RUP - Rational Unified Process

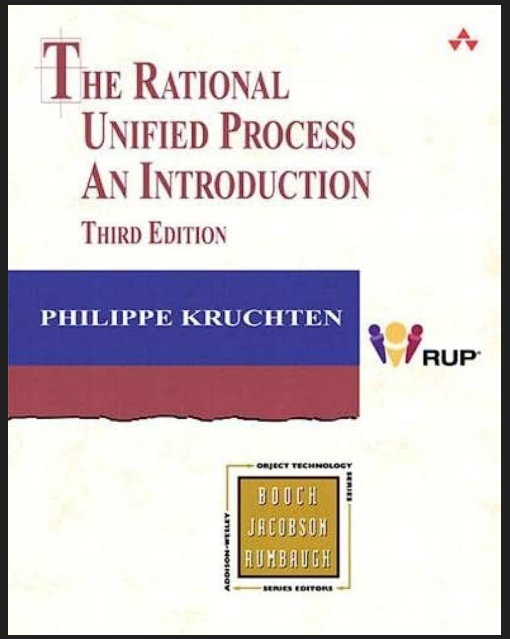
Fausto Panello

¿Qué es RUP?

El Proceso Racional Unificado o RUP (por sus siglas en inglés de Rational Unified Process) es un proceso de desarrollo de software desarrollado por la empresa Rational Software, actualmente propiedad de IBM. Junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

Origen de RUP

Los orígenes de RUP se remontan al modelo espiral original de Barry Boehm. Ken Hartman, uno de los contribuidores claves de RUP colaboró con Boehm en la investigación. En 1995, Rational Software compró una compañía sueca llamada Objectory AB, fundada por Ivar Jacobson, famoso por haber incorporado los casos de uso a los métodos de desarrollo orientados a objetos. El Rational Unified Process fue el resultado de una convergencia de Rational Approach y Objectory (el proceso de la empresa Objectory AB). El primer resultado de esta fusión fue el Rational Objectory Process, la primera versión de RUP, fue puesta en el mercado en 1998, siendo el arquitecto en jefe Philippe Kruchten, quien escribió un libro “The Rational Unified Process - An Introduction”, en 1999.



RUP - Definición

RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables, que se apoya en diversos artefactos, incluido en el software Rational Method Composer (RMC) como herramienta de aplicación para este método. Trabaja en conjunto con **UML**.

Originalmente se diseñó un proceso genérico y de dominio público, el Proceso Unificado, y una especificación más detallada, el Rational Unified Process, que se vendiera como producto independiente.

RUP - Casos de Uso

En el proceso unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones.

La idea es que en cada iteración, tome un conjunto de **casos de uso** y desarrolle todo el camino por las disciplinas (análisis, diseño, pruebas, implementación, etc.). Acá va un pequeño ejemplo:

Título: Procesar una venta en el punto de venta

Actores principales: Vendedor, Cajero, Comprador

Descripción: Este caso de uso describe cómo se procesa una venta en el punto de venta, desde la interacción del vendedor con el comprador hasta el pago y la finalización de la transacción por parte del cajero.

El vendedor registra los productos seleccionados por el comprador en el sistema de punto de venta.

