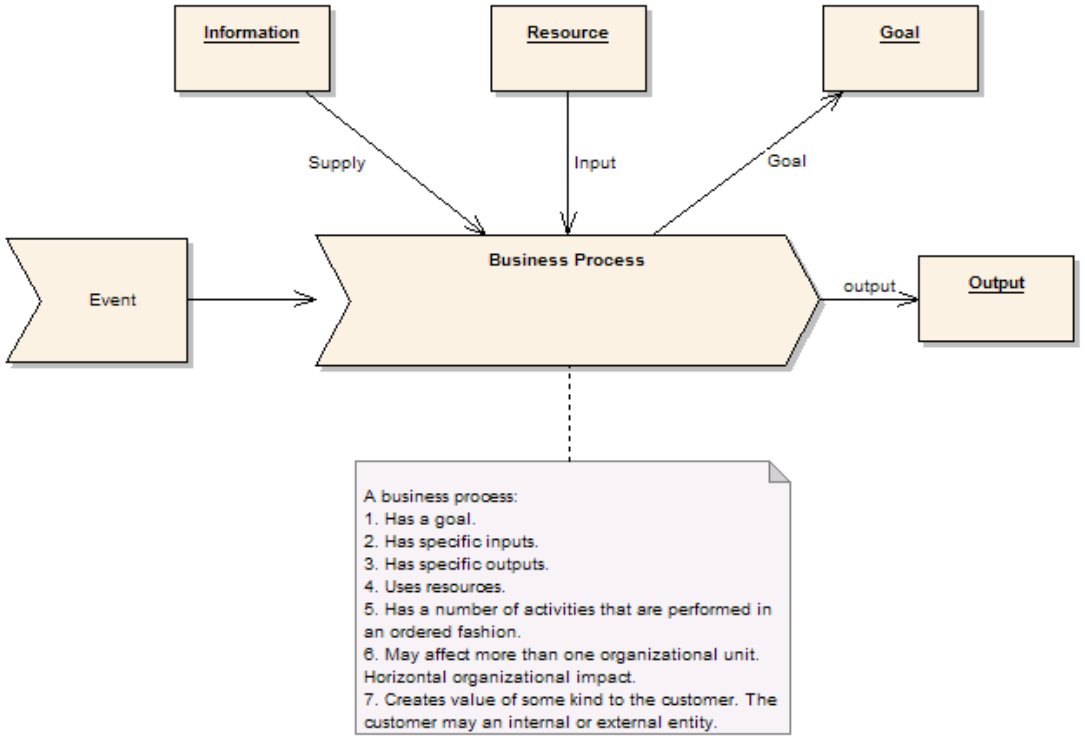
Proceso de Desarrollo

Conducido por Casos de Uso

1. Capturar un Modelo de Procesos de Negocio.



- Se utiliza para definir las actividades de alto nivel de negocios y procesos que ocurren en una organización y proporcionar una base para el modelo de casos de uso.

- El modelo de procesos de negocio normalmente captura más de lo que un sistema de software implementará (es decir, que incluye manuales y otros procesos).

2. Relacionar el Modelo de Casos de Uso con el Modelo de Procesos de Negocio

- Para definir exactamente qué funcionalidad tiene la intención de ofrecer desde la perspectiva del usuario de negocios, a medida que cada caso de uso se añade, crear un vínculo trazable desde el proceso de negocio apropiado para el caso de uso.

- Esta asignación indica claramente lo que la funcionalidad del nuevo sistema proporcionará para cumplir con los requerimientos del negocio indicados en el modelo de proceso. También asegura que no existen casos de uso sin un propósito.

3. Refinar los Casos de Uso

- Incluir los requisitos, limitaciones, rango de complejidad, notas y escenarios. Esta información sin ambigüedades describe lo que el caso de uso hace, cómo se ejecuta y las limitaciones en su ejecución.
- Verificar que el caso de uso sigue cumpliendo los requisitos de procesos de negocio.
- Incluir la definición de las pruebas del sistema para cada caso de uso para definir los criterios de ACEPTACIÓN para cada uno de ellos. También se incluyen algunos scripts de prueba de aceptación de los usuarios para definir la forma en que el usuario pondrá a prueba esta funcionalidad y cuáles son los criterios de aceptación.

4. Construir un Modelo de Dominio

- De las entradas y salidas del modelo de procesos de negocio y los detalles de los casos de uso, empezar a construir un modelo de dominio (objetos de negocio de alto nivel), diagramas de secuencia, diagramas de colaboración y los modelos de interfaz de usuario.

- Estos describen las "cosas" en el nuevo sistema, la forma en que esas partes interactúan y la interfaz que un usuario va a utilizar para ejecutar los escenarios de casos de uso.

5. Crear el Modelo de Clases

- A partir del modelo de dominio, el modelo de interfaz de usuario y los diagramas de escenario crear el modelo de clases.
  Se trata de una especificación precisa de los objetos en el sistema, sus datos o atributos y su comportamiento u operaciones.

- Los Objetos de dominio pueden ser abstraídos en jerarquías de clase mediante la herencia.
  Los Mensajes del diagrama de escenario normalmente representan operaciones de clase.
  Si se va a utilizar un marco existente o un patrón de diseño, se pueden importar los elementos del modelo existente para su uso en el nuevo sistema.

- Para cada clase definir las pruebas unitarias y pruebas de integración para probar completamente
  - i) las funciones de clase tal como se especifica internamente y que
  - ii) la clase interactúa con otras clases y componentes conexos como se esperaba.

