UNIVERSIDADE DA CORUÑA

fic
facultade de
informática
da coruña

TEMA 4:
MODELO DE DISEÑO en CommonKADS

CONTENIDO

1. Principios de diseño y criterios de calidad
2. Arquitectura de sistemas CommonKADS
3. Modelo de Diseño en cuatro pasos
4. Ejemplos

Del análisis al diseño

DOMINIO DE APLICACIÓN

Libros

Estrategias de razonamiento

Expertos

Manuales

Casos

Protocolos

Tiempos de respuesta

SISTEMA SOFTWARE

Arquitectura software

Diseño de algoritmos

Modelo de diseño

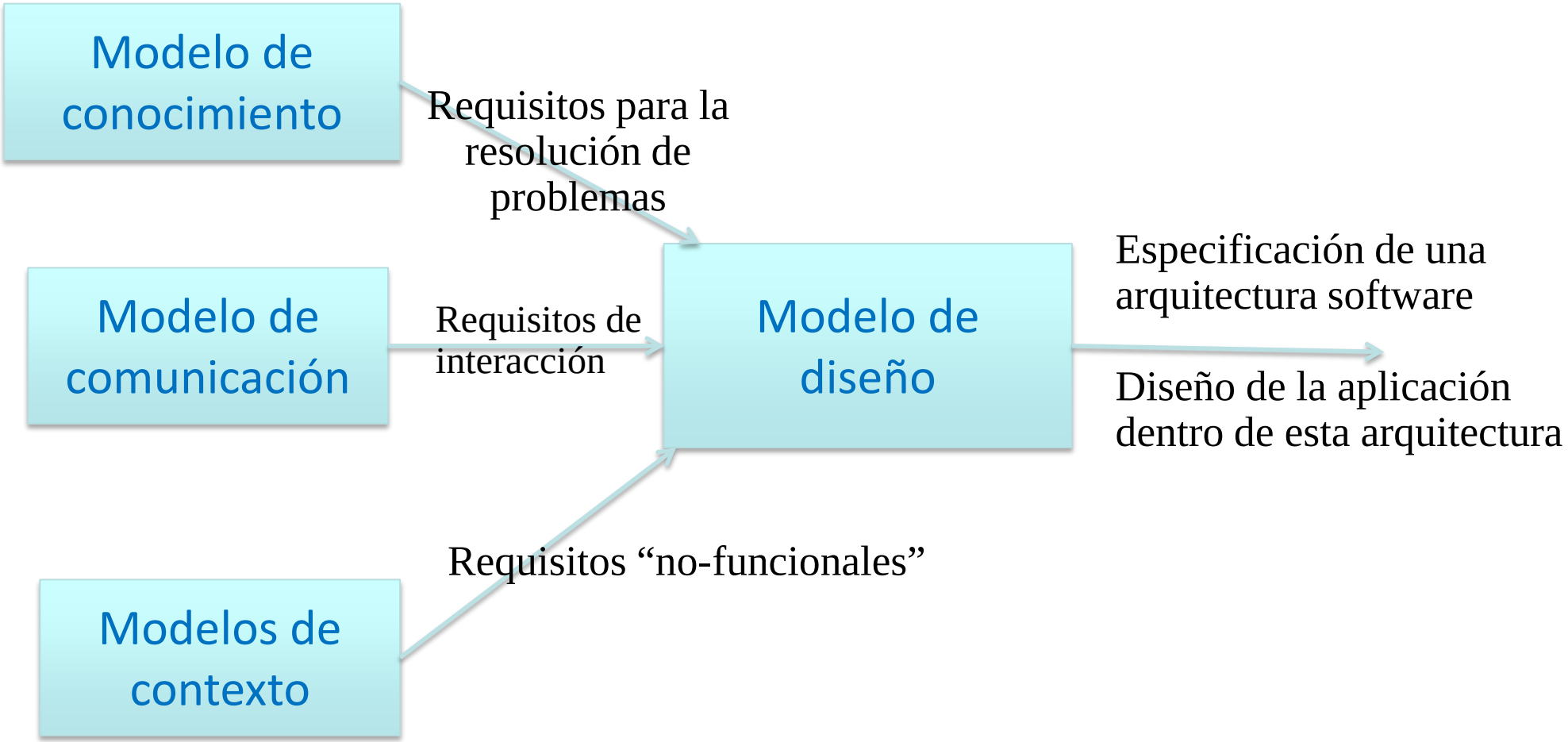
plataforma hardware

Lenguaje de implementación

Diseño de estructuras de datos

Diseño de datos
Diseño arquitectura
Diseño interfaz
Diseño de los componentes

Entradas y salidas del Modelo de diseño



Arquitectura del sistema

- Descripción del software en términos de:
  - descomposición en sub-sistemas
  - selección del régimen(es) de control
  - descomposición de los sub-sistemas en módulos software
- Punto central del proceso de diseño
- Arquitectura de referencia para sistemas basados en CommonKADS.

Diseño manteniendo la estructura

- Principio central del diseño moderno

  *"Mantener el **contenido** y la **estructura** del modelo de análisis durante el diseño"*

- Diseñar es "anadir detalles específicos de implementación a los resultados del análisis"
- Mantener la información es la noción clave
- Directamente relacionado con los criterios de calidad

Criterios de calidad de diseño de software

Diseño de software

Minimización de acoplamiento
Maximización de cohesión
Transparencia
Mantenimiento

