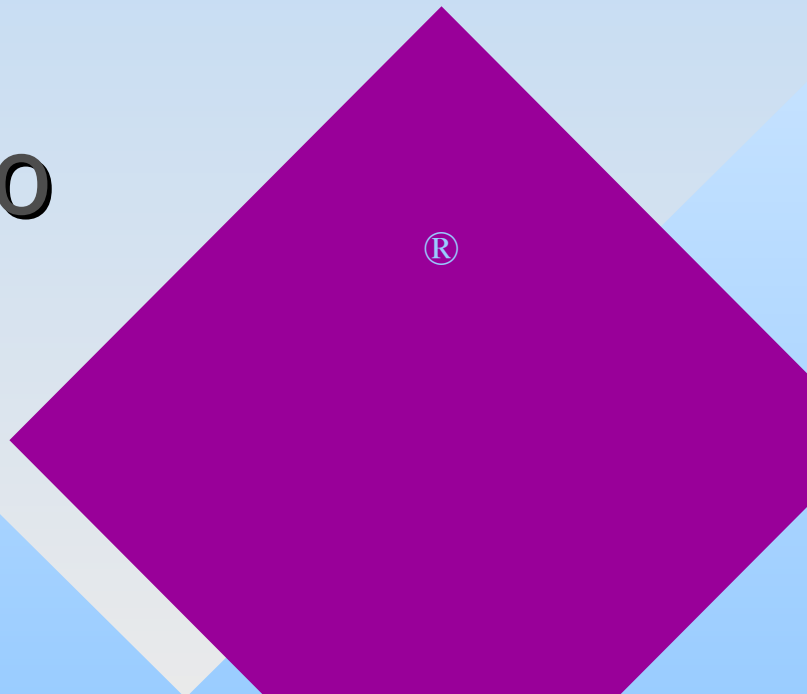
# UML

# Unified Modeling Language

Prof. Daniel Riesco



# Introducción

94 - Booch & Rumbaugh -> Rational

95 - UML . Primera Ver. 0.8

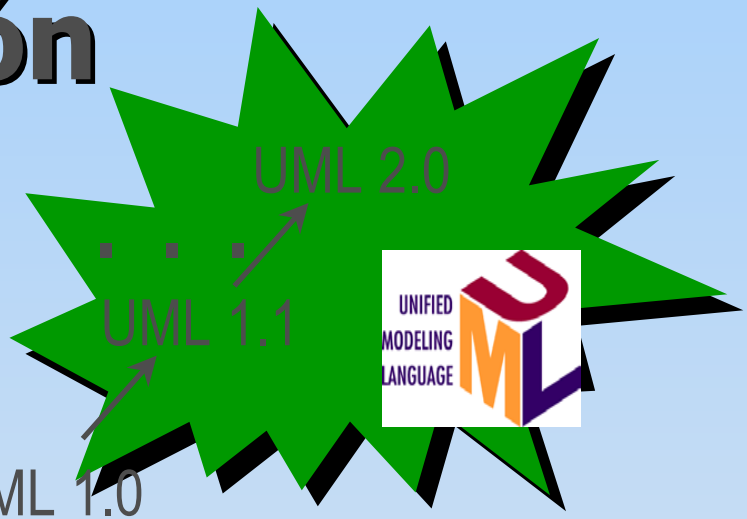
95 - Finales se une Jacobson (OOSE)

OMG - Object Management Group.

- Estándar
- Requisito básico: UML no sea de una empresa
- No existen los derechos de autor