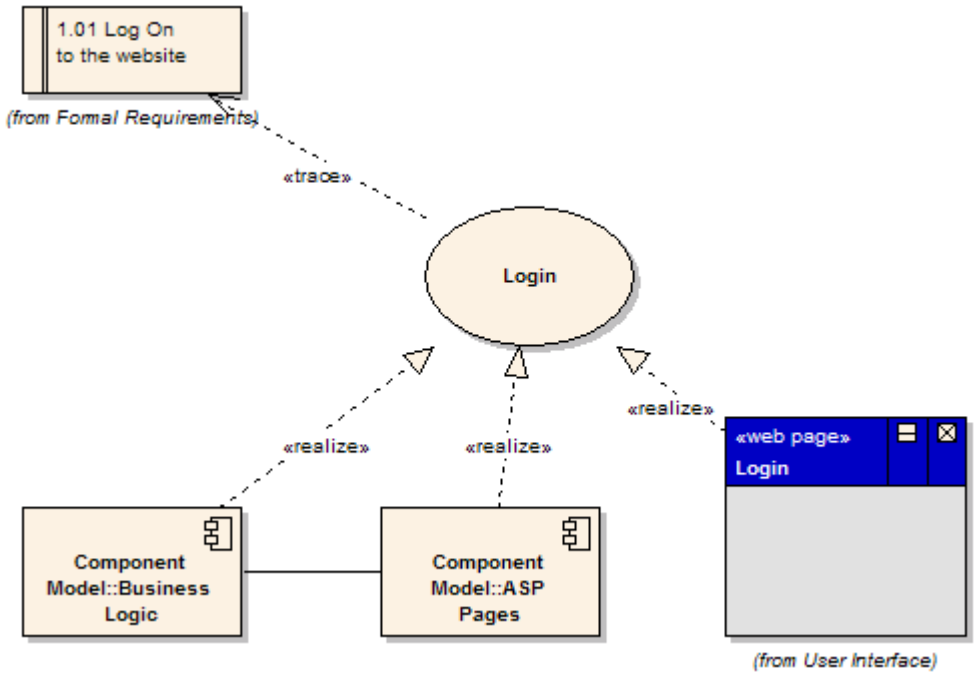
6. Diseñar un modelo de Componentes

- A medida que se desarrolla el modelo de clases, puede ser dividido en paquetes discretos y componentes.
- Un componente representa un pedazo de despliegue de software que recoge el comportamiento y los datos de una o más clases y expone una interfaz estricta a otros consumidores de sus servicios.
  Así que desde el modelo de clases se diseña un modelo de componentes para definir el embalaje lógico de clases.
- Para cada componente definir las pruebas de integración para confirmar que la interfaz del componente cumple con la especificación dada en relación a los elementos de software.

7. Recoger Requerimientos Adicionales

- Simultáneamente con el trabajo que se ha hecho, son capturados y documentados requisitos adicionales. Por ejemplo - los requisitos funcionales, requisitos de desempeño, los requisitos de seguridad, las responsabilidades, los planes de implementación, etc.
- Recoger estos requerimientos dentro del modelo y mantenerlo actualizado, a medida que el modelo madura.

8. Construir el Modelo de Implementación



- Define la arquitectura física del sistema.
- Este trabajo puede ser iniciado al principio, para capturar las características de despliegue físico - el tipo de hardware, sistemas operativos, capacidades de red, interfaces y software de apoyo que conforman el nuevo sistema, donde se desplegará y qué parámetros se aplican a la recuperación de desastres, la confiabilidad, copias de respaldo (back ups) y soporte.
- A medida que se desarrolla el modelo, la arquitectura física se actualizará para reflejar el sistema que se está proponiendo.

9. Construir y probar el sistema

- Tomar piezas discretas del modelo y asignar a uno o más desarrolladores. En un construcción guiada por casos de uso esto significa la asignación de un caso de uso para el equipo de desarrollo, teniendo que construir las pantallas, los objetos de negocio, las tablas de base de datos, y los componentes relacionados necesarios para ejecutar ese caso de uso.

- A medida que cada caso de uso se construye debe ir acompañado de las pruebas de unidad, de integración y del sistema. Una construcción impulsada por componentes puede ver los componentes discretos de software asignados a los equipos de desarrollo para su construcción.

10. Efectuar el seguimiento de los defectos

- Efectuar el seguimiento de los defectos que surgen en las fases de prueba contrastando los elementos del modelo relacionados
  - Defectos del sistema de prueba versus casos de uso,
  - Defectos de la prueba Unitaria versus clases, etc.
- Seguir cada cambio comparando con los elementos del modelo relacionado, para gestionar el "sobredimensionamiento del alcance".

11. Actualizar y refinar los modelos

- Actualizar y refinar los modelos a medida que avanza el trabajo - siempre evaluando el impacto de los cambios y refinamientos en el modelo en el trabajo posterior.
- Utilizar un enfoque iterativo para trabajar a través del diseño en trozos discretos, siempre evaluando la versión actual, los requisitos por delante y los descubrimientos que salen a la luz durante el desarrollo.

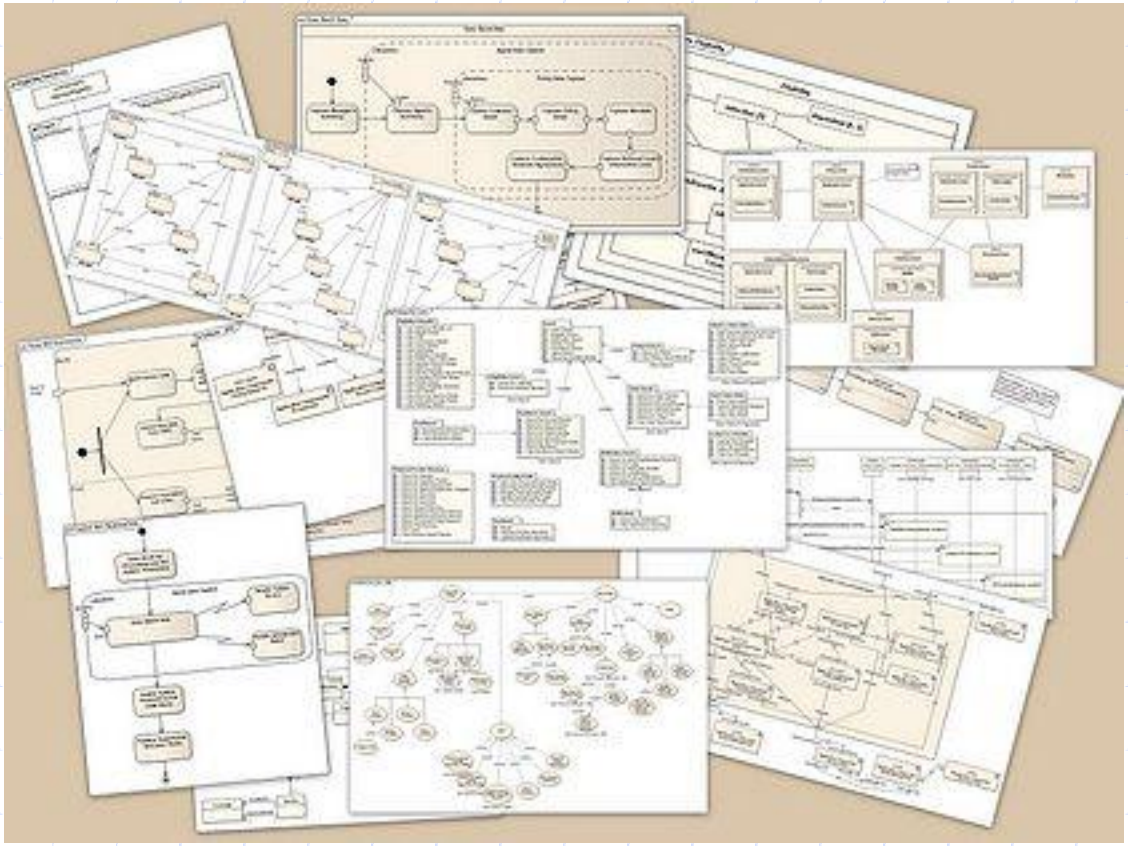
12. Implementar el software en Producción

