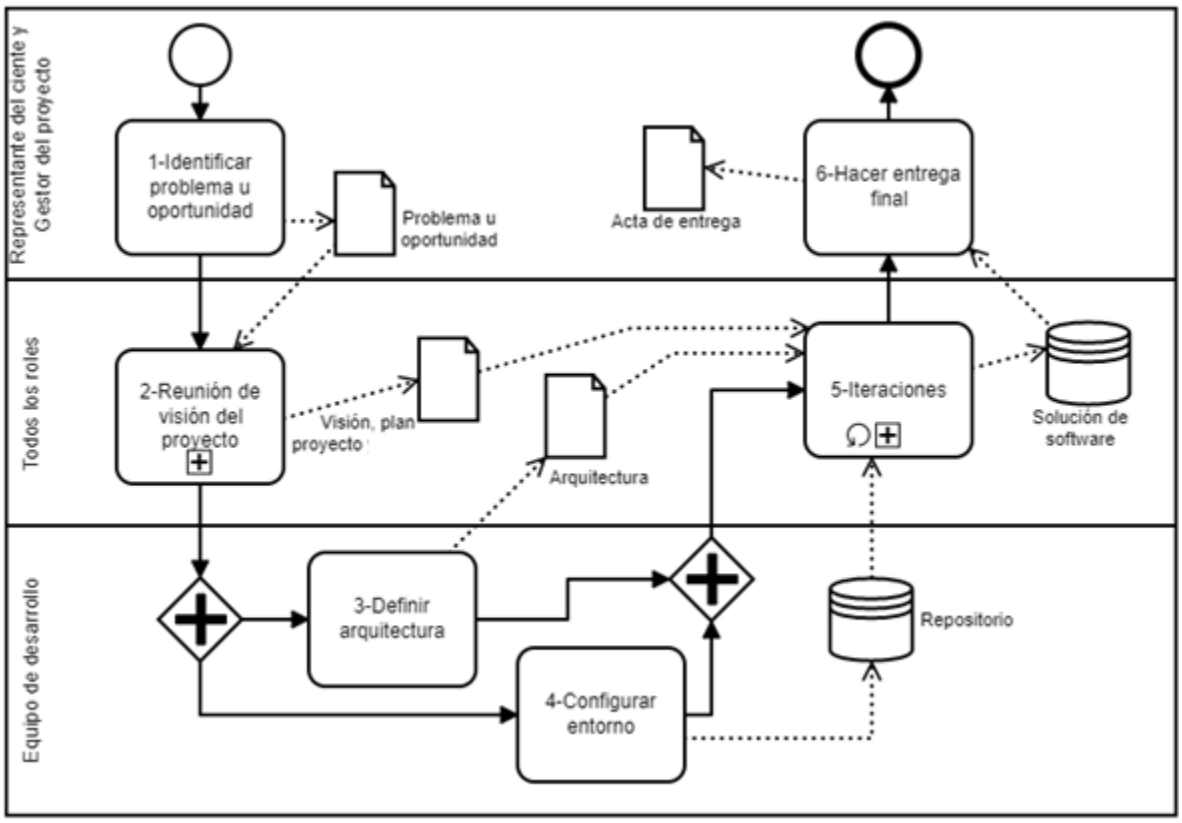
- Ser propositivos y estar dispuestos a cambiar sus hábitos de trabajo para aplicar las diferentes propuestas de mejora que se realicen en cada iteración.

## Productos

En ÁgilUC los productos corresponden a las entradas y salidas de las diferentes actividades. Pueden ser documentos, modelos, código, etc., y todos corresponden a elementos importantes para el desarrollo de la Solución de Software para el cliente. Para los documentos que tengan formatos, estos se encontrarán en una carpeta compartida.

## Actividades detalladas

Las actividades generales de ÁgilUC, esta vez mostrando los roles y los productos asociados, son:



## Actividad 1: Identificar problema u oportunidad

**Entradas**:

Ninguna.

**Salidas**:

Descripción del problema u oportunidad.

**Roles participantes**:

Representante del cliente y gestor del proyecto. Pueden participar los dos roles, o también puede ser solo uno de los dos (cualquiera).

**Descripción**:

El representante del cliente o el gestor del proyecto identifican un problema que puede ser resuelto mediante un software, el cual puede ser para un área de una empresa u organización, para una comunidad o para una persona en particular.

También puede ser que se identifique una oportunidad de negocio a partir del desarrollo de un software. En este caso es posible que el software no sea para un cliente en particular, sino que sea para un mercado más amplio.

En ambos casos es importante establecer la idea o problema en términos generales, junto con elementos que permitan delimitar el alcance del proyecto.

## Actividad 2: Reunión de visión del proyecto

**Entradas**:

Descripción del problema u oportunidad.

**Salidas**:

- Documento de visión.
- Plan del proyecto.

**Roles participantes**:

Todos los roles.

**Descripción**:

Esta reunión se debe realizar antes de comenzar formalmente el proyecto, puesto que aquí se definen los requisitos de más alto nivel (requisitos del negocio), que permiten establecer, entre otros, el alcance y las restricciones, e incluso un plan preliminar para el proyecto.

**Tareas**:

Determinar los objetivos:

1. El representante del cliente presenta los beneficios que traería un sistema de información, las metas que se esperan lograr con el mismo.
2. Todos los participantes realizan preguntas que permitan identificar los objetivos y/o el valor agregado del producto. Ejemplo de preguntas: **¿para qué? ¿por qué?** ¿qué pasa si no se cuenta con el software?

   Por lo general los objetivos están relacionados con los problemas u oportunidades. Por ejemplo, si un problema es que tomaba mucho tiempo realizar algún proceso de negocio, o que presentaba muchos errores, el objetivo puede estar relacionado con disminuir este tiempo y/o los errores.

   Se deben identificar pocos objetivos (idealmente solo uno). Se recomienda que los objetivos sean medibles, para que sea más fácil la evaluación de su cumplimiento. Por ejemplo, el objetivo: "Incrementar las veces que compra un mismo cliente", puede refinarse, estableciendo tiempos y porcentaje de incremento: "Durante el primer año de puesta en marcha del sistema se espera incrementar en un 30 % las veces que compra un mismo cliente".