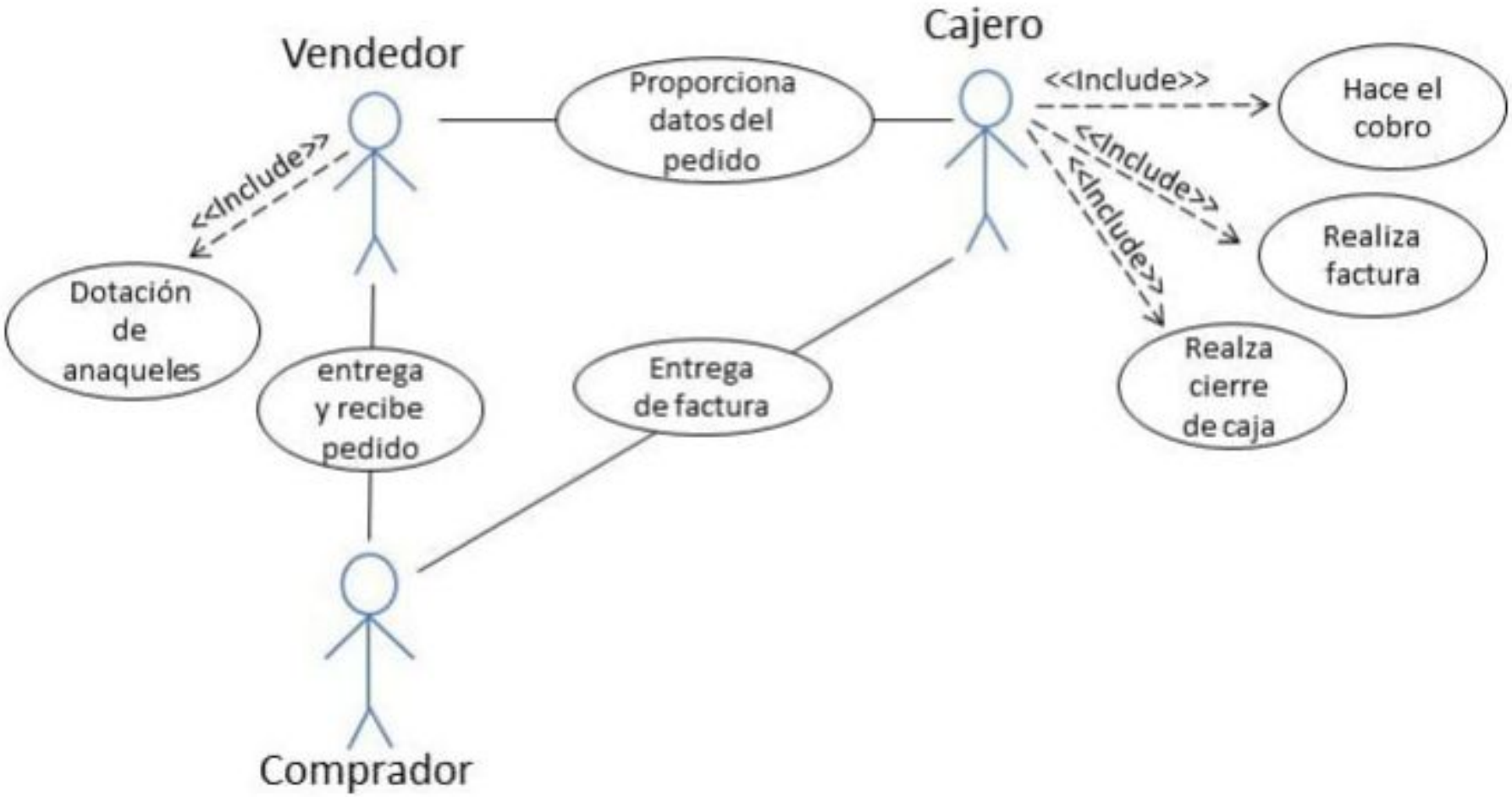
El sistema muestra una interfaz de registro de productos.

El vendedor escanea o ingresa manualmente los códigos de barras de los productos.

El sistema agrega los productos al carrito de compra y muestra el total acumulado.

El vendedor finaliza la selección de productos.

… seguiría el caso de uso.