- Entregar el software completo y probado antes de promoverlo al entorno de producción.
- Si se lleva a cabo una entrega por etapas, entonces esta migración de software desde el ambiente de prueba al de producción puede ocurrir varias veces durante la vida útil del proyecto.

UML

Diagramas de Estructura
Diagramas de Comportamiento
Diagramas de Interacción

UML



Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Aceptado como Estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Diagramas de Estructura

- Enfatizan en los elementos que deben existir en el sistema modelado:
  - Diagrama de clases
  - Diagrama de componentes
  - Diagrama de objetos
  - Diagrama de estructura compuesta (UML 2.0)
  - Diagrama de despliegue
  - Diagrama de paquetes

Diagramas de Comportamiento

- Enfatizan en lo que debe suceder en el sistema modelado:
  - Diagrama de actividades
  - Diagrama de casos de uso
  - Diagrama de estados

Diagramas de Interacción

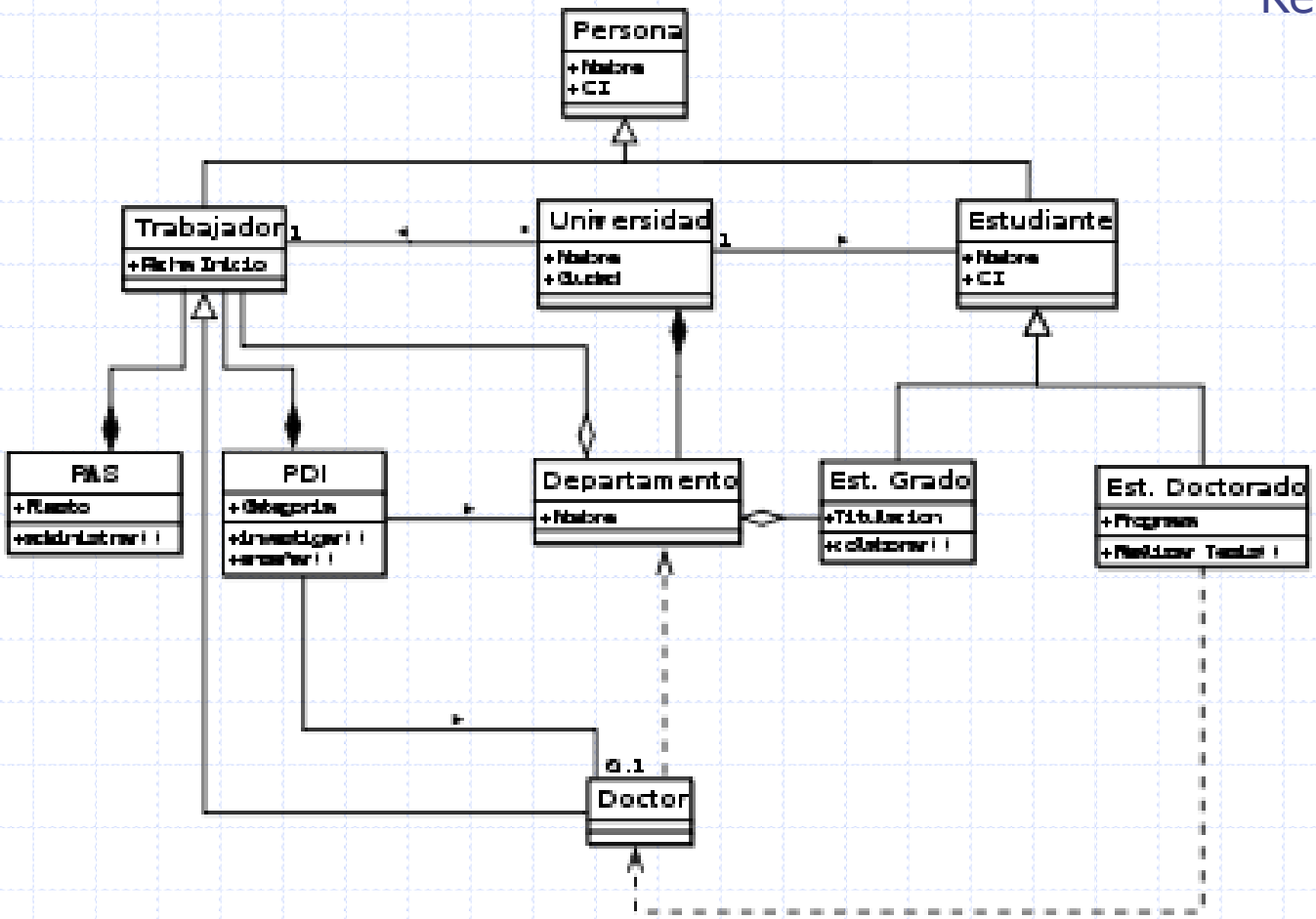
- Enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:
  - Diagrama de secuencia
  - Diagrama de comunicación, que es una versión simplificada del Diagrama de colaboración (UML 1.x)
  - Diagrama de tiempos (UML 2.0)
  - Diagrama global de interacciones o Diagrama de vista de interacción (UML 2.0)