# Origen y Evolución de UML

Aceptación en la OMG en Nov 1997




UML 1.0

UML 0.9

Unified Method 0.8

OTHER METHODS

Booch method

OMT

OOSE

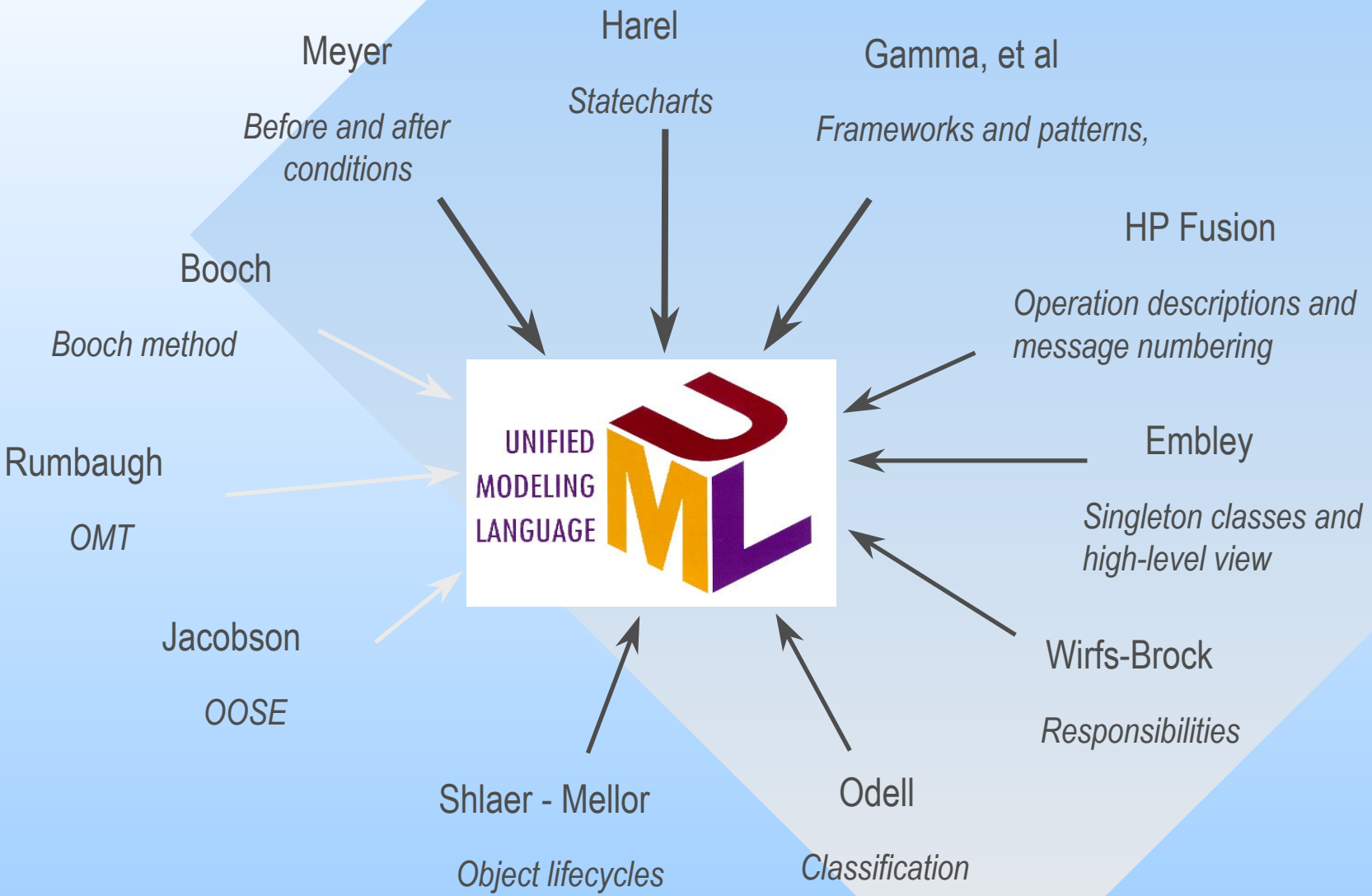
# UML Partners

- Rational Software Corporation
- Hewlett-Packard
- I-Logix
- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Microsoft
- ObjecTime
- Oracle
- Platinum Technology
- Taskon
- Texas Instruments/Sterling Software
- Unisys

# Metas

- Proveer a los usuarios un lenguaje de modelado visual expresivo y fácil de usar
- Proveer mecanismos de extensibilidad y especialización (ampliación del núcleo)
- Ser independiente de un lenguaje y un proceso
- Proveer una base formal para entendimiento del lenguaje de modelado
- Estimular el crecimiento de herramientas OO

# Contribuciones a UML



# UML: Definiciones

- UML es un lenguaje para **especificar, construir, visualizar y documentar** artefactos de un sistema de software orientado a objetos.
  - Artefacto es información que es utilizada o producida mediante un PDS.
- Un metamodelo es un modelo que define el lenguaje para expresar otros modelos.
- Un modelo es una abstracción que se elabora para comprender algo antes de construirlo.
- Un diagrama es una representación gráfica de una colección de elementos del modelo (grafo).

# Importancia del Modelado

- Un modelo es una simplificación de la realidad.
- Construimos modelos de forma de comprender mejor el sistema que estamos desarrollando.
- Construimos modelos de sistemas complejos porque no podemos comprender tales sistemas en su totalidad.

Los modelos pueden ser informales o formales.

# Principios de Modelado

- La elección de qué modelos crear tiene una influencia profunda sobre cómo un problema es atacado y cómo una solución es alcanzada.
- Todo modelo puede ser expresado en diferentes niveles de precisión.
- Un modelo simple no es suficiente. Todo sistema no trivial es mejor aproximado a través de un conjunto de modelos cercanamente independientes.

# UML

*UML es un lenguaje estándar para el modelado de sistemas de software*

- Sirve como puente entre la especificación de requerimientos y la implementación.
- Provee un significado para especificar y documentar diseños de un sistema de software.
- Está particularmente preparado para desarrollo de programas orientados a objetos.

# Architecting a dog house

Can be built by one person
Requires
- Minimal modeling
- Simple process
- Simple tools