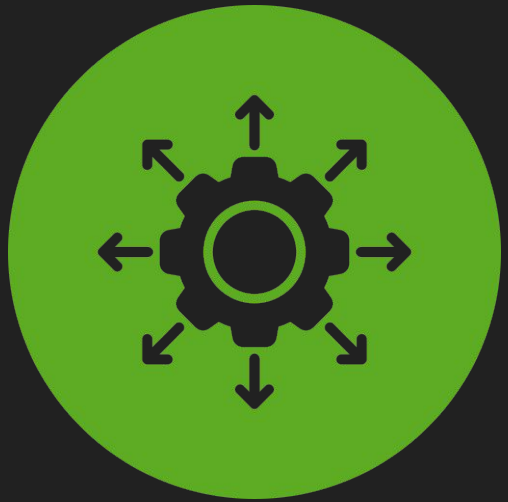

Vendedor
Cajero
Proporciona datos del pedido
<<Include>>
Hace el cobro
<<Include>>
<<Include>>
Realiza factura
Realza cierre de caja
<<Include>>
Dotación de anaqueles
entrega y recibe pedido
Entrega de factura
Comprador

RUP - Principios clave

La Filosofía del RUP está basado en 6 principios clave que son los siguientes:

- Adaptar el proceso
- Equilibrar prioridades
- Demostrar valor iterativamente
- Colaboración entre equipos
- Elevar el nivel de abstracción
- Enfocarse en la calidad

Vamos a ver cada uno de los principios.



RUP - Principios clave - Adaptar el proceso

El proceso deberá adaptarse a las necesidades del cliente ya que es muy importante interactuar con él. Las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto en un área sub formal.

RUP - Principios clave - Equilibrar prioridades

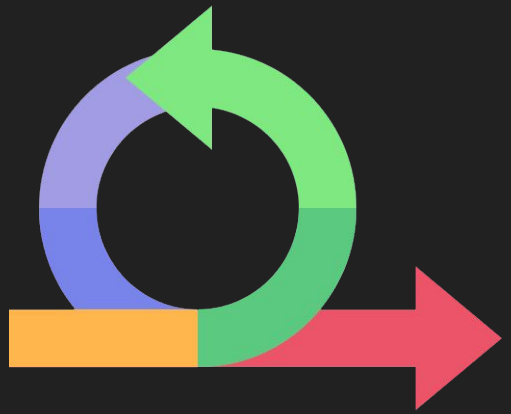
Los requisitos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un equilibrio que satisfaga los deseos de todos. Gracias a este equilibrio se podrán corregir desacuerdos que surjan en el futuro.

RUP - Principios clave - Colaboración entre equipos

El desarrollo de software no lo hace una única persona sino múltiples equipos. Debe haber una comunicación fluida para coordinar requisitos, desarrollo, evaluaciones, planes, resultados, etc.

RUP - Principios clave - Demostrar valor iterativamente

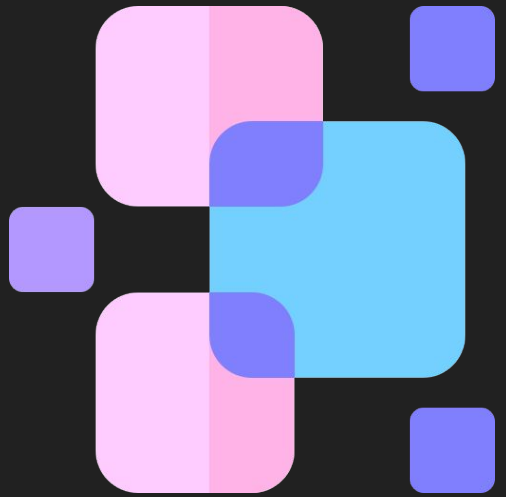
Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto, así como también los riesgos involucrados



RUP - Principios clave - Elevar el nivel de abstracción

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requisitos y sin comenzar desde un principio pensando en la reutilización del código.

Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo, con el lenguaje UML.



RUP - Principios clave - Enfocarse en la calidad

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

RUP - Ciclo de vida

El ciclo de vida RUP es una implementación del desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan pocas pero grandes y formales iteraciones en número variable según el proyecto.

Las fases son:

- Inicio
- Elaboración
- Construcción
- Transición

RUP - Fase de inicio

Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

RUP - Elaboración

En esta fase se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollan en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

RUP - Construcción

El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requisitos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

RUP - Transición

El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

Fases del Método RUP

Ventajas y desventajas

- Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo)..