Diagrama de Clases

- Tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.
- Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Diagrama de Clases

Diagrama de Clases



Representación de:

- Requerimientos en entidades y relaciones.
- La arquitectura conceptual de un dominio
- Soluciones de diseño en una arquitectura
- Componentes de software orientados a objetos

Diagrama de Clases - Definiciones

- **Propiedades** también llamados atributos o características, son valores que corresponden a un objeto, como color, material, cantidad, ubicación. Generalmente se conoce como la información detallada del objeto. Suponiendo que el objeto es una puerta, sus propiedades serían: la marca, tamaño, color y peso.

- **Operaciones** comúnmente llamados métodos, son aquellas actividades o verbos que se pueden realizar con/para este objeto, como por ejemplo abrir, cerrar, buscar, cancelar, acreditar, cargar. De la misma manera que el nombre de un atributo, el nombre de una operación se escribe con minúsculas si consta de una sola palabra. Si el nombre contiene más de una palabra, cada palabra será unida a la anterior y comenzará con una letra mayúscula, a excepción de la primera palabra que comenzará en minúscula. Por ejemplo: abrirPuerta, cerrarPuerta, buscarPuerta, etc.

Diagrama de Clases - Definiciones

- **Interfaz** es un conjunto de operaciones que permiten a un objeto comportarse de cierta manera, por lo que define los requerimientos mínimos del objeto. Hace referencia a polimorfismo.

- **Herencia** se define como la reutilización de un objeto padre ya definido para poder extender la funcionalidad en un objeto hijo. Los objetos hijos heredan todas las operaciones y/o propiedades de un objeto padre. Por ejemplo: Una persona puede especializarse en Proveedores, Acreedores, Clientes, Accionistas, Empleados; todos comparten datos básicos como una persona, pero además cada uno tendrá información adicional que depende del tipo de persona, como saldo del cliente, total de inversión del accionista, salario del empleado, etc.

Clases

- Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como ser una persona, un transporte, un documento o un paquete.
- Durante el proceso del diseño de las clases se definen las propiedades que identifican únivocamente al objeto y otras propiedades adicionales como datos que corresponden al objeto.

Clases

- Ejemplo 1: Una persona tiene número de documento de identificación, nombres, apellidos, fecha de nacimiento, género, dirección postal, posiblemente también tenga número de teléfono de casa, del móvil, FAX y correo electrónico.

- Ejemplo 2: Un sistema informático puede permitir administrar la cuenta bancaria de una persona, por lo que tendrá un número de cuenta, número de identificación del propietario de la cuenta, saldo actual, moneda en la que se maneja la cuenta.

- Ejemplo 3: Otro objeto pueden ser "Manejo de Cuenta", dónde las operaciones bancarias de una cuenta (como en el ejemplo 2) se manejarán realizando diferentes operaciones que en el diagrama de clases sólo se representan como operaciones:
  - Abrir
  - Cerrar
  - Depósito
  - Retiro
  - Acreditar Intereses