Identificar restricciones:

3. El representante del cliente presenta las restricciones o limitantes que ha identificado para el desarrollo del software. Por ejemplo, restricciones de tiempo o de presupuesto. También pueden ser restricciones de tipo tecnológico, por ejemplo, los servidores o tecnologías con que cuenta la empresa.
4. El gestor del proyecto y el equipo de desarrollo presentan las restricciones que tiene, por ejemplo, de tiempo o de conocimiento de tecnologías por parte de sus integrantes.
5. Todos los participantes realizan preguntas para identificar posibles restricciones que no se hayan tenido en cuenta. Se recomienda realizar una revisión de aspectos LEGALES. Esto muchas veces no es tenido en cuenta por equipos pequeños de desarrollo y puede afectar en gran medida el software. Considerar, por ejemplo, reglamentaciones que existan para facturación, impuestos, privacidad de la información, formatos para los

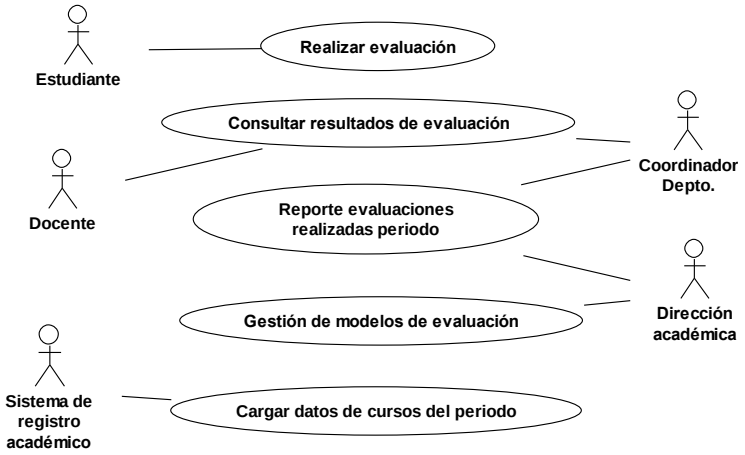
datos, tipos de aplicaciones permitidas, etc. Estas reglamentaciones pueden ser de orden local, pero también nacional e incluso internacional.

Identificar *stakeholders*:

6. De manera colaborativa se identifican las entidades externas -roles u otros sistemas- que van a interactuar **directamente** con el nuevo sistema, ingresando u obteniendo algún dato o información.
7. Se adicionan los diferentes roles del proyecto a la lista de *stakeholders*: representante del cliente, gestor del proyecto y equipo de desarrollo. Estos son muy importantes, pues están encargados del desarrollo de la solución.
8. Se procede a realizar una serie de preguntas para complementar la lista de *stakeholders*, considerando personas (roles) o entidades que NO usan directamente el sistema, pero indirectamente tienen alguna relación. Entre las preguntas que se pueden hacer están: ¿Quiénes resultan beneficiados por este sistema? ¿Quiénes pueden resultar afectados negativamente por el desarrollo de este sistema? ¿Dónde se alojará el sistema (por ejemplo, con un proveedor en la nube, que por lo tanto sería un *stakeholder*)? ¿Quién le realizará mantenimiento? ¿Cuáles entes definen reglamentaciones o leyes que puedan tener relación con el sistema? ¿Quiénes realizarán el mercadeo y venta del sistema (si es una idea de negocio)?

Elaborar diagrama de casos de uso:

9. Se elabora, de forma colaborativa, un diagrama de casos de uso del sistema.



- Se revisa que los casos de uso identificados permitan cumplir con los objetivos definidos. Este paso es muy importante. Se debe verificar que no aparecen casos de uso innecesarios, y que lo casos de uso planteados sí contribuyen a dar valor al negocio.
- Se hacen preguntas para identificar casos de uso que no son tan evidentes. Algunas preguntas que se pueden hacer son: ¿alguien necesita recibir alguna información no periódica que se obtenga del sistema? ¿se debe informar al sistema de alguna situación externa poco común? ¿se realizarán procesos de sincronización - exportar o importar información? ¿se necesita hacer gestión de datos básicos, que facilite el ingreso o consulta de datos en otros casos de uso? ¿se pueden obtener reportes o visualizadores de información consolidada que ayuden en la toma de decisiones?

Elaborar lista de ítems de desarrollo:

10. Se identifican los **ítems de desarrollo** de la siguiente forma:
    - Para cada caso de uso se analiza si es pequeño y puede tomarse como un solo ítem de desarrollo (por ejemplo: “consultar un cliente por cédula”).
    - Si el caso de uso puede subdividirse, por ejemplo, porque tiene flujos alternos importantes o elementos complejos, se anota cada parte como un ítem de desarrollo diferente (por ejemplo, si se tiene “consultar horarios de asesoría de un profesor”, se puede dividir: “consultar todos los horarios del semestre”, “filtrar consulta por materias” y “filtrar consultar por rango de fechas”).

      *Es importante considerar el esquema de trabajo del equipo, pues esto puede llevar a generar ítems de desarrollo diferentes para un caso de uso. Por ejemplo: “backend de consultar cliente por céula” y “frontend de consultar cliente por cédula”.*
11. Se registran en el plan del proyecto los ítems de desarrollo definidos.

Estimar tiempos:

12. El equipo de desarrollo establece el tamaño estimado de los ítems de desarrollo que.

    Para esto se usa la estimación basada en tallas de camiseta, siendo “S” el más pequeño, “M” aproximadamente dos veces y media uno pequeño, “L” más o menos cinco veces el tamaño de uno pequeño y “XL” más o menos diez veces el tamaño de uno pequeño.

Si se encuentran ítems de tamaño más grande, se deben dividir. Este tamaño se registra en el plan del proyecto.

13. El equipo establece los días que son necesarios para desarrollar completamente un ítem pequeño (“talla S”), considerando todas las actividades necesarias para que se tenga un resultado de alta calidad (requisitos, modelado, revisiones, construcción, pruebas, etc.). Este tiempo dará la pauta para los tiempos de los demás tamaños. Este tiempo se registra en la sección “valores” del documento del plan del proyecto.
14. Al definir los tamaños de cada ítem se puede calcular el total de días estimados para el proyecto. A estos días, que corresponden a las actividades técnicas, se le debe sumar los días de las actividades de gestión (1.5 días por iteración, si es de dos semanas, o 2.5 días por iteración, si es de tres semanas). Con estos datos se calcula el número de iteraciones estimadas, considerando que una iteración de dos semanas son diez días, y una de tres semanas son quince días.

    Este cálculo se hace automáticamente en el documento de plan del proyecto.
15. Si el representante del cliente no está de acuerdo con el tiempo que resulta de la estimación, se pueden cambiar los ítems para que el proyecto tenga un alcance menor. NO se debe disminuir el tiempo estimado para el desarrollo de los ítems, pues esto puede afectar la calidad.