RATIONAL SOFTWARE

# Architecting a house

Built most efficiently and timely by a team

Requires

- Modeling
- Well-defined process
- Power tools

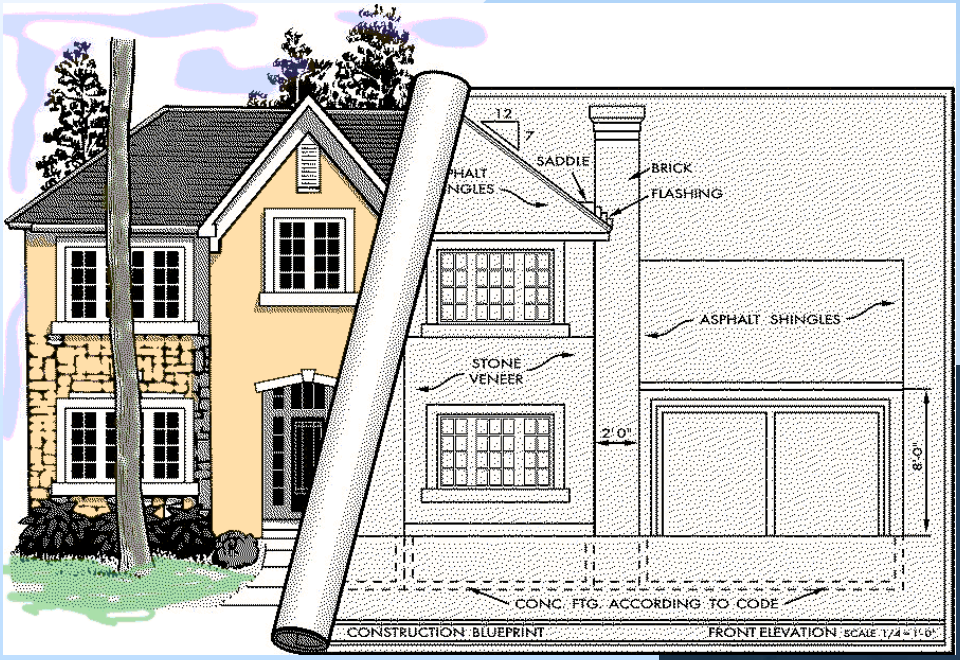
# Architecting a high rise

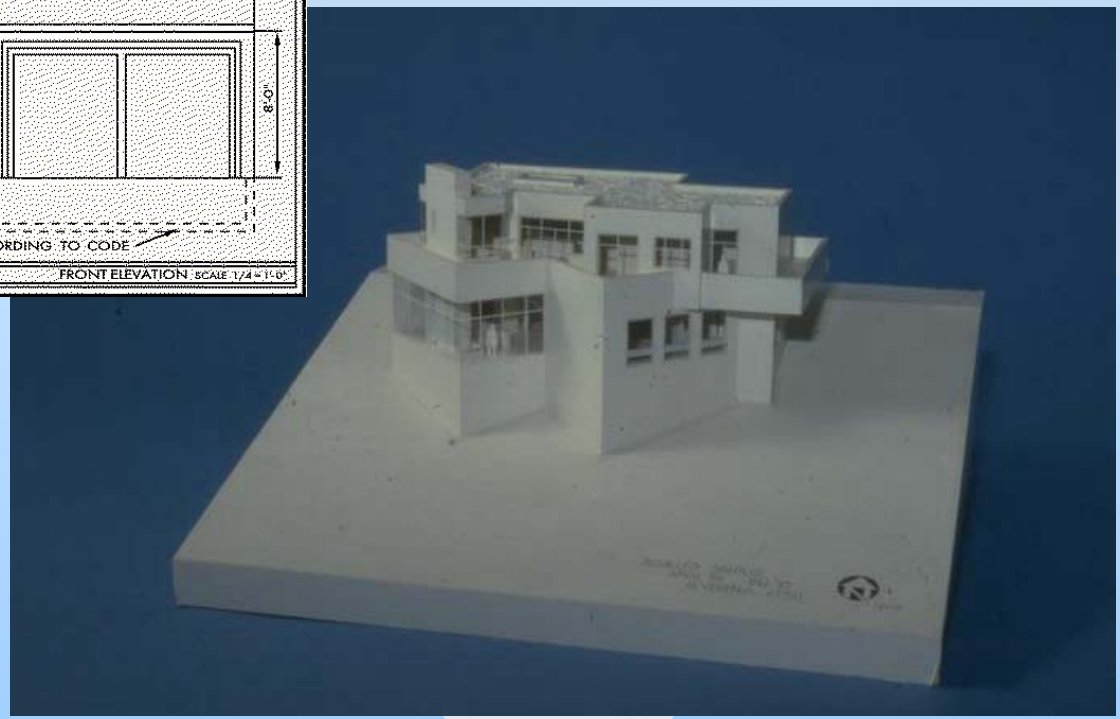
# Early architecture

**Progress**

- Limited knowledge of theory

# Modeling a house



RATIONAL SOFTWARE

# Abstracción

- Diferentes vistas del sistema.
- Una misma vista pero diferentes niveles de abstracción. Dominio del Problema, Usuario final, Analista, Programador.
- Diagrama con distinto detalle de un mismo modelo.
  - Ocultar: bloques de construcción, relaciones, adornos, flujos, estereotipos.
- Distintos modelos (uno más abstracto que el otro).
  - Casos de uso y su realización (Colaboración)
  - Colaboraciones y su realización (Clases)
  - Componentes y su diseño (modelo de diseño)
  - Nodos y sus componentes (modelo de implementación)

# Desarrollo de Diagramas

- Propósito no es dibujar sino visualizar, especificar, construir y documentar.
- No hacer diagramas mínimos (excepto para presentaciones)
- Equilibrio entre diagramas de comportamiento y estructurales.
- Un diagrama debe centrarse en comunicar un aspecto de la vista, con los elementos importantes para ello, consistente con su nivel de abstracción.

# ¿Para que modelamos?

- Clasificar y entender información
  - Organizar, encontrar, filtrar, recuperar, examinar y editar información
- Explorar Soluciones Alternativas
  - Construir y evaluar diferentes modelos
  - Determinar el "mejor" modelo basado en
    - Análisis cuantitativo: Simulación, Complejidad Temporal.
    - Exámen cualitativo de aspectos/capacidades
  - Económicamente factible

# Niveles de Modelado

- Alto nivel en etapas tempranas
  - Destinado a Stakeholders no técnicos
  - Exploración Conceptual del Problema
  - Refinamiento vía modelos medios detallados
- Modelos de niveles medios
  - Especificación de Capacidades escenciales del sistema
  - Historicamente: ERs, DFDs, FSMs, etc.
  - Recientemente: Escenarios, Patrones de Diseño, etc.
- Modelos Detallados
- Modelos Formales
- ¿Cuál es el rol de UML?

# ¿Qué define un Modelo?

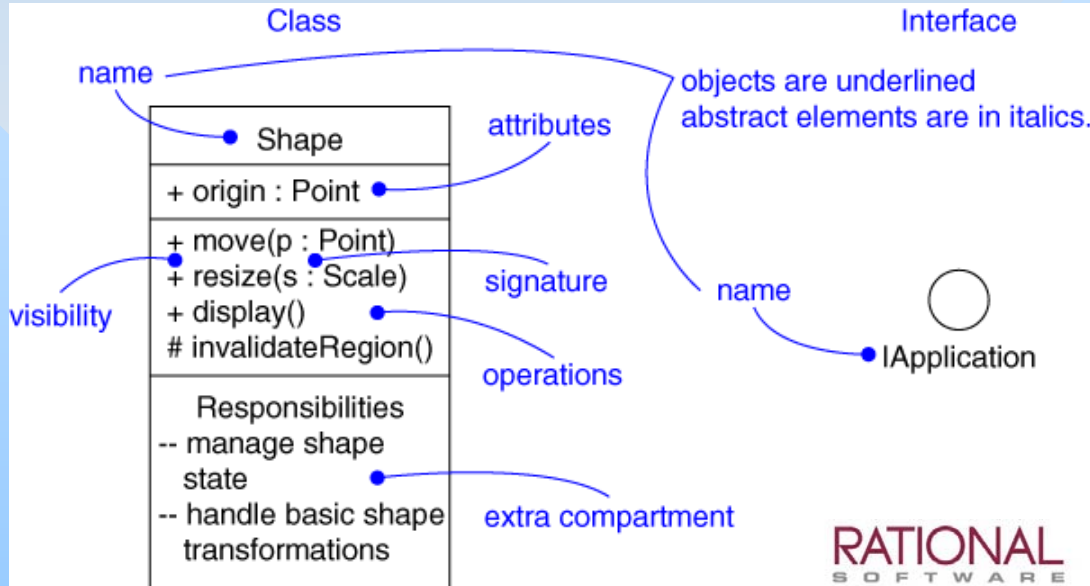
- Lenguajes definidos por
  - Sintaxis: Construcciones y contexto sintáctico
  - Semantica: Significado de las distintas construcciones
  - Pragmatica: Semántica operacional del sistema
- En Lenguajes de Progrmación:
  - Sintaxis: Análisis léxico y parsing
  - Semantica: Gramatica de Atributos/Traducción
  - Pragmatica: Ambiente Runtime Dinámica
- ¿Cómo son definidos los modelos?
  - Semantica
  - Presentación Visual
  - Nota: Pueden tener Sintaxis y Pragmática!