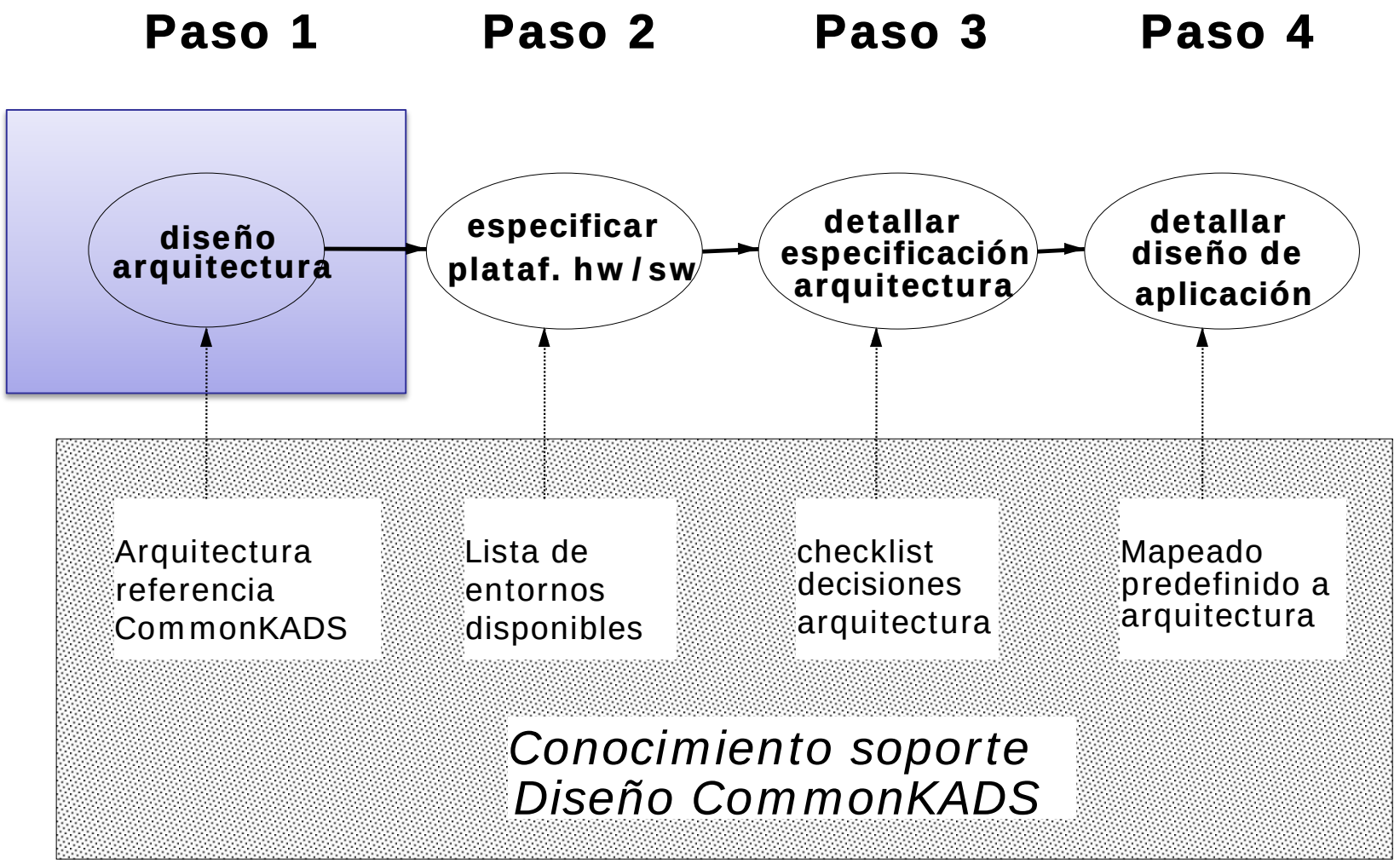
Diseño de SBCs

Reusabilidad de elementos de diseño/ código
Mantenimiento y adaptabilidad
Potencia explicativa
Adquisición de conocimiento/facilidad para refinado

Objetivos del modelo de diseño

1. Separación del análisis y la implementación.
2. Garantizar la calidad
3. Especificación independiente de la plataforma
4. Descomposición de la tarea de diseño
5. Separación de arquitectura y contenidos
6. Inclusión de los requisitos del entorno
7. Reutilización
8. Diseño de la interfaz y la interacción