- Método pesado
- Por el grado de complejidad puede ser no muy adecuado para proyectos de corto alcance.
- En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios.

Adecuado para proyectos:

- Al ser las iteraciones un proceso interno, aplica para sistemas de gran tamaño, donde las entregas no necesariamente sean un sistema completo preliminar (ejemplo: sistema operativo).

Introducción a UML

Fausto Panello

¿Qué es UML?

UML es un standard de OMG (Object Management Group) grupo dedicado a gestionar estándares de tecnología orientada a objetos.

Es un lenguaje de modelado. Un modelo es una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema.

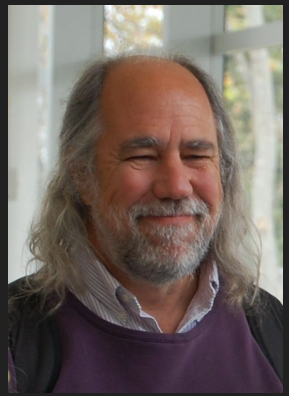
El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar.

UML es ante todo un **lenguaje**. Un lenguaje proporciona un **vocabulario** y una serie de **reglas** para permitir una comunicación.

En este caso, este lenguaje se centra en la representación gráfica de un sistema.

UML - Origen

El lenguaje UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos reputados investigadores en el área de metodología del software). El objetivo de ambos era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool). El primer borrador apareció en octubre de 1995.



En esa misma época otro reputado investigador, Jacobson, se unió a Rational y se incluyeron ideas suyas. Estas tres personas son conocidas como los “tres amigos”. Además, este lenguaje se abrió a la colaboración de otras empresas para que aportaran sus ideas. Todas estas colaboraciones condujeron a la definición de la primera versión de UML.

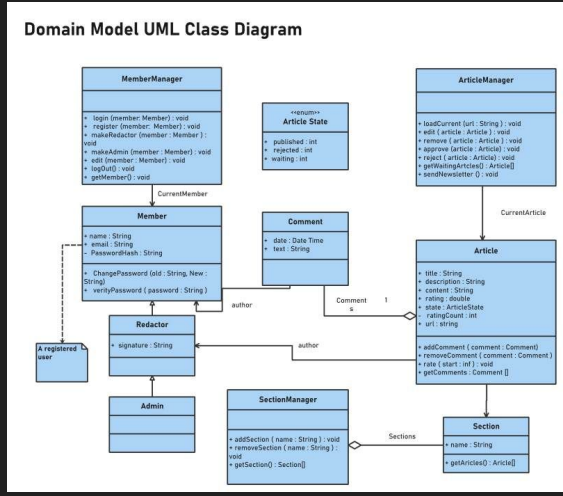
UML - Origen

Esta primera versión mencionada la diapositiva anterior se ofreció a un grupo de trabajo para convertirlo en 1997 en un estándar del OMG (Object Management Group http://www.omg.org). Este grupo, que gestiona estándares relacionados con la tecnología orientada a objetos (metodologías, bases de datos objetuales, CORBA, etc.), propuso una serie de modificaciones y una nueva versión de UML (la 1.1), que fue adoptada por el OMG como estándar en noviembre de 1997. Desde aquella versión ha habido varias revisiones que gestiona la OMG Revision Task Force.

UML - Modelado visual

Tal como indica su nombre, UML es un lenguaje de modelado. Un modelo es una simplificación de la realidad. El objetivo del modelado de un sistema es capturar las partes esenciales del sistema. Para facilitar este modelado, se realiza una abstracción y se plasma en una notación gráfica. Esto se conoce como modelado visual.

El modelado visual permite manejar la complejidad de los sistemas a analizar o diseñar. De la misma forma que para construir una choza no hace falta un modelo, cuando se intenta construir un sistema complejo como un rascacielos, es necesario abstraer la complejidad en modelos que el ser humano pueda entender.