# Generalidades de UML

- Elementos de Modelado
- Relaciones
- Mecanismos de Extensión
- Diagramas

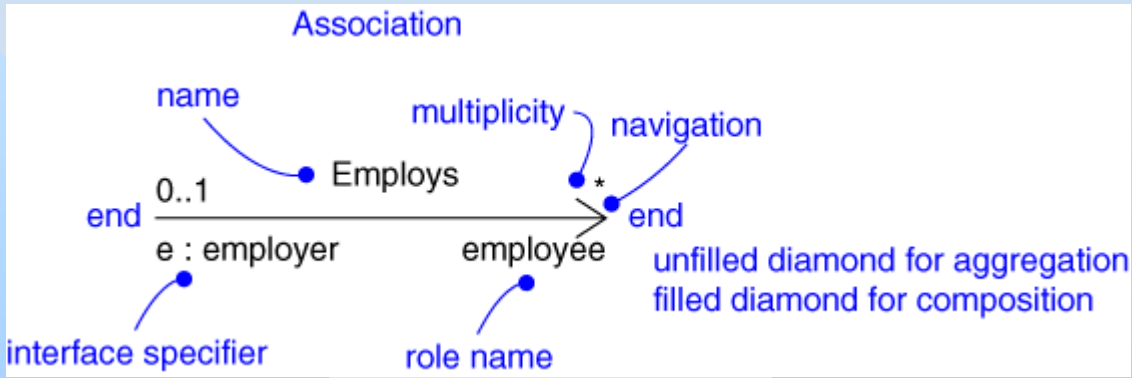
# Elementos del Modelo

- Elementos estructurales
  - clases, interfaces, colaboraciones, casos de uso, clases activas, componentes, nodos
- Elementos de Comportamiento
  - interacción, máquinas de estado
- Elementos de Agrupación
  - Paquetes, subsistemas
- Otros elementos
  - notas



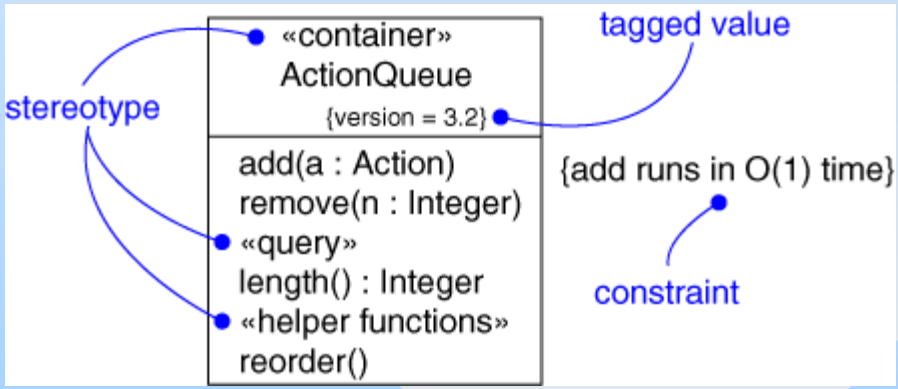
# Relaciones

- Dependencia
- Asociación
- Generalización
- Realización



# Mecanismos de Extensión

- Estereotipos
- Valores rotulados
- Restricciones

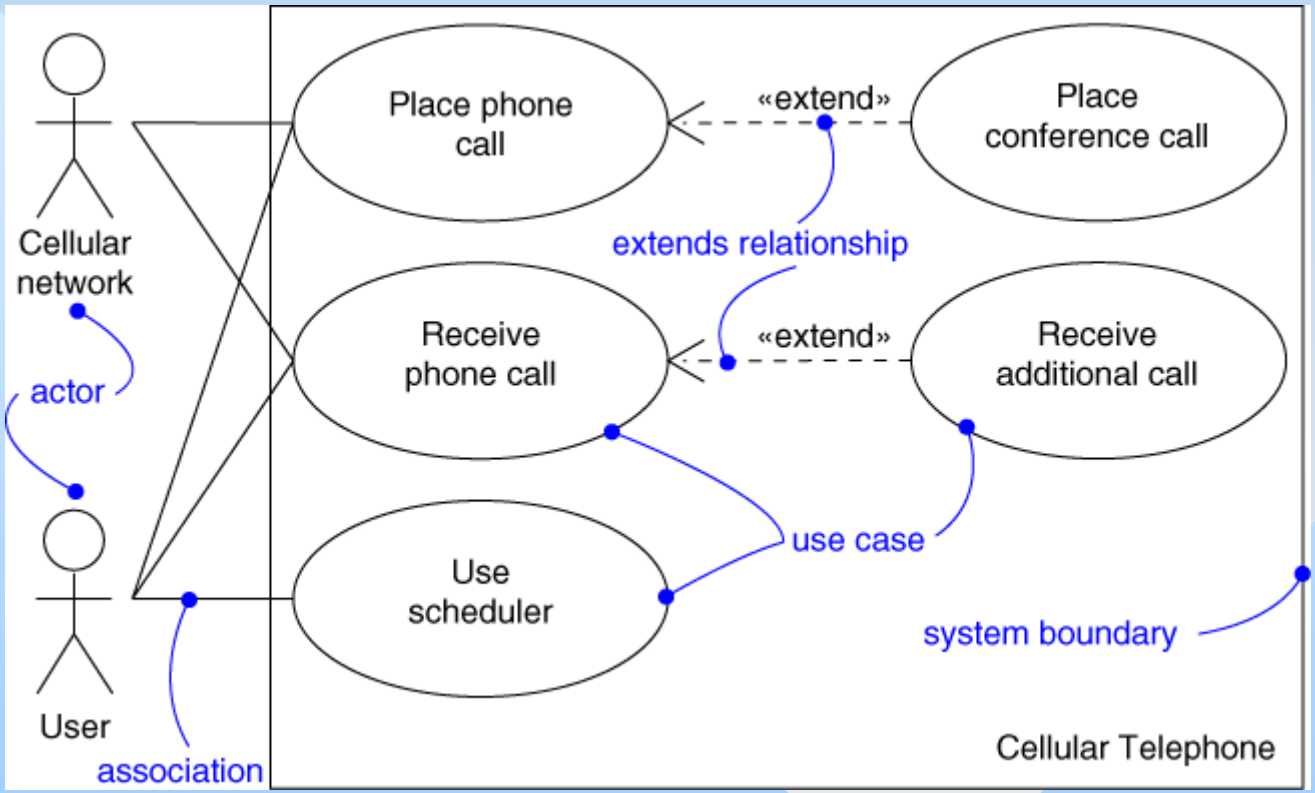


# Diagramas

- Un diagrama es una vista dentro de un modelo
  - Presenta aspectos de un stakeholder particular
  - Provee una representación parcial del sistema
  - Es semánticamente consistente con otras vistas
- En UML, hay nueve diagramas estándar
  - Vistas estáticas: caso de uso, clase, objeto, componente, despliegue
  - Vistas dinámicas: secuencia, colaboración, de estado, actividad