## Actividad 3: Definir arquitectura

**Entradas**:

- Documento de visión.
- Plan del proyecto.

**Salidas**:

Documento de arquitectura.

**Roles participantes**:

Equipo de desarrollo.

**Descripción**:

Para describir la arquitectura se usa un conjunto de vistas, que permiten considerar diferentes aspectos del sistema, así: una vista o visión general de la infraestructura para

identificar los elementos de hardware y software involucrados, una vista lógica para establecer cómo se deben distribuir las responsabilidades los componentes de la aplicación, una vista dinámica para entender cómo se comunicarían los componentes y opcionalmente una vista conceptual para identificar la información que será persistente.
Después de elaborar la arquitectura debe ser revisada y aprobada por todo el equipo de desarrollo.

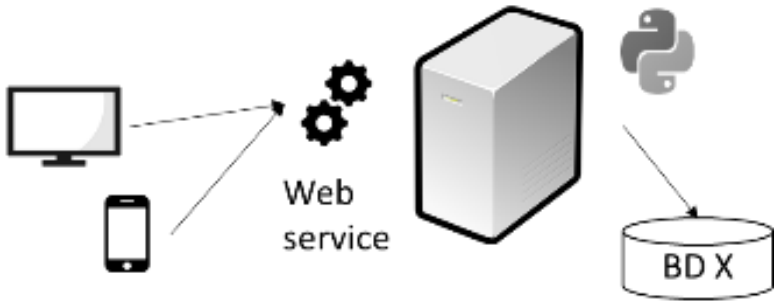
**Tareas**:

Determinar los atributos de calidad:

1. Los participantes analizan los atributos de calidad que son más importantes para el producto, basados en el modelo ISO/IEC 25010 (http://iso25000.com/index.php/normas-iso-25000/iso-25010). Se seleccionan las subcaracterísticas presentadas en el modelo que son más prioritarias para el software que se va a desarrollar. Es importante revisar que los atributos (subcaracterísticas) no entran en conflicto y son realmente justificados (pues cada uno tendrá un costo en el desarrollo). Se pueden hacer preguntas como: ¿qué pasa si no se cuenta con …?, ¿invertiría más con tal de tener …? Por ejemplo: ¿qué pasa si no es modular? ¿invertiría un poco más para garantizar mejor comportamiento temporal (tiempo de respuesta)?
2. Se refinan los atributos de calidad identificados para que incluyan elementos cuantitativos o alguna forma de medición, y que no queden genéricos.
   Por ejemplo, si se seleccionó como atributo la capacidad (desempeño), se puede concretar como: “El sistema debe poder soportar al tiempo entre 150 y 300 usuarios simultáneos ingresando evaluaciones docentes”.

Elaborar la visión general de la infraestructura:

3. Tomando como base el entorno de desarrollo definido (lo cual se hace en una actividad paralela), se elabora un diagrama libre que represente los componentes tecnológicos involucrados -de hardware y software- y sus relaciones. Por ejemplo:



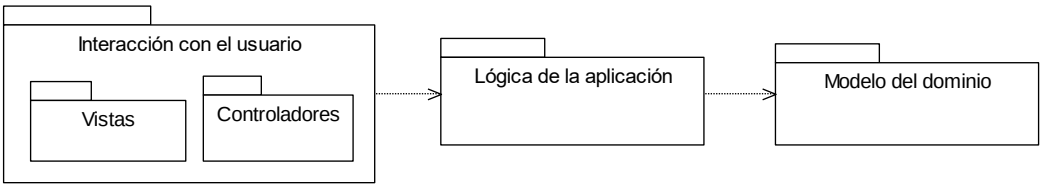

Es importante complementar el diagrama con una breve descripción para poder interpretarlo adecuadamente.

4. (Opcional) Se identifican los elementos de hardware y software que sean externos al sistema, y que necesiten relacionarse de alguna manera, ya sea porque el sistema obtiene datos de estos elementos, o porque necesita enviarles alguna notificación o datos. Se pueden considerar sensores, servicios de proveedores externos (por ejemplo, un banco), etc. Para cada uno de estos elementos se describe la interfaz que se necesitará para la comunicación.

Elaborar la vista lógica:

5. Se determina el tipo de software que se desarrollará (por ejemplo, un sistema distribuido, una aplicación web, una aplicación móvil, etc.), y se selecciona un estilo o estilos arquitectónicos que apliquen para ese tipo. Por ejemplo, un estilo por capas puede ser usado para aplicaciones web.

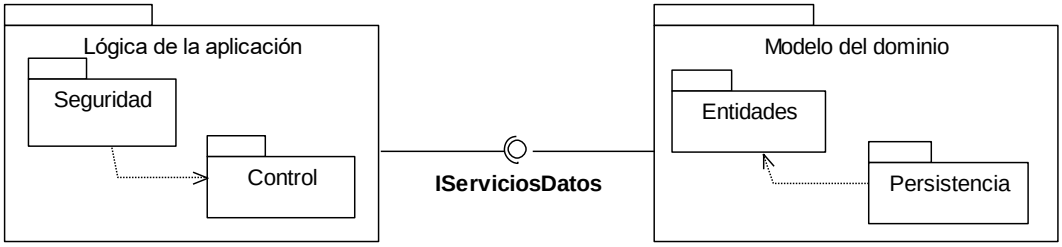
   El diagrama correspondiente a la vista lógica puede ser un diagrama de paquetes UML. Por ejemplo:




6. Se analizan los casos de uso y se determina si son necesarios sub-paquetes –entendidos como módulos o componentes- que se necesitarían para su desarrollo. Por ejemplo: un componente de seguridad, uno de lógica del negocio, otro de persistencia, etc. Se

incluyen estos sub-paquetes en el diagrama, considerando que deben estar acordes con el estilo arquitectónico seleccionado previamente.