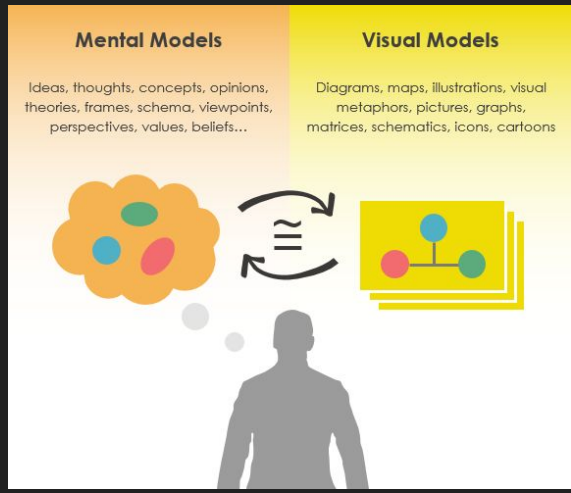


UML - Modelado visual

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.



UML - Objetivos

Recordemos que en la primera clase vimos “modelos”. Las metodologías de desarrollo siempre tienen como objetivo interpretar y leer los modelos, pero no cómo crearlos. Siempre hay que tener eso en cuenta antes de marcar los objetivos reales de una metodología de desarrollo. Dicho esto, las funciones, propias y naturales de UML incluyen:

- **Visualizar**: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar**: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir**: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Diagramar** estados
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

UML - Objetivos

Aunque UML está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es los suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (workflow) en una empresa, diseño de la estructura de una organización y por supuesto, en el diseño de hardware.

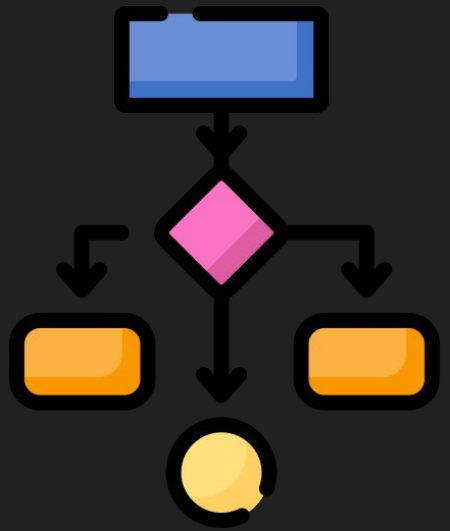
Un modelo UML está compuesto por **tres clases de bloques de construcción**:

- **Elementos**: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
- **Relaciones**: relacionan los elementos entre sí.
- **Diagramas**: Son colecciones de elementos con sus relaciones.

UML - Diagramas

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. UML incluye los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de colaboración.
- Diagrama de estados.
- Diagrama de actividades.
- Diagrama de componentes.
- Diagrama de despliegue.



UML - Diagramas

Si bien hay 9 diagramas en UML, nos vamos a centrar en:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de secuencia

El diagrama de estados también se usa mucho, pero no lo vamos a ver en principio.