# Diagrama de Caso de Uso

- Captura la funcionalidad del sistema vista por los usuarios

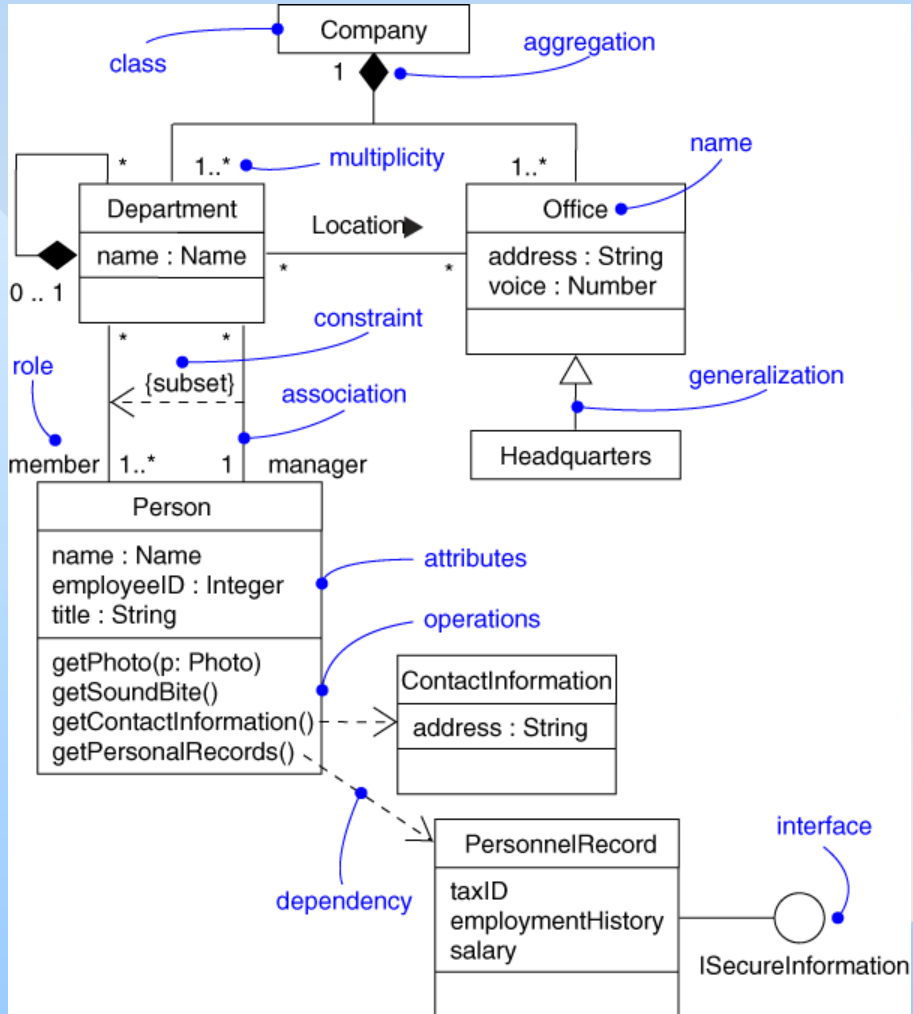


# Diagrama de Caso de Uso

- Se construye en las primeras etapas del desarrollo de un sistema
- Objetivo
  - Especificar el contexto del sistema
  - Capturar los requerimientos de un sistema
  - Validar la arquitectura del sistema
  - Dirigir la implementación y generar los casos de test
- Desarrollado por analistas y expertos del dominio

# Diagrama de Clase

- Captura el vocabulario del sistema (modelo de analisis)

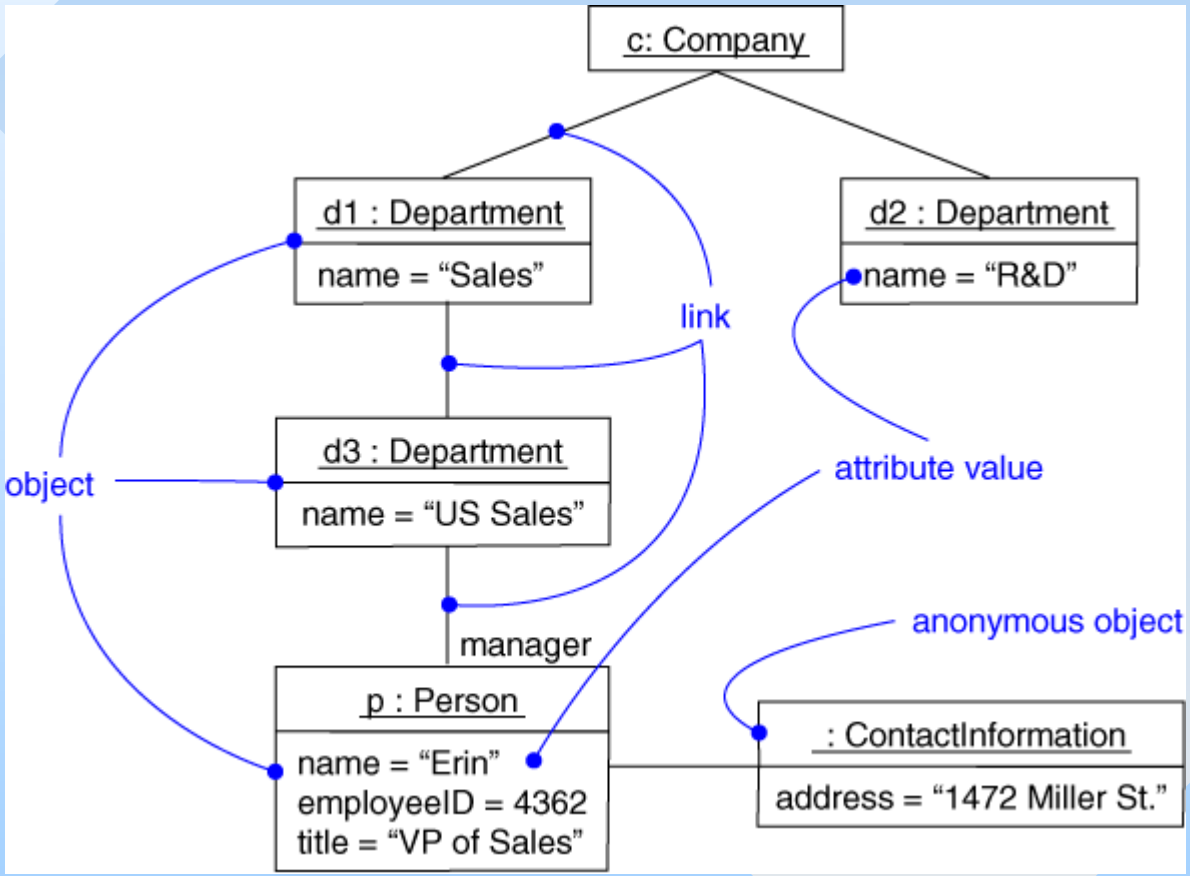


# Diagrama de Clase

- Se contruye y refina a través del desarrollo del sistema.
- Objetivo
  - Nombrar y modelar conceptos en el sistema
  - Especificar colaboraciones
  - Especificar esquemas lógicos de bases de datos
- Desarrollado por analistas, diseñadores y programadores.