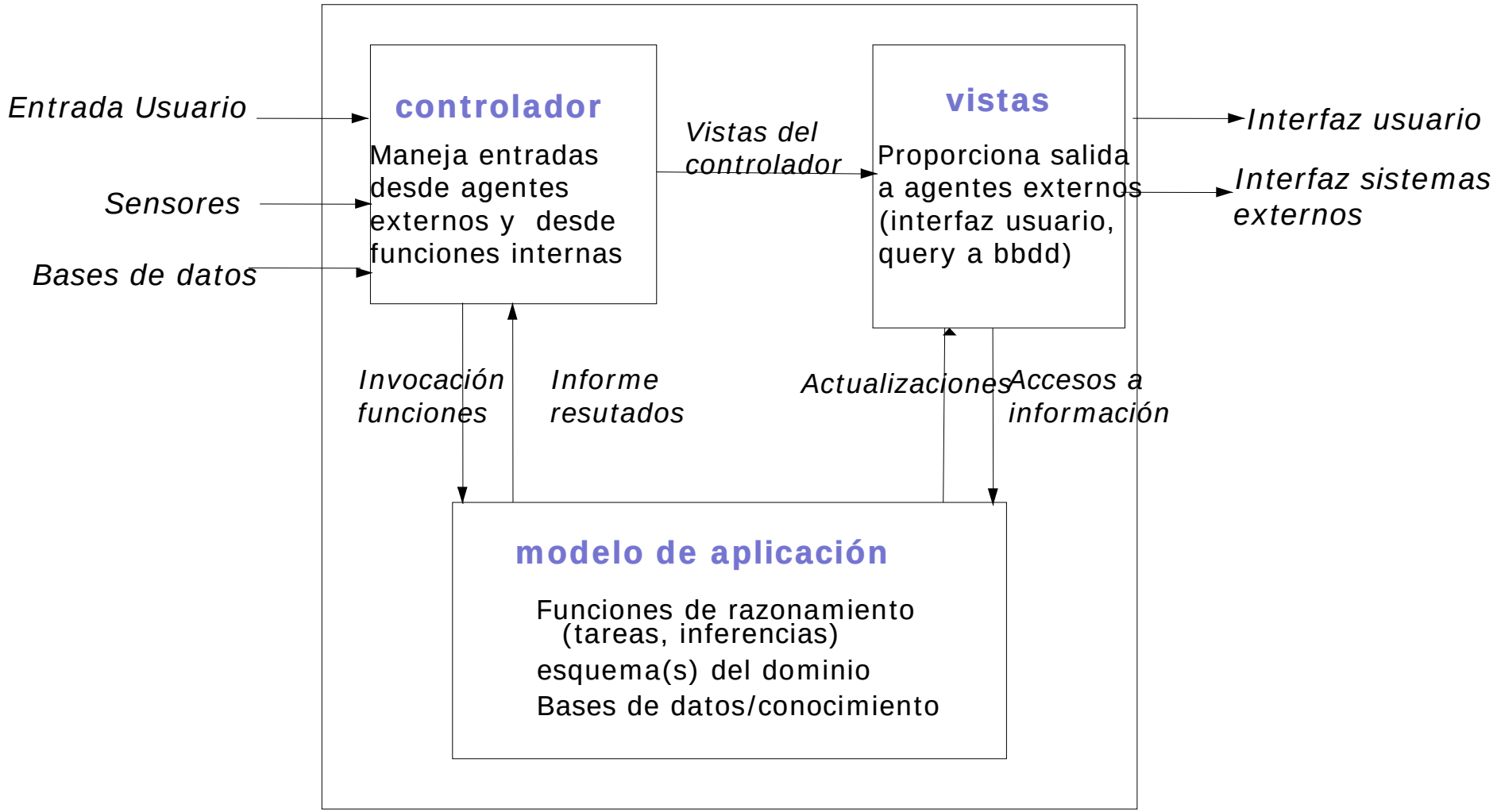
Pasos en el diseño del sistema



Paso 1: Especificar arquitectura global

- Principio: Separar funcionalidad de interfaz
  - descomposición del sistema en subsistemas
  - régimen de control global
  - descomposición de subsistemas en módulos software
- Arquitectura global:
  - Modelo MVC (Model-View-Controller)
  - distingue entre los objetos de una aplicación y su visualización
  - unidad de control central con régimen dirigido por eventos