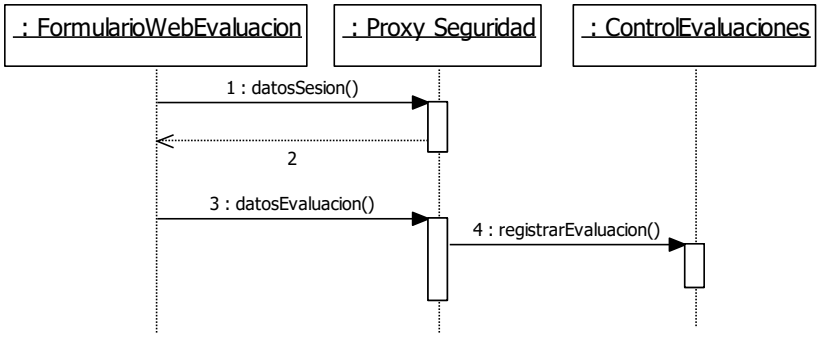
También se establecen relaciones entre los componentes, considerando: si están en el mismo paquete puede ser una relación directa, pero si están en diferentes paquetes se deberían comunicar mediante una interfaz. Por ejemplo:




Se debe incluir una explicación para definir la responsabilidad de cada paquete, las relaciones entre ellos y cualquier información adicional importante. Por ejemplo, se puede aclarar que para la persistencia se usará el *framework* Hibernate.

Elaborar la vista dinámica:

7. Teniendo como base el caso de uso más complejo se elabora un diagrama de secuencia que muestre cómo sería el paso de mensajes entre los componentes para lograr la funcionalidad de este caso de uso. Por ejemplo:



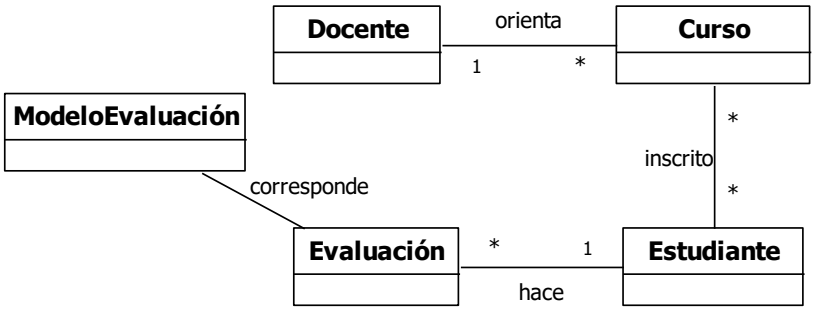

Este diagrama, además de ser una vista arquitectónica que sirve como referente para el desarrollo posterior, cumple con una finalidad de revisión:

Al elaborarlo se verifica que es clara la responsabilidad de cada componente y los servicios que puede ofrecer. También se verifica que las relaciones definidas entre los componentes permiten lograr el objetivo. Si se encuentran dificultades para saber cuál

componente ofrece cada servicio y cómo se comunica con los otros, entonces puede ser necesario hacer cambios en la estructura definida. También es posible identificar si un componente tiene demasiadas responsabilidades, en cuyo caso se puede dividir.

Elaborar la vista conceptual:

8. Se identifican los principales conceptos o entidades del negocio que deberán ser manejados en el software, junto con las relaciones que existen entre ellos. Cada concepto y relación tienen un nombre significativo que corresponde a los términos que se utilizan en el negocio. Este diagrama contribuye a mejorar el entendimiento del negocio y también permite identificar la información que sería persistente. Por ejemplo:




Revisar la arquitectura:

9. Se realiza una revisión por parte de todo el equipo de desarrollo. En esta reunión todos tienen el papel de aprendices y de evaluadores, es decir, no solo deben entender la arquitectura que se está presentando, sino que la deben evaluar desde su experiencia para saber si es adecuada para el proyecto. Para esto, los asistentes deben realizar preguntas y aportes propositivos, y de esta manera se pueden hacer ajustes en la arquitectura si es del caso.

## Actividad 4: Configurar entorno

**Entradas**:

- Documento de visión.
- Plan del proyecto.

**Salidas**:

Repositorios del proyecto.

**Roles participantes**:

Equipo de desarrollo.

**Descripción**:

Se crean el repositorio o repositorios del proyecto, donde se guarda toda la documentación pertinente y también el código, éste último controlado mediante un sistema de versiones. Al mismo tiempo se definen e instalan las herramientas necesarias para el desarrollo. Complementado lo anterior, el equipo establece un conjunto de lineamientos y estándares básicos para el proyecto.

**Tareas**:

Crear repositorio de documentos del proyecto:

1. Se crea un repositorio de documentos para el proyecto, usando alguna herramienta que permita el trabajo colaborativo. Puede usarse, por ejemplo, un drive compartido, un gestor de contenidos, una aplicación especializada en este tipo de repositorios o el servidor de versiones (si permite incluir documentación).
   El repositorio debe tener un nombre relacionado con el proyecto, e internamente se recomienda organizar los documentos en carpetas.
   Se verifica que todos los miembros del equipo tengan acceso al repositorio creado.

Definir e instalar herramientas para el desarrollo:

2. El equipo determina las herramientas que se utilizarán para el desarrollo, considerando el tipo de software que se desea, la experiencia del equipo y las tendencias del mercado. También se deben tener en cuenta las restricciones que se tengan (las cuales están en el documento de visión).
   Es importante definir no solo las herramientas, sino también **la versión** que se usará de cada una.
3. La lista de herramientas, con la versión y el enlace donde se encuentra se incluyen en el documento de arquitectura. El enlace puede corresponder al sitio web oficial de la herramienta.
4. Todos los miembros del equipo instalan las herramientas seleccionadas y verifican su funcionalidad básica.

Crear repositorio de código del proyecto:

5. Se crea el repositorio para el código en el servidor de manejo de versiones seleccionado, con la estructura básica del proyecto, de acuerdo con la arquitectura definida.
6. Cada integrante accede al repositorio y se encarga de copiar lo necesario, para quedar con su área de trabajo local configurada.

Definir estándares y lineamientos de trabajo:

7. El equipo se reúne, y con la orientación del gestor del proyecto, definen los mecanismos de comunicación que usará el equipo. Todos deben estar de acuerdo y comprometerse a usar estos mecanismos para facilitar el desarrollo. Se recomienda:
   - Definir periodicidad, hora y lugar de las reuniones de seguimiento.
   - Definir la forma de comunicar las noticias y los inconvenientes que se presenten (complementario a las reuniones de seguimiento).
   - Definir la forma en la cual se plantearán o se resolverán inquietudes por parte del representante del cliente.
8. Se definen los estándares que se usarán para el proyecto. Como mínimo se deben definir estándares de codificación y estándares de nombre de documentos.
   Los estándares y lineamientos definidos se publican en el repositorio de documentos.

## Actividad 5 (Subproceso): Iteraciones

**Entradas**:

- Documento de visión.
- Plan del proyecto.
- Documento de arquitectura.
- Repositorios del proyecto.

**Salidas**:

Solución de software.