# Diagrama de Objectos

- Captura instancias y enlaces

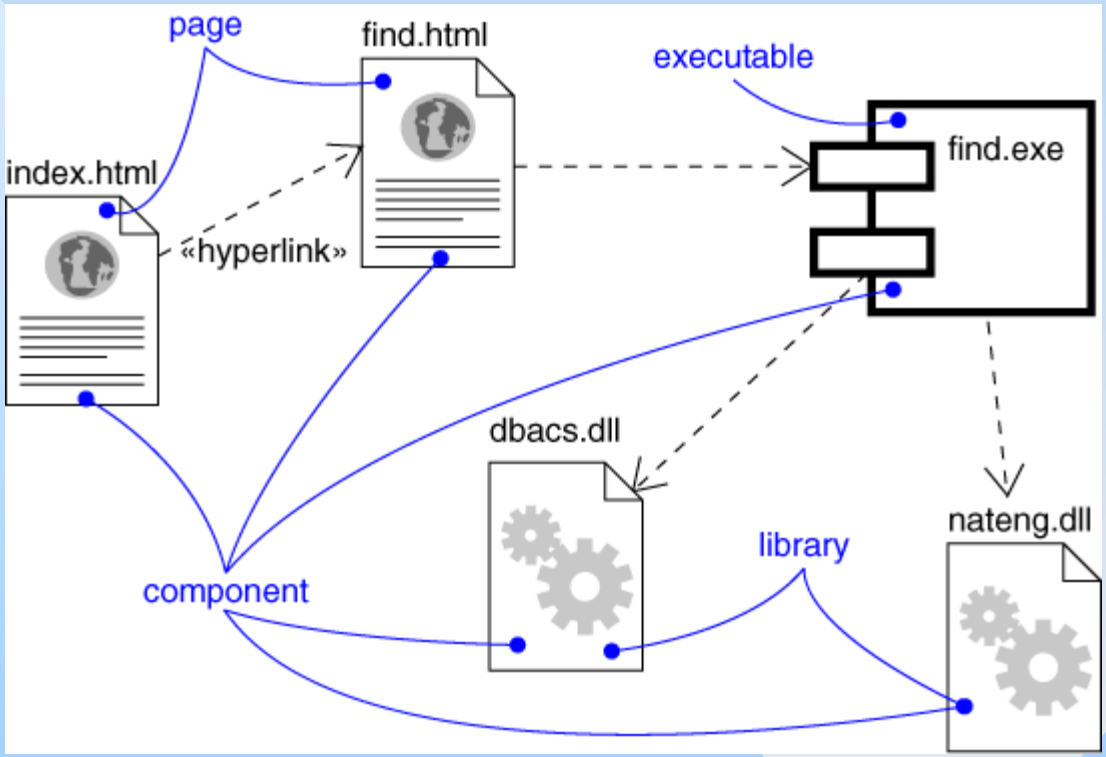


# Diagrama de Objetos

- Se construye durante el análisis y diseño
- Objetivo
  - Ilustrar estructuras de datos/objetos
  - Especificar una instantánea del sistema
- Desarrollado por analistas, diseñadores y programadores