Jerarquía de los diagramas UML 2.0, mostrados como un diagrama de clases

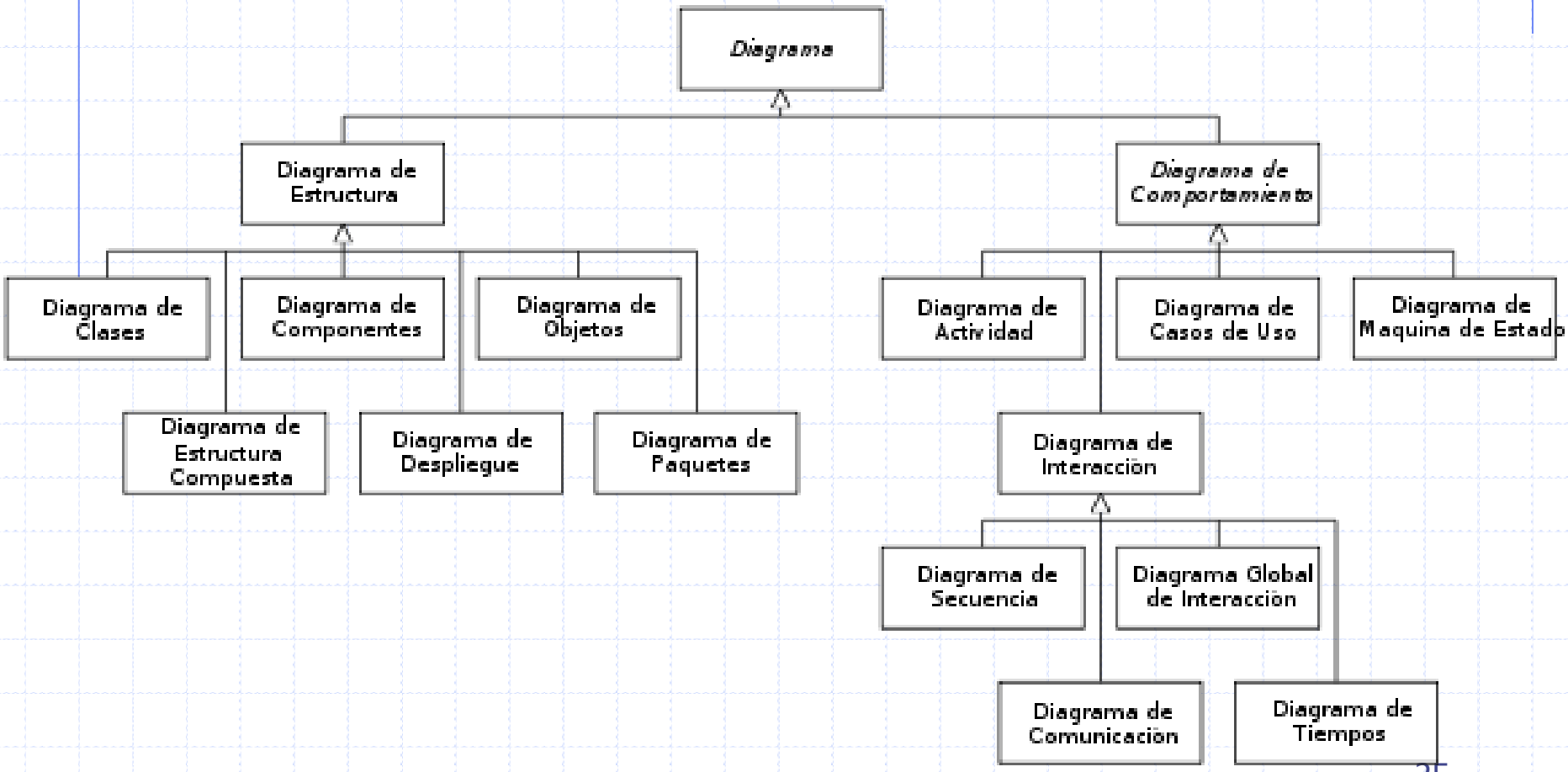


Diagrama de Componentes

- Representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes.
- Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes.
- Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Diagrama de Objetos

- Se puede considerar un caso especial de un diagrama de clases en el que se muestran instancias específicas de clases (objetos) en un momento particular del sistema.
- Los diagramas de objetos utilizan un subconjunto de los elementos de un diagrama de clase.
- Los diagramas de objetos no muestran la multiplicidad ni los roles, aunque su notación es similar a los diagramas de clase.

- Una diferencia con los diagramas de clase es que el compartimiento de arriba va en la forma Nombre de objeto: Nombre de clase
  - Por ejemplo, Miguel: Persona

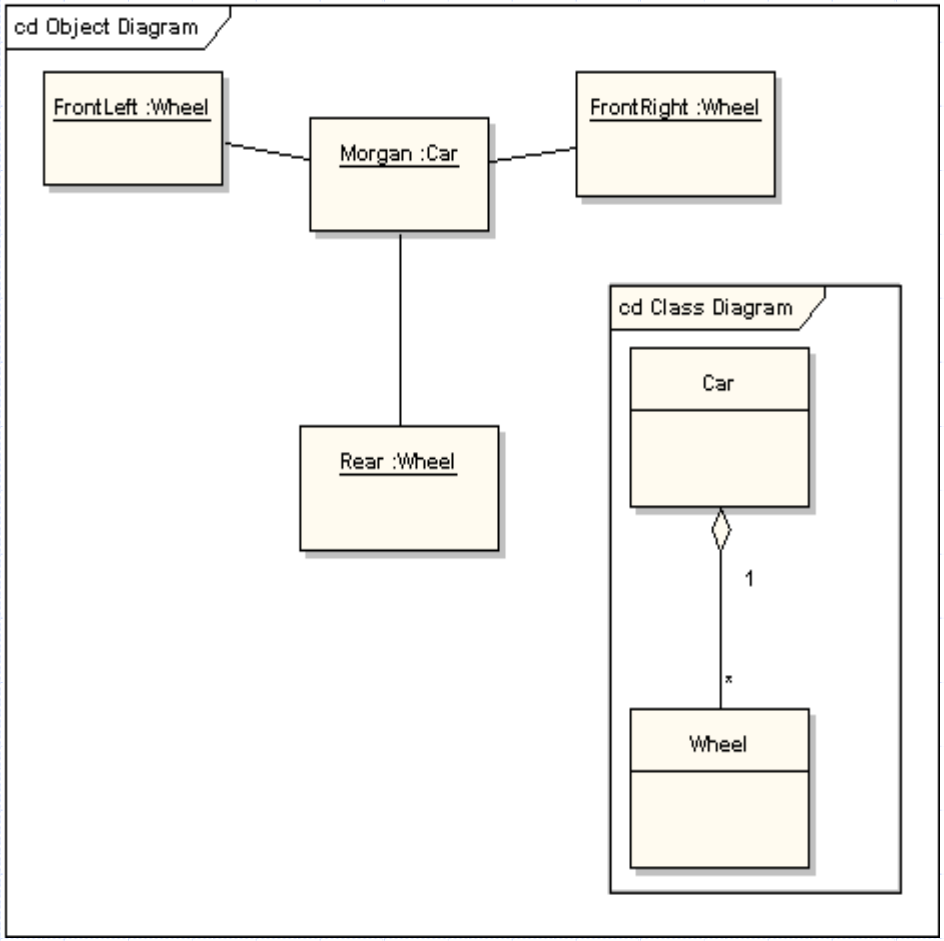


Diagrama de estructura compuesta

- Tipo de diagrama de estructura estática, que muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posibles.
- Esto puede incluir partes internas, puertas mediante las cuales, las partes interactúan con cada una de las otras o mediante las cuales, instancias de la clase interactúan con las partes y con el mundo exterior, y conectores entre partes o puertas.
- Una estructura compuesta es un conjunto de elementos interconectados que colaboran en tiempo de ejecución para lograr algún propósito.
- Cada elemento tiene algún rol definido en la colaboración

Diagrama de estructura compuesta

- Parte
  Una parte representa un rol jugado en tiempo de ejecución por una instancia de una clase o por una colección de instancias. La parte puede nombrar solamente un rol, una superclase abstracta, o puede nombrar una clase concreta específica. La parte puede incluir un factor de multiplicidad (cardinalidad), tal como el [0..*] mostrado para Viewer en el diagrama.

- Puerta
  Una puerta es un punto de interacción que puede ser usado para conectar clasificadores estructurados con sus partes y con el ambiente. Las puertas pueden opcionalmente especificar los servicios que proveen y los servicios que requieren de otras partes del sistema. En el diagrama, cada uno de los cuadrados pequeños es una puerta. Cada puerta tiene un tipo y esta etiquetado con un nombre, tal como "var", "indVar1", or "view" en el diagrama. Las puertas pueden contener un factor de multiplicidad, por ejemplo [3].