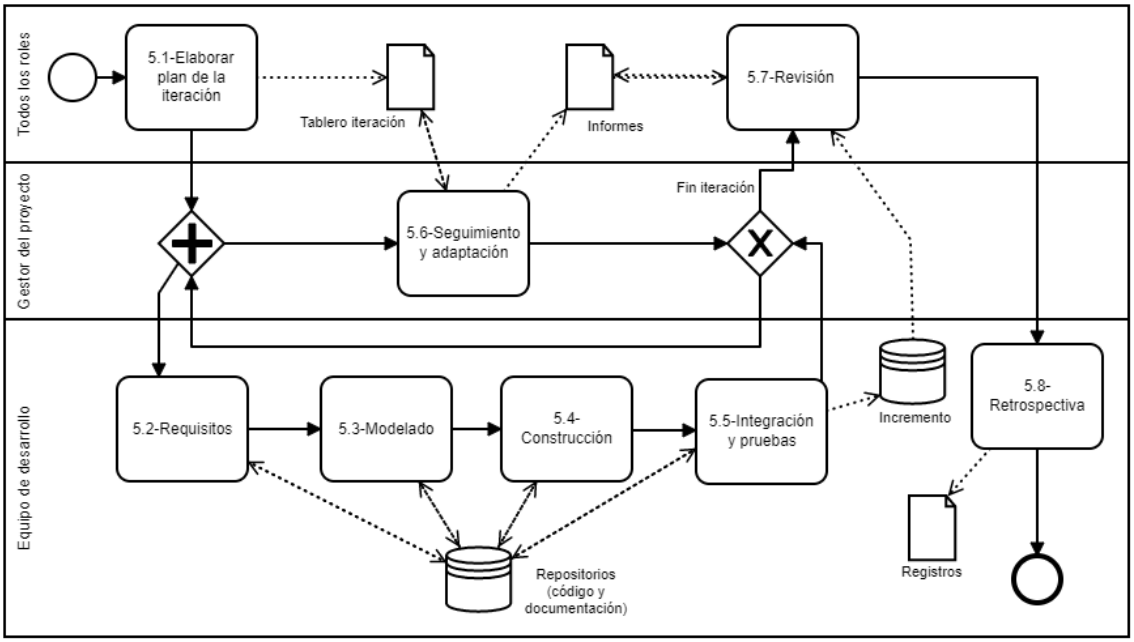
**Roles participantes**:

Todos los roles.

**Descripción**:

El desarrollo del proyecto se lleva a cabo mediante iteraciones de dos o tres semanas, donde se realizan actividades de ingeniería y actividades de gestión. Cada iteración comienza con un plan, donde se definen las tareas concretas que realizará el equipo de desarrollo para producir un incremento de software. Durante la iteración se ejecuta el plan y se realiza seguimiento y control frecuente a éste. Al finalizar cada iteración el incremento resultante es revisado y aprobado por el cliente, para que pueda hacer parte de la solución.

**Diagrama del subproceso:**



**Actividad 5.1: Elaborar plan de la iteración**

**Entradas**:

- Documento de visión.
- Plan del proyecto.

**Salidas**:

- Tablero de la iteración.

**Roles participantes**:

Todos los roles (aunque el principal responsable es el equipo de desarrollo).

**Tareas**:

Refinar y priorizar ítems de desarrollo:

1. El representante del cliente revisa los ítems de desarrollo, para determinar si es necesario realizar ajustes, y los ordena por prioridad.
   El equipo de desarrollo y el gestor del proyecto puede realizar preguntas al representante del cliente para entender mejor la prioridad que se está definiendo, y también pueden hacer propuestas si lo consideran pertinente.

Seleccionar ítems para la iteración:

2. El equipo de desarrollo empieza a seleccionar los ítems de desarrollo que alcanzará a realizar en la iteración, de la siguiente forma:
   - Se analiza el primer ítem del listado, pues se deben tomar siempre en el orden dado por el representante del cliente. Si el tiempo estimado alcanza para ser desarrollado en la iteración, se selecciona el ítem.
   - Se continúa con los siguientes ítems (en orden) de la misma forma que con el primero, pero teniendo en cuenta que el tiempo de este ítem suma a los que ya estaban seleccionados. También se debe tener en cuenta dejar un margen para las actividades de gestión (1.5 días para iteraciones de dos semanas o 2.5 para iteraciones de tres semanas).

- Si un ítem es demasiado grande para la iteración, se analiza si es posible dividirlo, convirtiendo ese ítem en dos o tres -y teniendo en cuenta que cada ítem debe agregar algo de valor por sí solo.

3. Se coloca un identificador a cada ítem de desarrollo seleccionado. Puede ser algo sencillo, como un número secuencial, o puede incluir alguna convención definida por el equipo. Por ejemplo: ID-01.

Establecer tareas para el desarrollo:

4. Se crea un tablero para la iteración, con las siguientes columnas: Ítems seleccionados, Tareas pendientes, Tareas en ejecución y Tareas terminadas (este tablero se puede hacer en Excel o en alguna herramienta como Trello).
5. **Para cada ítem** de desarrollo seleccionado para la iteración:
   - El equipo determina las tareas concretas que se deben realizar. Para saber cuáles tareas se necesitan, el equipo debe conocer los pasos que propone ÁgilUC, pero también debe tener como referentes los entregables acordados (como manuales) y su propia experiencia. Se deben considerar tareas de requisitos, diseño, codificación, integración y pruebas.

     Es importante que las tareas no se dejen demasiado generales (como "hacer el diseño"), sino que sean más detalladas; por ejemplo: "elaborar un diagrama de clases para el *backend* de registrar producto".
   - Cada tarea se anota en la columna "Tareas pendientes", y se coloca su tiempo estimado, en horas, entre paréntesis.

     Antes del nombre de cada tarea se escribe el identificador del ítem al cual corresponde. Por ejemplo: "ID-02 - Diseñar pruebas para la consulta de productos por tipo". Esto permite evitar confusiones cuando hay tareas muy similares, pero que corresponden a ítems diferentes.


Revisar el plan:

6. Se revisa que cada tarea tenga un tiempo máximo de 4 horas. Si hay tareas de más tiempo se deben dividir en otras más pequeñas. Por ejemplo, si se tenía una tarea de "elaborar el código para la consulta de ventas", se puede dividir (por ejemplo) en "elaborar el código de acceso datos para la consulta de ventas" y "elaborar el código con las reglas de negocio para la consulta de ventas".
7. Se revisa que las horas para las tareas de cada ítem de desarrollo corresponda (aproximadamente) a los días definidos para ese ítem. En caso de que la diferencia sea muy grande se deben revisar los tiempos y tamaños, para ajustar lo que sea pertinente. Todos los participantes deben quedar satisfechos con el plan, y comprometidos con el logro de los objetivos planteados.