UML - Diagrama de caso de uso

El diagrar
de uso qu
cada inter
represent
lo que tier

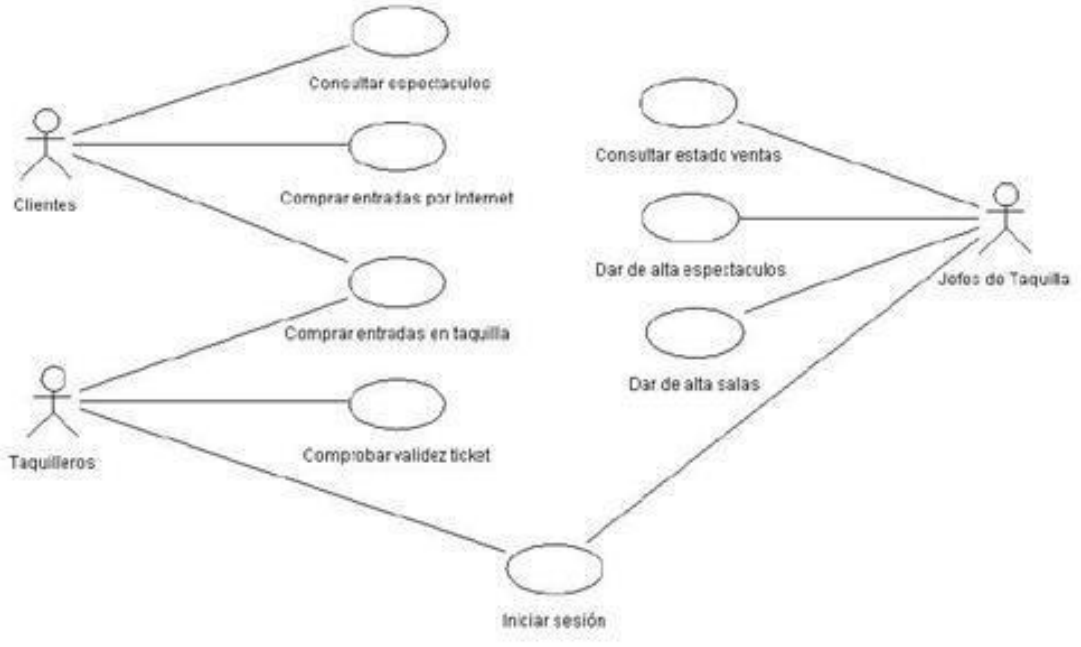


Figura 3: Diagrama de casos de uso

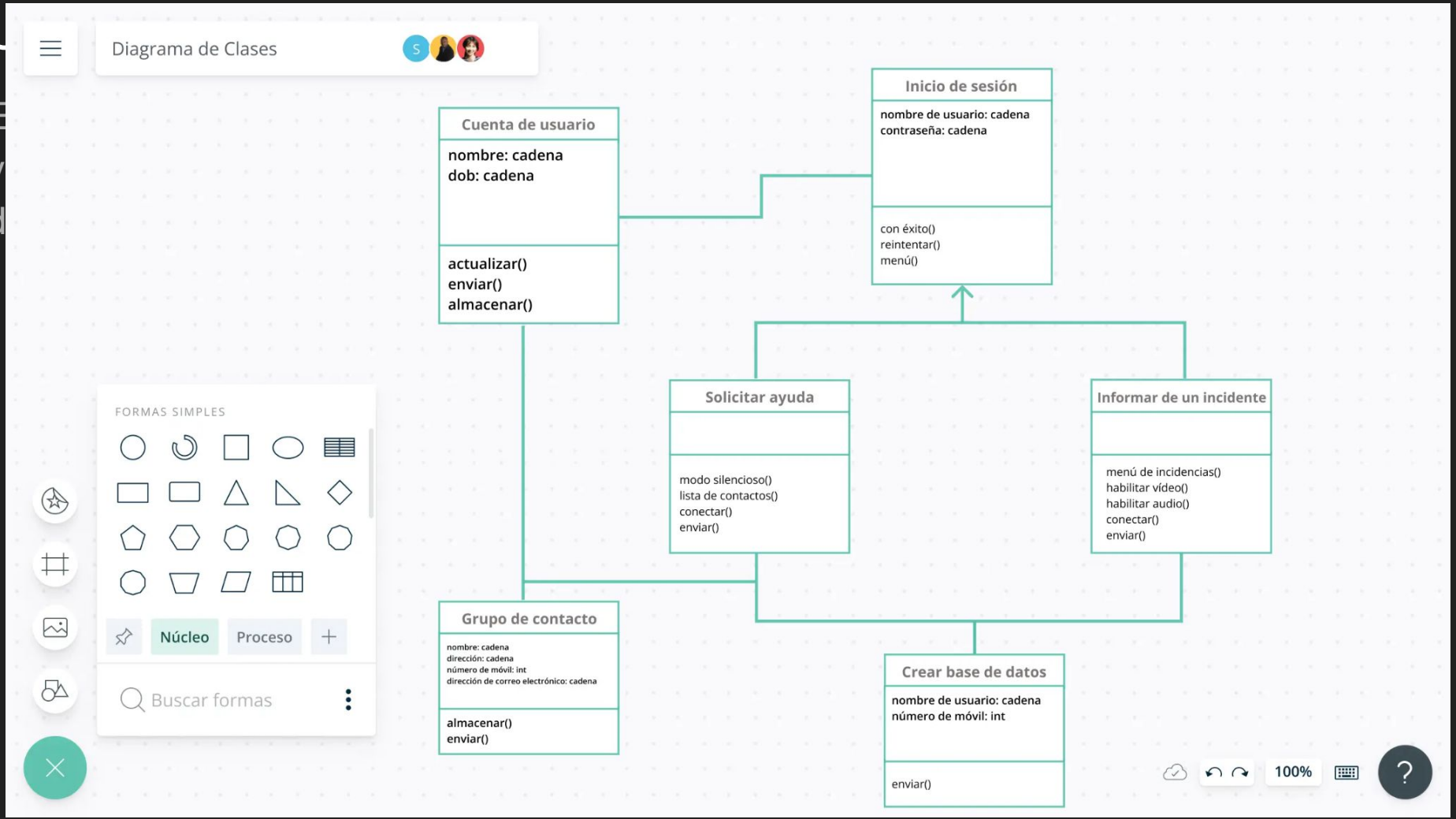


Diagrama de Clases
Cuenta de usuario
nombre: cadena
dob: cadena
actualizar()
enviar()
almacenar()
Inicio de sesión
nombre de usuario: cadena
contraseña: cadena
con éxito()
reintentar()
menú()
Solicitar ayuda
modo silencioso()
lista de contactos()
conectar()
enviar()
Informar de un incidente
menú de incidencias()
habilitar vídeo()
habilitar audio()
conectar()
enviar()
Grupo de contacto
nombre: cadena
dirección: cadena
número de móvil: int
dirección de correo electrónico: cadena
almacenar()
enviar()
Crear base de datos
nombre de usuario: cadena
número de móvil: int
enviar()
FORMAS SIMPLES
Núcleo
Proceso
Buscar formas
100%

UML - Diagrama de secuencia

En esta figura, el diagrama de secuencia muestra la interacción de los objetos que componen un sistema de forma temporal.

Se muestra la interacción de crear una nueva sala para un espectáculo, por ejemplo.