- Las puertas pueden ya sea delegar los requerimientos recibidos a partes internas, o pueden entregarlos directamente para el comportamiento del clasificador estructurado en el que la puerta está contenido. Las puertas públicas que son visibles en el ambiente son mostradas sobre el borde (límite o frontera), mientras que las puertas protegidas que no son visibles en el ambiente son mostradas dentro de la frontera (borde o límite). Todas las puertas en el diagrama son privadas, excepto por la puerta view a lo largo del límite derecho de FibonacciSystem.

Diagrama de estructura compuesta

- Conector
  Un conector une dos o más entidades, permitiéndoles interactuar en tiempo de ejecución. Un conector es representado por una línea que une una combinación de partes, puertas y clasificadores estructurados. El diagrama muestra tres conectores entre puertas, y un conector entre un clasificador estructurado y una parte.

- Colaboración
  Una colaboración es generalmente más abstracta que un clasificador estructurado. Esta es mostrada como un óvalo sin relleno conteniendo los roles que las instancias pueden jugar en la colaboración.

- Clasificador estructurado
  Un ClasificadorEstructurado representa una clase, frecuentemente una clase abstracta, cuyo comportamiento puede ser completa o parcialmente descrito mediante interacciones entre partes.

- Un ClasificadorEncapsulado
  es un tipo de clasificador estructurado que contiene puertas.

Diagrama de estructura compuesta

- En el siguiente diagrama, ambas Clases FibonacciSystem y Variable son clasificadores encapsulados, porque ambos tienen puertas a lo largo de sus límites.

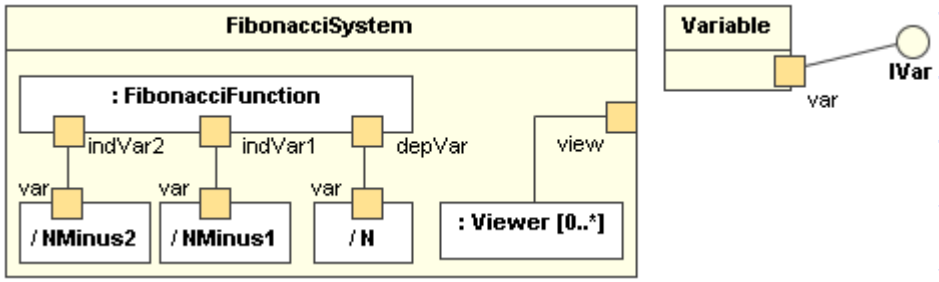


Diagrama de Despliegue

Tipo de diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes

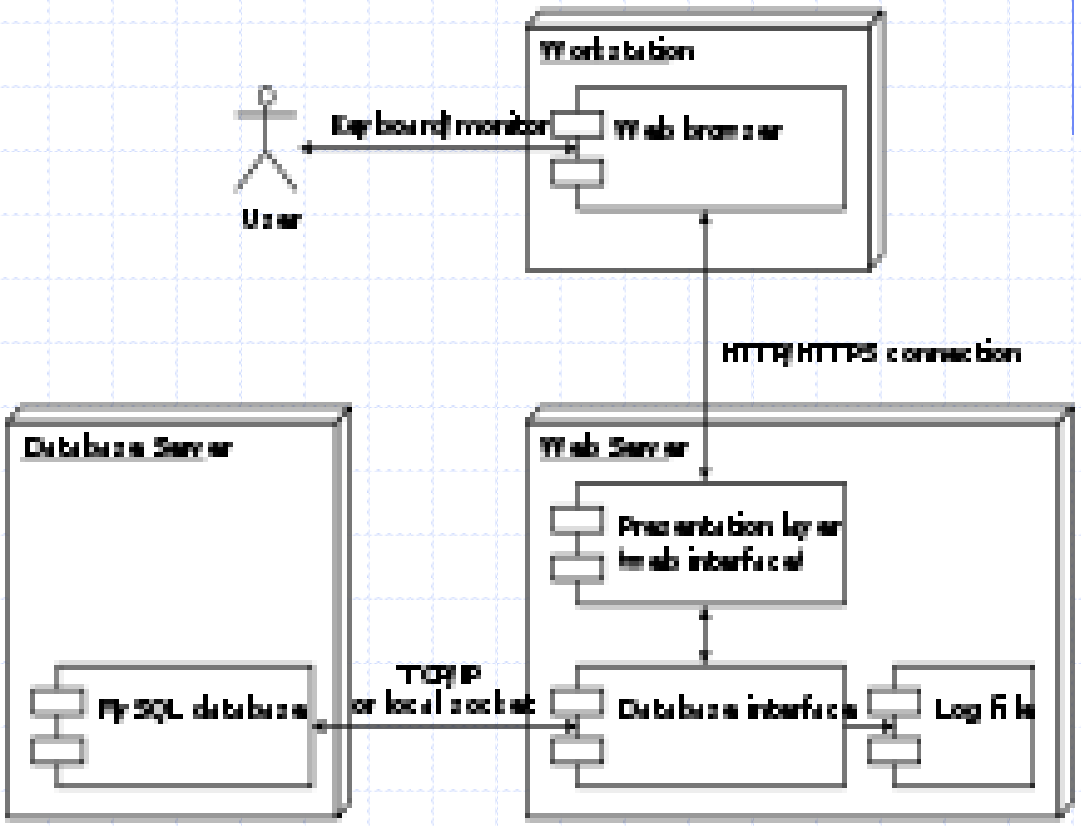


Diagrama de Paquetes

- Muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.
- Normalmente un paquete está pensado como un directorio, por lo que suministran una descomposición de la jerarquía lógica de un sistema.
- Los Paquetes están organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre ellos.
- Los paquetes son buenos elementos de gestión: Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

Diagrama de Actividades

Representa los flujos de trabajo paso a paso del negocio y operacionales de los componentes en un sistema.

Un Diagrama de Actividades muestra el flujo de control general.