**Actividad 5.2: Requisitos**

**Entradas**:

- Documento de visión.
- Repositorios del proyecto.

**Salidas**:

Requisitos.

**Roles participantes**:

Equipo de desarrollo y representante del cliente.

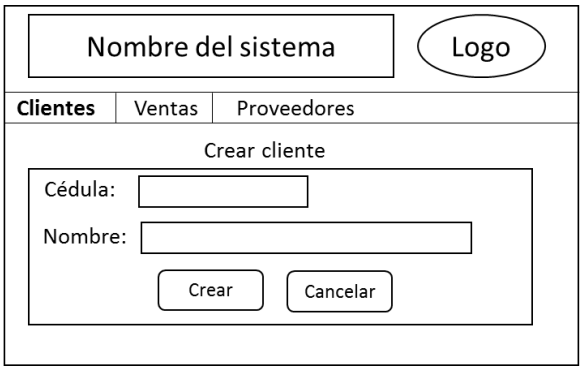
**Tareas**:

Previamente, durante la reunión de coordinación y seguimiento alguno de los miembros del equipo ha seleccionado una o más tareas de requisitos para llevar a cabo.

Obtener y especificar los requisitos:

1. Las personas encargadas analizan la información que se tiene de los ítems de desarrollo seleccionados (verificando los documentos en el repositorio del proyecto), para determinar cuáles pueden ser las técnicas apropiadas para recolectar la información.
2. Se realiza la reunión con los *stakeholders* (dependiendo de las técnicas seleccionadas) para obtener los requisitos.

   Durante la reunión se recomienda elaborar, de manera colaborativa, un boceto de la interfaz de usuario (sketch o *mockup*), y a medida que se elabora ir realizando preguntas sobre las entradas, el proceso, las salidas y los casos excepcionales. También se pueden realizar propuestas de formas alternativas de lograr los resultados.




   Es importante identificar las reglas del negocio asociadas a cada ítem de desarrollo, las cuales incluyen las validaciones que se deben realizar, los cálculos o procedimientos asociados, la forma de presentar resultados y el comportamiento del sistema ante entradas no válidas.

   Se especifican los requisitos recolectados usando el formato de requisitos.

Revisar y publicar los requisitos:

3. Los requisitos especificados en el formato deben ser revisados por otro miembro del equipo de desarrollo, quien revisará: que sean claros, completos y que se puedan probar. En caso de encontrar algún problema se debe realizar la corrección inmediatamente.

4. Se realiza una reunión con el representante del cliente para validar los requisitos especificados. El representante confirma si son claros, si usan términos del negocio (no técnicos) y -lo más importante- si corresponden a lo que se necesita.
   En caso de encontrar algún problema se debe realizar la corrección inmediatamente.
   Estos requisitos pasan a estado: **verificado y validado**.
5. Los documentos con la especificación de los requisitos se publican en el repositorio de documentos del proyecto, para que todos los participantes puedan consultarlos.

**Actividad 5.3: Modelado**

**Entradas**:

- Documento de visión.
- Documento de arquitectura.
- Requisitos.
- Repositorios del proyecto.

**Salidas**:

Modelos de diseño.

**Roles participantes**:

Equipo de desarrollo.

**Tareas**:

Previamente, durante la reunión de coordinación y seguimiento alguno de los miembros del equipo ha seleccionado una o más tareas de modelado para llevar a cabo.

Elaborar los modelos estructural y dinámico para el ítem de desarrollo:

1. Utilizando la vista lógica de la arquitectura como referente, se identifican los componentes generales que tendrá la solución (por ejemplo: vista, control, datos) y cuál es la interfaz de comunicación entre cada uno de ellos.
   Para cada componente se establece con cuáles elementos concretos se implementará. Puede ser con clases, páginas web, funciones, etc. Se debe considerar la infraestructura seleccionada para el proyecto, que incluye el software de desarrollo y los marcos de trabajo o *frameworks*.

Estos elementos se incluyen en un diagrama, que será el modelo estructural.
En el caso de programas orientados a objetos, puede ser un **diagrama de clases**.

2. Para cada elemento se establecen los principales servicios (métodos u operaciones) que tendrá, lo cual se define considerando los requisitos que se estén trabajando y el tipo de componente al cual corresponda.
3. Se definen las relaciones entre los elementos, conservando los lineamientos de comunicación definidos en la arquitectura cuando se trate de partes que sean de diferentes componentes.
4. (Opcional) Para requisitos complejos se puede elaborar un diagrama de secuencia que muestre cómo sería el proceso de comunicación entre las partes; es decir, el modelo dinámico. Al elaborar este diagrama se pueden refinar los servicios asociados a cada elemento, ya sea porque se identifican nuevos servicios, se descartan algunos o se modifica la información que recibe o retorna. También se pueden refinar las relaciones o los mismos elementos (nuevas clases o clases innecesarias, por ejemplo).

Revisar y publicar los modelos:

5. Quien elaboró los modelos debe revisarlos para verificar que cumplan con los principios de diseño y que se han considerado todos los aspectos de los requisitos (reglas del negocio y excepciones).
6. Los diagramas se publican en el repositorio de documentos del proyecto.
7. Se actualiza la información del requisito o requisitos asociados, cambiando su estado a: **modelado**.

**Actividad 5.4: Construcción**

**Entradas**:

- Documento de arquitectura.
- Requisitos.
- Modelos de diseño.
- Repositorio del proyecto.

**Salidas**:

Código de ítems de desarrollo.

**Roles participantes**:

Equipo de desarrollo.

**Tareas**:

Previamente, durante la reunión de coordinación y seguimiento alguno de los miembros del equipo ha seleccionado una o más tareas de construcción para llevar a cabo.

Elaborar las pruebas unitarias:

1. Se hace un listado de las pruebas que se pueden realizar, considerando: por lo menos una prueba para el caso válido (cuando todo sale bien) y por lo menos una prueba para un caso no válido (cuando se tienen una o más entradas incorrectas, alguna validación que se deba realizar). Esto es para los métodos más complejos.
2. Se escriben las pruebas diseñadas en alguna herramienta de pruebas unitarias. Por lo general las pruebas quedan en una carpeta separada pero que hace parte de todo el código del proyecto.