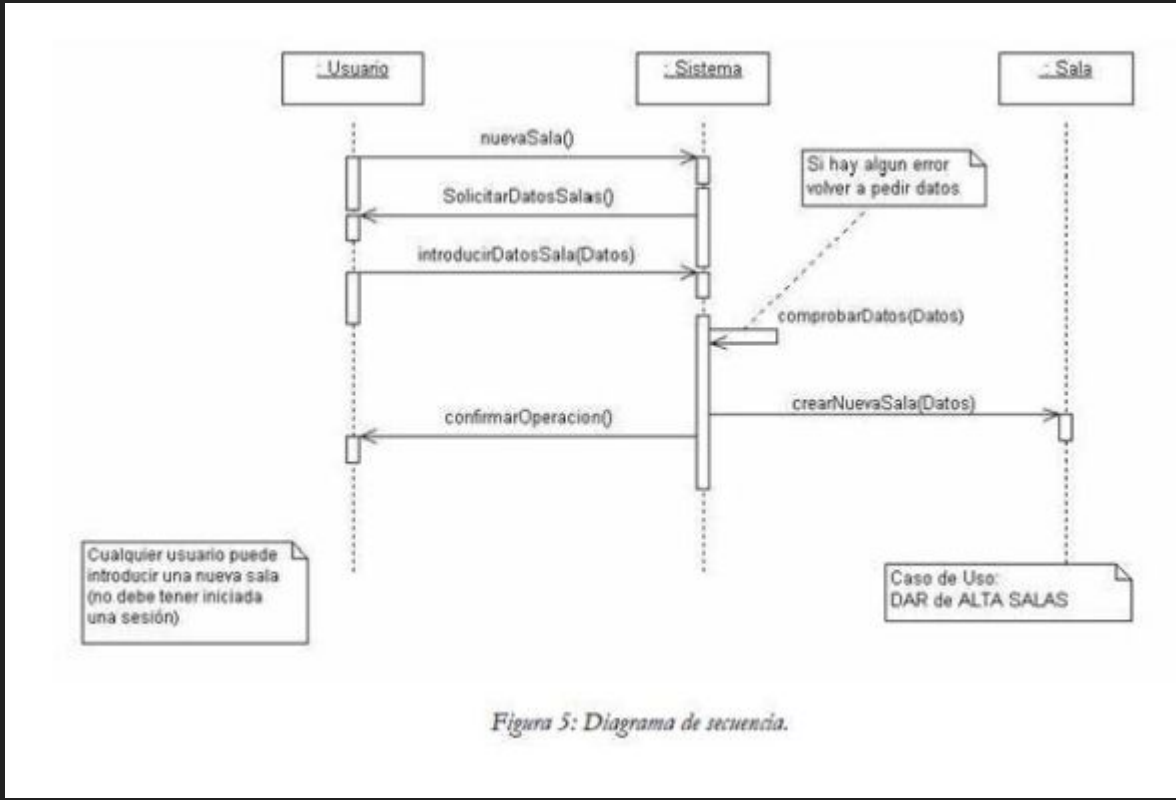


Figura 5: Diagrama de secuencia.

UML - Proceso de desarrollo

Aunque UML es bastante independiente del proceso de desarrollo que se siga, los mismos creadores de UML han propuesto su propia metodología de desarrollo, denominada el Proceso Unificado de Desarrollo.

El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidos. Además, el Proceso Unificado utiliza el UML para expresar gráficamente todos los esquemas de un sistema software. Pero realmente los aspectos que definen este Proceso Unificado son tres:

es iterativo e incremental,

dirigido por casos de uso,

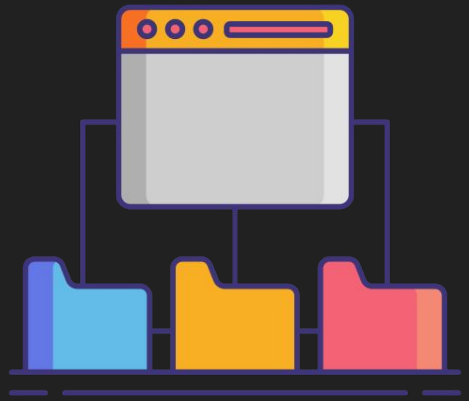
centrado en la arquitectura.

UML - Proceso de desarrollo - Iterativo e incremental

Todo sistema informático complejo supone un gran esfuerzo que puede durar desde varios meses hasta años. Por lo tanto, lo más práctico es dividir un proyecto en varias fases. Actualmente se suele hablar de ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración en el proyecto en la que se realizan varios tipos de trabajo (denominados flujos). Además, cada iteración parte de la anterior incrementando o revisando la funcionalidad implementada. Se suele denominar como “proceso”.

UML - Proceso de desarrollo - Dirigido por casos de uso

Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.



UML - Proceso de desarrollo - Centrado en la arquitectura

En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas, fontanería, etc. Cada uno de estos aspectos está representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema.

UML - Resumen

Resumiendo, el Proceso Unificado es un modelo complejo con mucha terminología propia, pensado principalmente para el **desarrollo de grandes proyectos**. Es un proceso que puede adaptarse y extenderse en función de las necesidades de cada empresa.