Arquitectura del sistema: Tres subsistemas principales



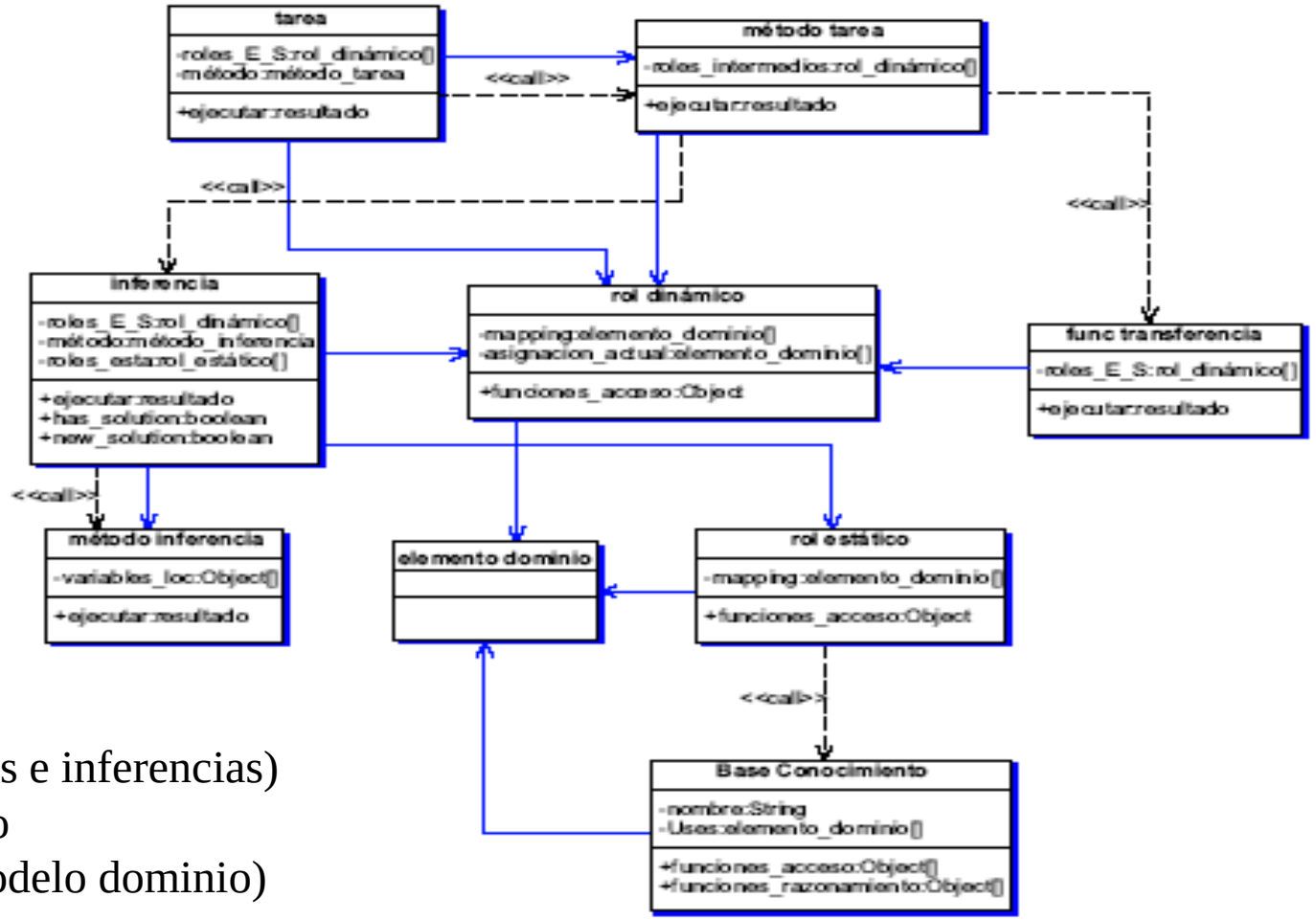
Subsistema 1: Modelo de aplicación

- Contiene los datos y las funciones de la aplicación =objetos del modelo de conocimiento

- Datos:
  - bases de conocimiento
  - datos dinámicos manipulados durante el razonamiento (roles dinámicos)

- Funciones
  - tareas, inferencias, funciones de transferencia

Arquitectura de referencia: Subsistema del modelo de la aplicación



Contiene:

- F. raz (tareas e inferencias)
- Estruct. info

y conoc. (modelo dominio)

Subsistema 2: Vistas

- Visualizaciones de datos y funciones de la aplicación

- Posibilitan visualización de información estática y dinámica a agentes externos, como usuarios u otro software.
  - son posibles las visualizaciones múltiples
  - agregar visualización de múltiples objetos de la aplicación
  - requiere actualización arquitectural /mecanismos de integridad
    - tabla de mapeos
    - protocolos de mensajes para cambios en el estado de los objetos