Escribir y probar las unidades del código:

3. Se escribe el código siguiendo los estándares definidos.
4. Cada vez que se completa una parte se ejecutan las pruebas unitarias que tenga asociadas para verificar su comportamiento. En caso de encontrar defectos, se corrigen inmediatamente.
5. Cuando el código pasa las pruebas se guarda una nueva versión en el repositorio local.

Realizar un análisis estático de código:

6. Se ejecuta un análisis estático de código (usando una herramienta para este fin).
7. Se analizan los resultados obtenidos, para identificar posibles advertencias y se corrigen inmediatamente las advertencias de mayor gravedad.

Publicar el código en el repositorio:

8. Al terminar de codificar el ítem o ítems:
    - Se guarda una versión en el repositorio de código del proyecto, por lo general en la rama que le corresponde al desarrollo actual.
    - Se actualiza la información del requisito o requisitos asociados, cambiando su estado a: **codificado**.

- Se notifica que está listo para integración y pruebas.

**Actividad 5.5: Integración y pruebas**

**Entradas**:

- Documento de arquitectura.
- Requisitos.
- Modelos de diseño.
- Código de los ítems de desarrollo.
- Repositorio del proyecto.

**Salidas**:

- Casos de prueba.
- Reportes de defectos.
- Incremento (puede ser también Solución de Software).

**Roles participantes**:

Equipo de desarrollo.

**Tareas**:

Previamente, durante la reunión de coordinación y seguimiento alguno de los miembros del equipo ha seleccionado una o más tareas de integración o pruebas para llevar a cabo.

Diseñar los casos de prueba:

1. Se aplica la técnica de partición equivalente para obtener un conjunto de casos de prueba a partir de los requisitos.
2. Los casos de prueba obtenidos con la técnica se especifican en el formato de casos de prueba. Para cada caso de prueba se debe incluir, como mínimo, un identificador, una descripción, los valores de entrada y los resultados esperados.
3. Considerando los valores de entrada que se establecieron para los casos de prueba, se crea un conjunto de datos que cumpla con lo requerido para que las pruebas funcionen apropiadamente. Estos datos pueden estar en un archivo o en una base de datos.

En caso de ser una base de datos se genera el script SQL para insertar los datos. De esta manera se facilita la ejecución de las pruebas.

4. El documento de casos de prueba se publica en el repositorio de documentos del proyecto, junto con el script SQL para crear datos de prueba (si se tiene).
5. Se revisa si existen atributos de calidad (en el documento de arquitectura) que estén relacionados con los ítems de desarrollo elaborados. Si es el caso entonces se adicionan casos de prueba correspondientes (pruebas no funcionales).

Ejecutar las pruebas:

6. Se configura el entorno de pruebas de acuerdo con la configuración requerida y se cargan los datos necesarios (por ejemplo, ejecutando el script creado previamente).
7. Se ejecutan los casos de prueba diseñados, y se van anotando los resultados en el mismo formato. Para cada prueba se indica si no se pudo ejecutar (bloqueada), si pasó o si falló, es decir, no se obtuvieron los resultados esperados. En caso de no poder ejecutar alguna prueba o no obtener los resultados esperados se explica un poco más el problema (para facilitar la corrección).
8. Si no se encuentran defectos se procede con la sección “Terminar las pruebas”. En caso contrario continúa con la sección de “Depurar”.

Depurar:

9. Los desarrolladores designados revisan los reportes de defectos y proceden a realizar la depuración, corrigiendo los defectos.
10. Se ejecutan todas las pruebas unitarias (si se tienen), para verificar que pasan.
    Al finalizar la corrección:
    - Se guarda la nueva versión del código en el repositorio del proyecto (en la rama correspondiente).
    - Se notifica que está listo para pruebas.
11. Regresa a la sección de “Ejecutar las pruebas”.

Terminar las pruebas:

12. Se actualiza la información del requisito o requisitos, quedando en estado: **probado**
13. Se guarda la nueva versión del incremento, que está conformado por los ítems de desarrollo integrados y probados, en el repositorio de código del proyecto.

Esto, por lo general, corresponde a unificar la rama del desarrollo con la rama principal en el repositorio.

Realizar pruebas de aceptación:

14. El representante del cliente realiza pruebas del incremento para verificar que se cumplan los requisitos de acuerdo con las necesidades del negocio.
15. Si el representante del cliente da su visto bueno se actualiza la información del requisito o requisitos, quedando en estado: **aprobado**.
    Si se encuentran inconformidades por parte del cliente, éstas se registran como defectos, y se vuelve a la sección de "Depurar".

Desplegar:

16. Si es la primera iteración, se realiza la configuración del entorno de producción.
    Para esto se debe contar con el permiso y los recursos del cliente.
17. (Opcional) Se despliega el código del incremento en el entorno de producción, donde los usuarios podrán hacer uso de las funcionalidades entregadas.
    En este caso se actualiza la información de los requisitos, para indicar que quedan en estado: **desplegado**.
18. Si se trata de la última iteración de todo el proyecto, entonces se realiza una copia de la última versión del repositorio de código del proyecto, que incluya todos los incrementos desarrollados, para conformar la Solución de Software que se entregará al cliente.

**Actividad 5.6: Seguimiento y adaptación**

**Entradas**:

- Plan del proyecto.
- Tablero de la iteración.

**Salidas**:

Tablero de la iteración (actualizado).

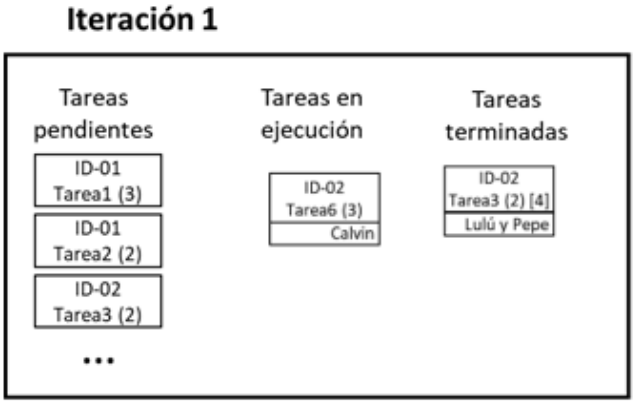
**Roles participantes**:

Gestor del proyecto y equipo de desarrollo.

**Tareas**:

Actualizar tareas terminadas:

1. Cada día (o día de por medio), a una hora acordada previamente, los integrantes del equipo de desarrollo y el gestor del proyecto se reúnen para revisar el tablero de la iteración.
2. Cada persona indica si terminó la tarea o tareas en las que estaba trabajando. En caso de haberla terminado, cambia la tarjeta a la columna "Tareas terminadas", y anota, entre corchetes, el tiempo real invertido.