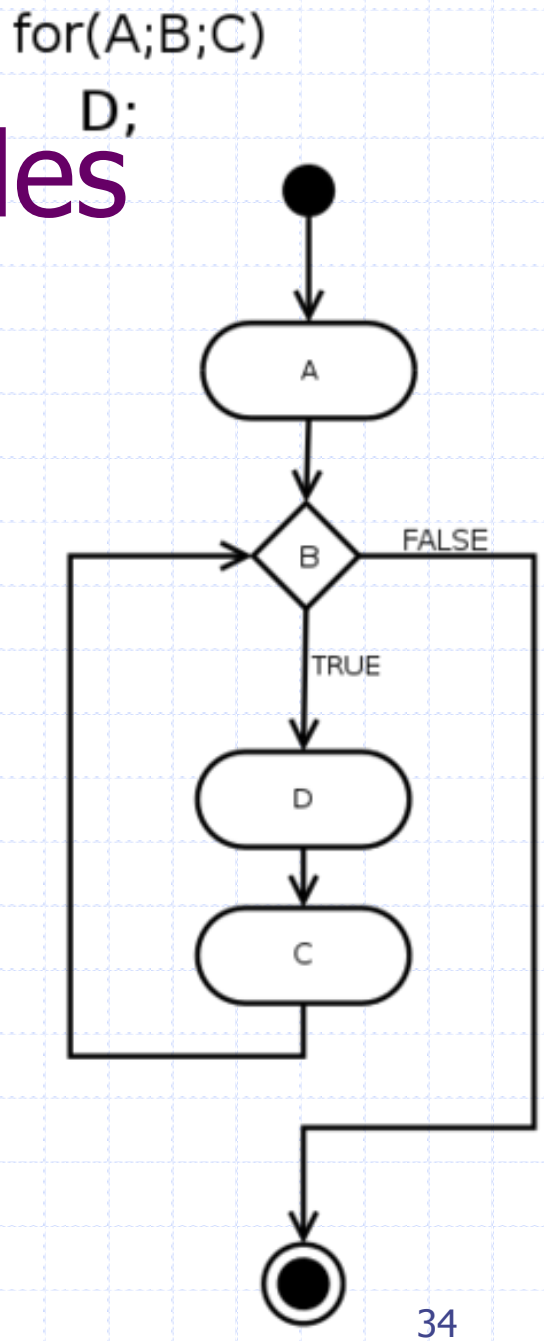
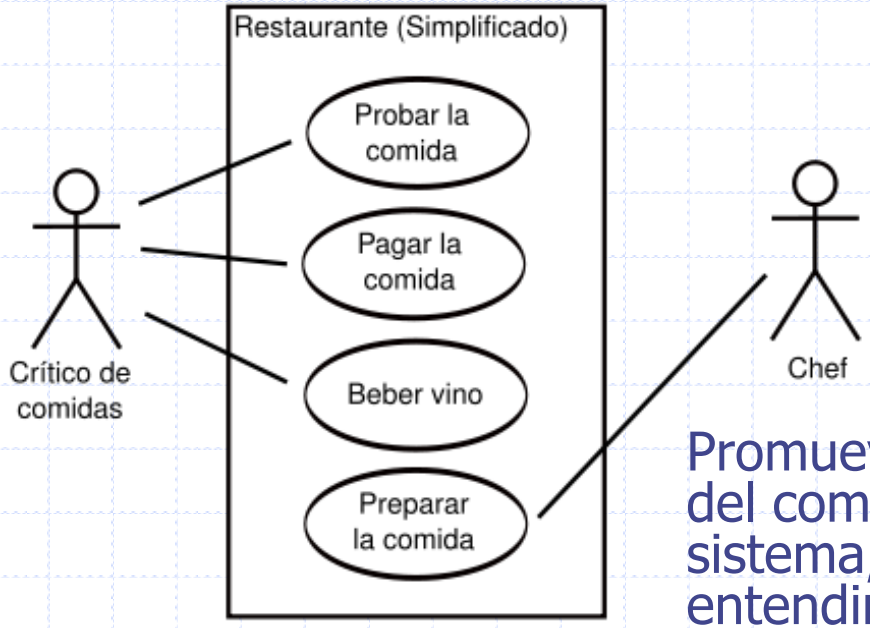


Diagrama de Caso de Uso

Describe el comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio.

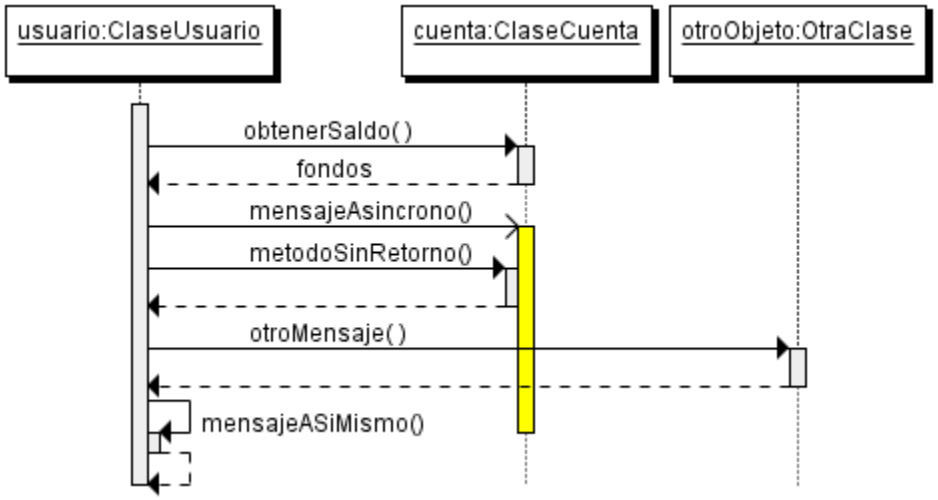


Promueve una imagen sencilla del comportamiento del sistema, que facilita un entendimiento común entre los actores involucrados en el desarrollo del Sistema (cliente/propietario/usuario y el equipo de desarrollo).

Diagrama de Estados

- Identifica cada una de las rutas o caminos que puede tomar un flujo de información luego de ejecutarse cada proceso.
- Permite identificar bajo qué argumentos se ejecuta cada uno de los procesos y en qué momento podrían tener una variación.
- El diagrama de estados permite visualizar de una forma secuencial la ejecución de cada uno de los procesos

Diagrama de secuencia



Tipo de diagrama usado para modelar interacción entre objetos en un sistema.