# Diagrama de Componentes

- Captura la estructura física de la implementación

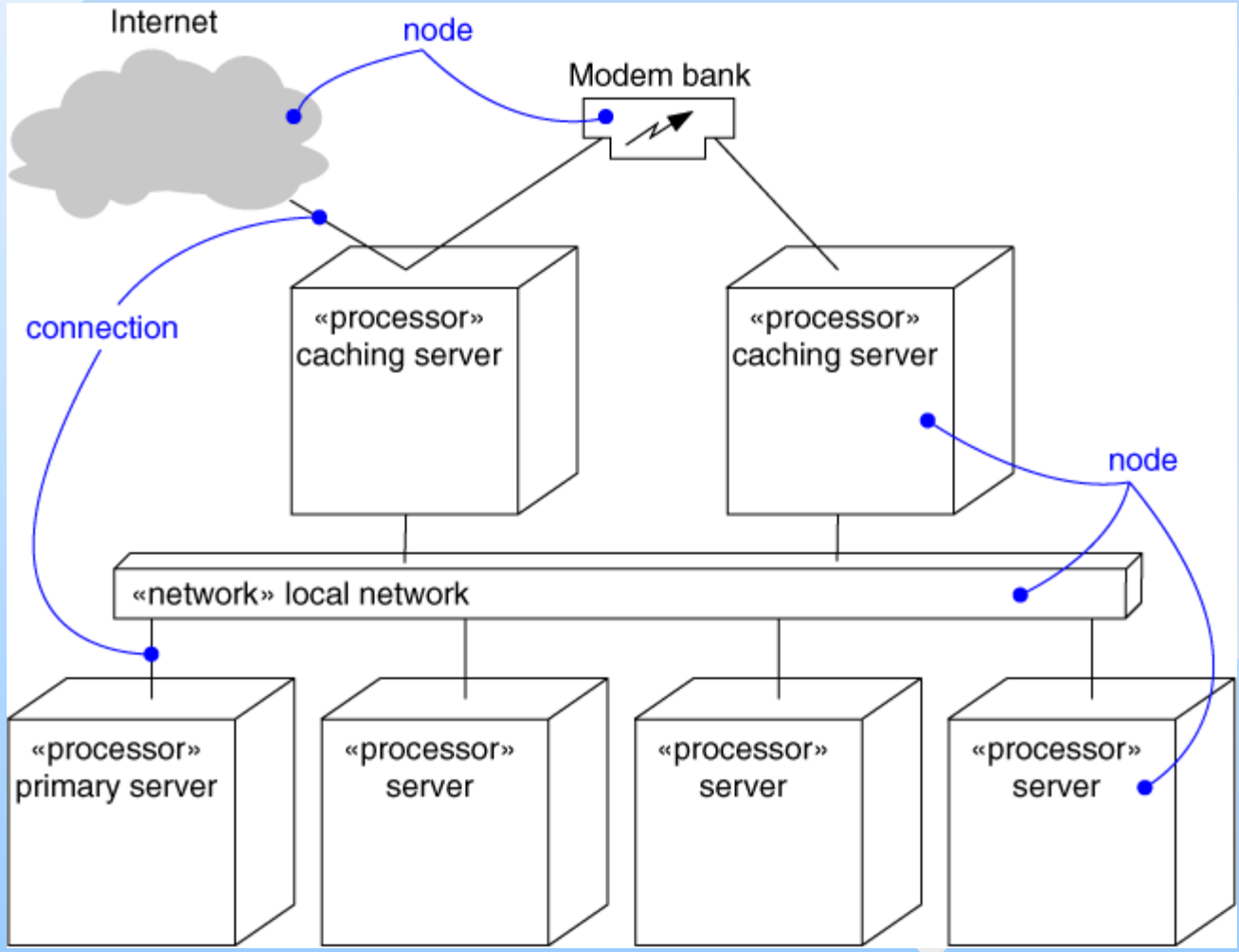


# Diagrama de Componentes

- Se construye como parte de la especificación arquitectónica
- Objetivos
  - Organizar el código fuente
  - Construir un release ejecutable
  - Especificar una base de datos fisica
- Desarrollado por arquitectos y programadores

# Diagrama de Despliegue

- Captura la topología del hardware del sistema

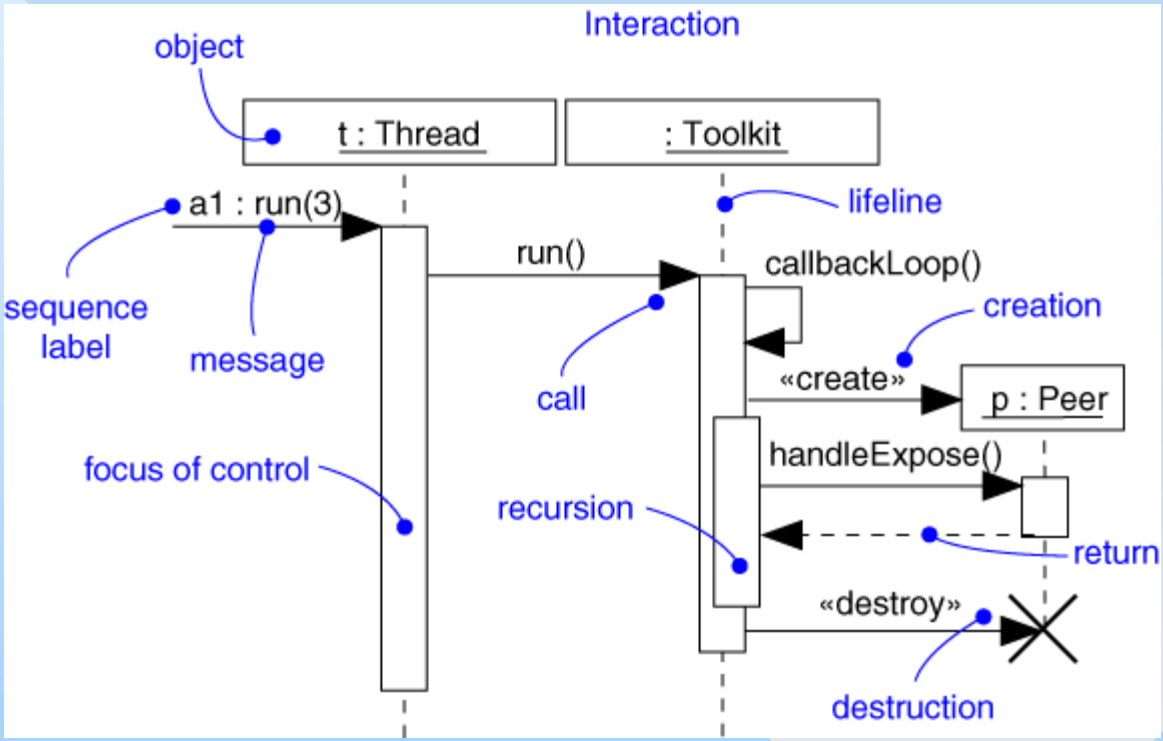


# Diagrama de Despliegue

- Se construye como parte de la especificación arquitectónica
- Objetivos
  - Especificar la distribución de componentes
  - Identificar cuellos de botellas respecto de la performance
- Desarrollado por arquitectos, ingenieros en redes y ingenieros en sistemas

# Diagrama de Secuencia

- Captura el comportamiento dinámico (muestra secuencia en el tiempo)

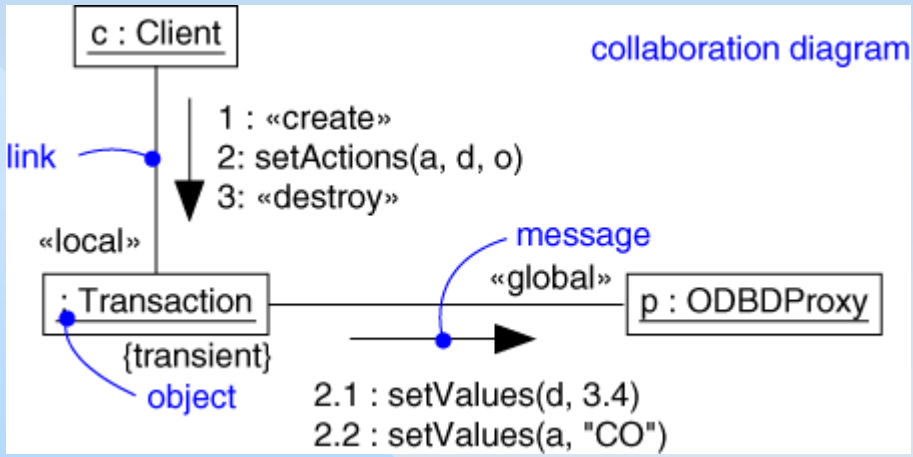


# Diagrama de Secuencia

- Objetivos
  - Modelar el flujo de control
  - Ilustrar escenarios típicos

# Diagrama de Colaboración

- Captura el comportamiento dinámico (orientado a mensajes)