   En caso de no haber terminado la tarea, se actualiza la información del tiempo estimado, pero la tarjeta queda en la misma columna ("Tareas en ejecución").

Asignar tareas:

3. Cada persona determina en cuál o cuáles tareas trabajará a continuación. Se deben lograr consensos entre todos los miembros del equipo, para que no se queden personas sobrecargadas. También es importante motivar el aprendizaje de las personas, de manera que es bueno ir rotando por diferentes tipos de actividades.
4. Se cambian las tareas seleccionadas en el tablero de la iteración, pasando de la columna "pendiente" a la columna "en ejecución". En cada tarea se escribe el nombre de la persona o personas encargadas, para que esto quede visible en el tablero.

Realizar otros ajustes:

5. Si hay algún punto de coordinación importante, se debe dar a conocer. El equipo asigna a una o más personas para que lo trabajen justo después de la reunión. No se debe tratar de solucionar durante la reunión, pues se puede volver poco productiva.

6. El gestor del proyecto toma una foto o pantallazo del estado del tablero de la iteración, de manera que quede como registro del seguimiento realizado. Esta evidencia se guarda en el repositorio de documentos del proyecto.

**Actividad 5.7: Revisión**

**Entradas**:

- Plan del proyecto.
- Incremento.
- Tablero de la iteración.

**Salidas**:

- Informe de revisión.

**Roles participantes**:

Todos los roles.

**Tareas**:

Preparar la revisión:

1. Se revisa el plan del proyecto y el tablero de la iteración para tener claros los siguientes datos:
   - Ítems de desarrollo planeados.
   - Ítems de desarrollo terminados (revisados por el representante del cliente).
   - Cantidad de días u horas estimadas.
   - Cantidad de días u horas invertidos.
   - Cantidad de horas que se estima faltan para terminar (en caso de no haber terminado todos los ítems).
   - Cantidad de ítems de desarrollo que faltan para el proyecto.
2. En caso de que no se hayan terminado todos los ítems, se analiza, junto con el representante del cliente y el equipo de desarrollo, las posibles razones.
3. Se invita a los interesados que participarán en la reunión.
4. De acuerdo con la cantidad de participantes, se configura un entorno de pruebas en varios equipos de cómputo, con el incremento desarrollado en la iteración.

Solo se deben incluir los ítems de desarrollo terminados. Los demás ítems (así estén avanzados), no deben incluirse para no generar falsas expectativas en los usuarios.

Realizar la revisión:

5. El gestor del proyecto y el representante del cliente presentan el análisis de los resultados de la iteración, resaltando los logros alcanzados. También se presentan las dificultades que se presentaron y lo que quedó sin terminar.

   Para terminar esta parte se pueden mostrar, de manera resumida, los ítems de desarrollo que faltan para terminar el proyecto, de manera que los participantes puedan tener una idea del avance global.
6. Se invita a los participantes a que hagan uso de la iteración desarrollada. Para esto se les da un tiempo definido (por ejemplo, media hora).

Recibir apreciaciones:

7. Se invita a los participantes de la reunión a opinar sobre la utilidad que el incremento presenta para su negocio y sobre el logro de los objetivos de la iteración.

   Si se realizan solicitudes de cambios a los ítems ya desarrollados o se piden nuevas funcionalidades, estas solicitudes deben quedar registradas.

Registrar los resultados de la revisión:

8. Se registran los resultados de la revisión en el formato correspondiente, incluyendo el resumen de lo que se presentó, los aportes de los participantes y lo que notaron los desarrolladores durante la sesión de pruebas informal.

   Se guarda en el informe en el repositorio de documentos del proyecto.

**Actividad 5.8: Retrospectiva**

**Entradas**:

- Tablero de la iteración.
- Registro de retrospectivas previas.

**Salidas**:

Registro de retrospectiva

**Roles participantes**:

Equipo de desarrollo.

**Tareas**:

Analizar aspectos positivos y negativos:

1. Si ya se han realizado retrospectivas previamente (en otras iteraciones), el equipo analiza brevemente cuáles fueron los resultados de los cambios implementados.
   Si los cambios dieron buenos resultados, se tienen en cuenta como aspectos positivos.
2. Los integrantes del equipo presentan lo que consideran positivo y negativo de la iteración que acabó de pasar. Se debe hacer énfasis en **aspectos del proceso** (planeación, coordinación, revisiones, productividad, etc.), no en problemas del producto.
3. Los participantes revisan las intervenciones y seleccionan las tres más importantes de las positivas y las tres más importantes de las negativas.
   Los aspectos positivos y negativos seleccionados se anotan en el registro de la retrospectiva.

Realizar propuestas de mejora:

4. Cada participante presenta propuestas para superar uno o más de los aspectos negativos.
5. Una persona recolecta todas las ideas y las escribe. Se revisa el registro de la retrospectiva anterior, para verificar si hay ideas que puedan considerarse de nuevo para esta iteración, y se escriben también.
   Es importante que se trate de cambios concretos, propuestas que inviten a la acción y a realizar modificaciones en la forma de trabajar. No se deben dejar escritas como buenas intenciones. Por ejemplo, no sería adecuado decir: “Hacer más revisiones”, sino que se puede concretar un poco más, como: “Planear y hacer por lo menos una revisión adicional a los requisitos”, o “Cambiar la definición de terminado para que el código deba incluir una revisión por un par”.

Seleccionar propuesta:

6. Se seleccionan por consenso cuál será la propuesta que se implementará en la siguiente iteración.
7. Se escribe la idea seleccionada en el registro de la retrospectiva, y se publica en el repositorio de documento del proyecto.

## Actividad 6: Hacer entrega final

**Entradas**:

- Documento de visión.
- Plan del proyecto.
- Solución de software.

**Salidas**:

- Acta de entrega.

**Roles participantes**:

Representante del cliente y gestor del proyecto.

**Descripción**:

Se recopila y organiza la documentación requerida para poder entregarla formalmente al cliente. También se pueden realizar algunas tareas adicionales como capacitaciones o migración de datos para garantizar que el producto queda en óptimas condiciones de funcionamiento.

Con esta actividad se formaliza la finalización del proyecto, lo cual es de especial importancia en algunos contratos, por lo que se solicita que se firme un acta confirmando la entrega a satisfacción de la solución de software. A partir de este momento las extensiones o cambios solicitados en el software se pueden considerar mantenimientos o nuevos proyectos de desarrollo.