Diagrama de comunicación

- Modela las interacciones entre objetos o partes en términos de mensajes en secuencia. Los diagramas de comunicación representan una combinación de información tomada desde el diagrama de clases, secuencia, y diagrama de casos de uso describiendo tanto la estructura estática como el comportamiento dinámico de un sistema.

- Los diagramas de comunicación y de secuencia describen información similar, y con ciertas transformaciones, pueden ser transformados unos en otros sin dificultad.

- Para mantener el orden de los mensajes en un diagrama de comunicación, los mensajes son etiquetados con un número cronológico y colocados cerca del enlace por el cual se desplaza el mensaje. Leer un diagrama de comunicación conlleva comenzar en el mensaje 1.0, y seguir los mensajes desde un objeto hasta el siguiente, sucesivamente

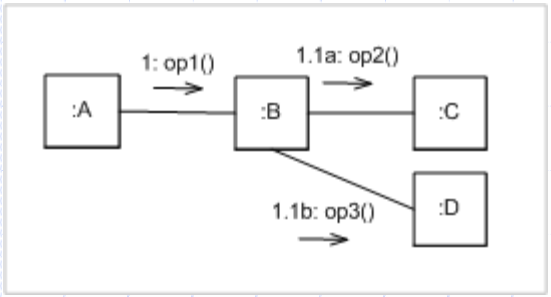


Diagrama de tiempos

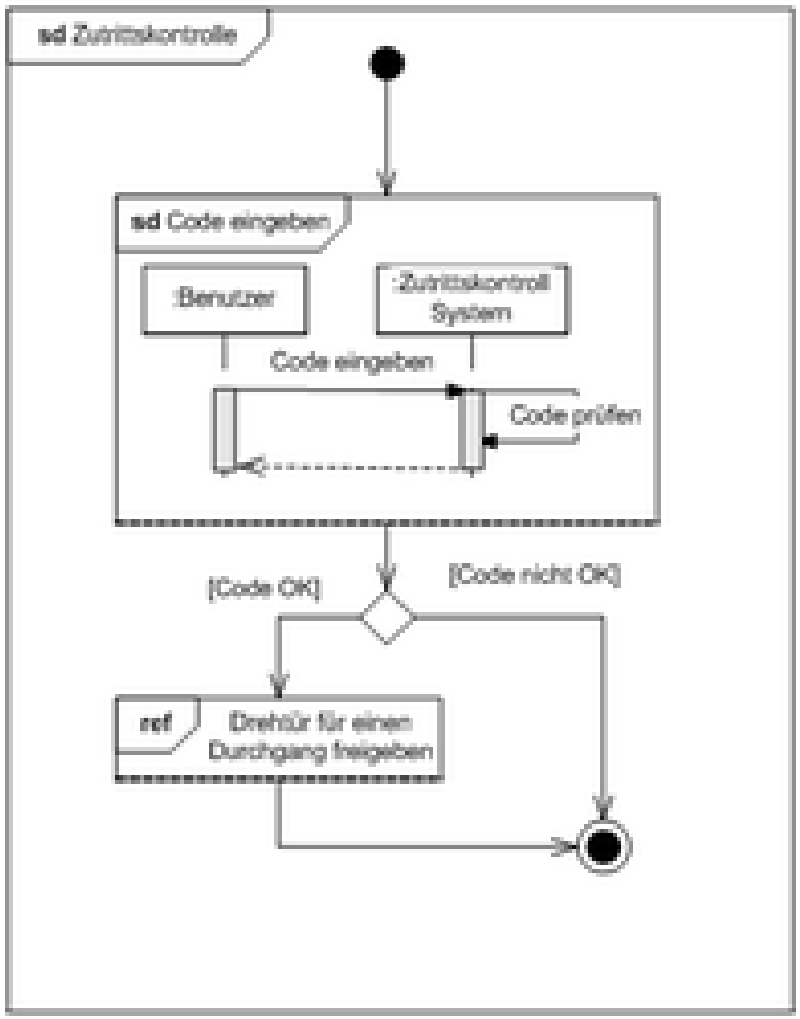
- Son una representación especial de interacción que se enfoca en el tiempo de los mensajes enviados entre objetos.
- Se pueden usar estos diagramas para mostrar restricciones detalladas sobre el tiempo, ó para mostrar los cambios con líneas de vida respecto al tiempo.
- Los diagramas de tiempo son generalmente utilizados con sistemas en tiempo real o en sistemas embebidos.

Diagrama global de interacciones

- Muestra una cierta vista sobre los aspectos dinámicos de los sistemas modelados. Aunque un diagrama global de las interacciones es una representación gráfica de una interacción, éste se distingue fuertemente de los diagramas de secuencia y de comunicación, dos de los otros diagramas de interacción. De hecho, algunos elementos gráficos del diagrama global de las interacciones están tomados del diagrama de actividades, otro diagrama de comportamiento para el modelado de actividades.

- Los modelos de interacción pueden llegar a ser muy grandes para sistemas complejos. Si el número de líneas de vida participantes y el número de mensajes intercambiados excede una cierta medida, se impone "modularizar" las interacciones y dividir en partes pequeñas, más manejables, de acuerdo a principios universales del diseño de sistemas, que también pueden ser visualizadas con la ayuda de un clásico diagrama de secuencias. La visión de conjunto de toda la interacción, de manera que la Big Picture o bien el cuadro global, puede entonces ser representada con la ayuda del diagrama global de las interacciones, provisto para eso.

Diagrama global de interacciones



- En este ejemplo, el diagrama global de interacciones combina un diagrama de secuencia, que está definido en el lugar (inglés inline), con una interacción (Drehtür für einen Durchgang freigeben o en español, Desbloquear la puerta giratoria para un libre paso), que está modelada en otra parte y que aquí está solo referenciada, reconocible en la palabra clave ref.
- El flujo de control entre estas dos interacciones es modelada con elementos de los diagramas de actividades. El proceso comienza en un nodo inicial y finaliza en un nodo terminal para actividades. Un nodo de ramificación entre las interacciones embebidas (el diagrama de secuencia y el de interacción) modela la decisión de si la entrada se abre o si debe permanecer cerrada