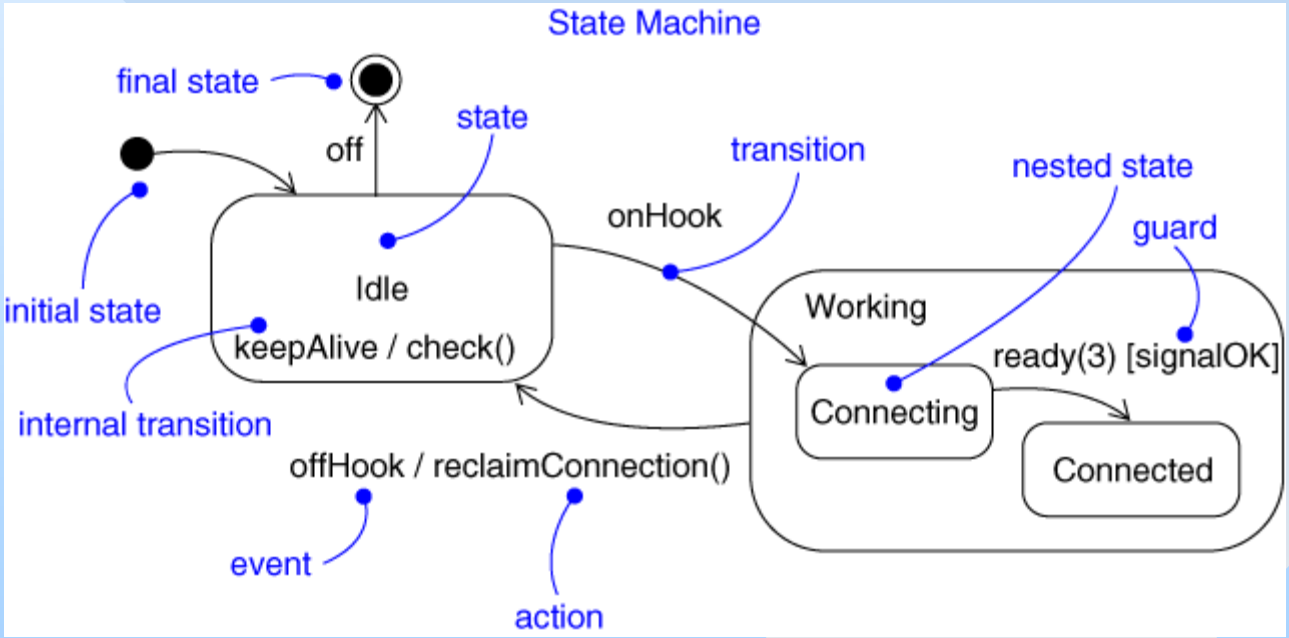


# Diagrama de Colaboración

- Objetivos
  - Modelar el flujo de control
  - Ilustrar la coordinación de control y estructura de objetos

# Diagrama de Estado

- Captura el comportamiento dinámico (orientado a eventos)

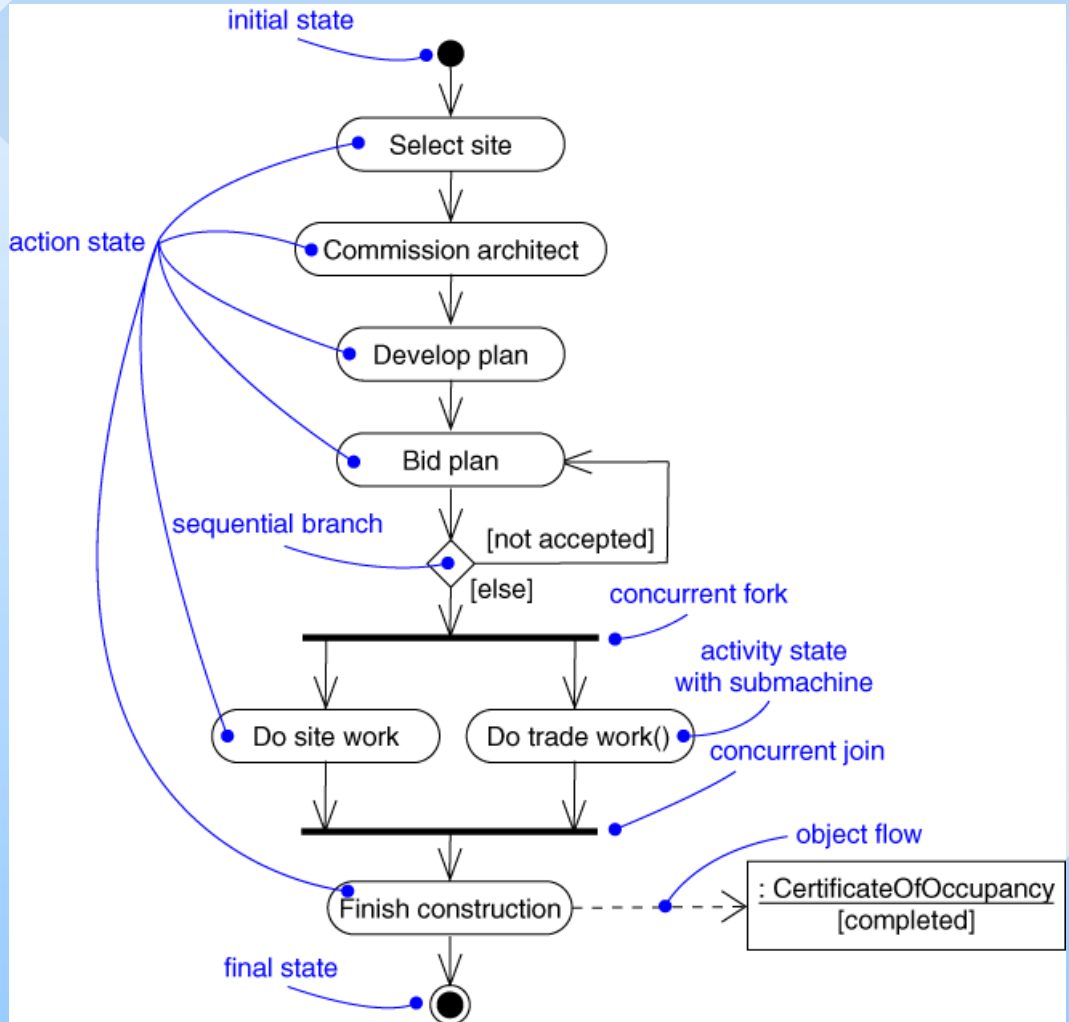


# Diagrama de Estado

- Objetivo
  - Modelar el ciclo de vida de un objeto
  - Modelar objetos reactivos (interfaces de usuario, dispositivos, etc.)

# Diagrama de Actividad

- Captura el comportamiento dinámico (orientado a la actividad)



# Diagrama de Actividad

- Objetivo
  - Modelar el flujo de trabajo del negocio (workflows)
  - Modelar operaciones

# Modelos, Vistas y Diagramas

A ***model*** is a complete description of a system from a particular perspective