Subsistema 3: Controlador

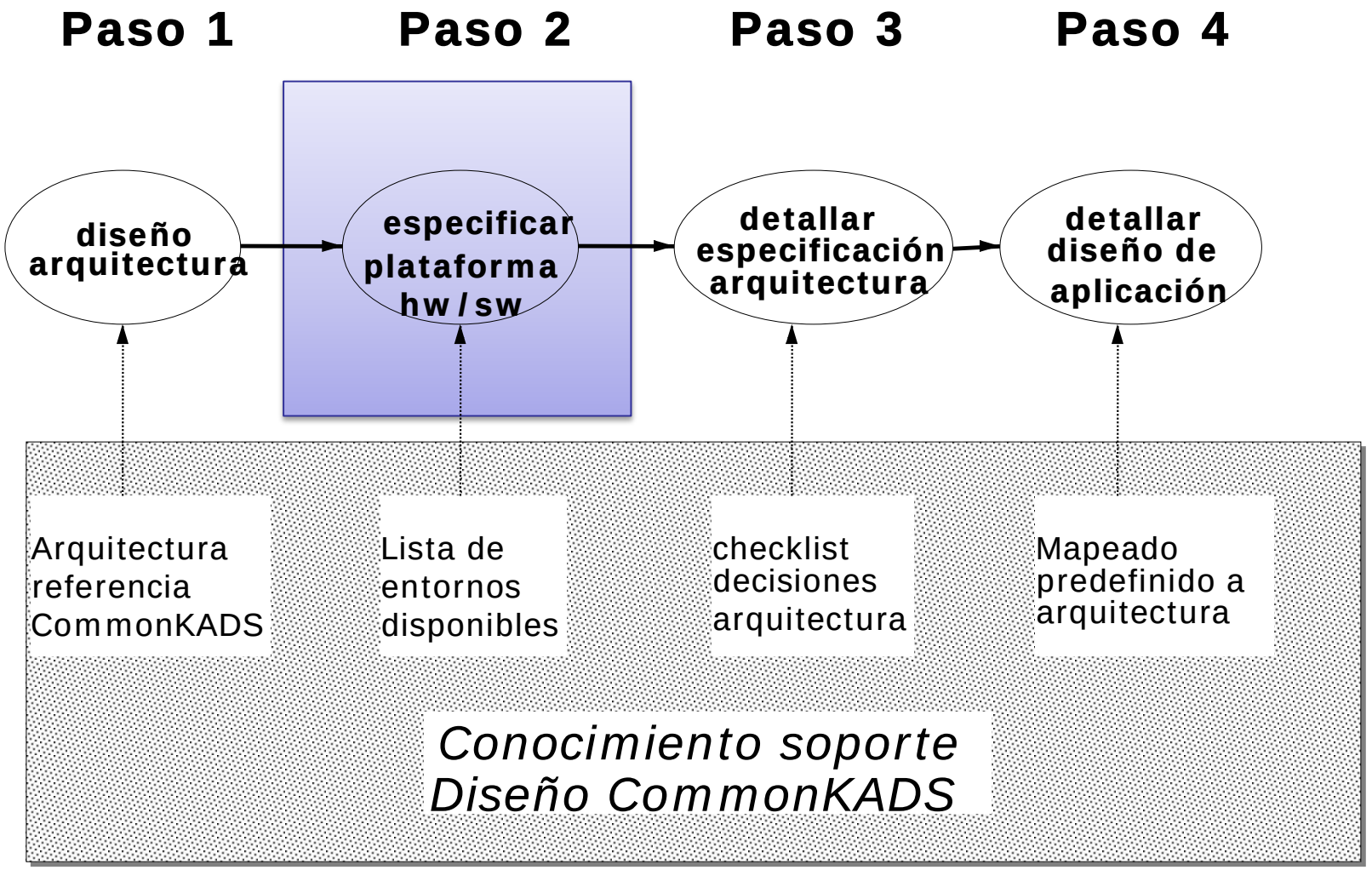
- Unidad central de control y comandos
- Suele ser dirigido por los eventos
- Proporciona handlers para eventos internos y externos
- Permite la activación de las funciones de la aplicación y decide qué hacer cuando llegan los resultados
- Puede definir sus propias vistas de “control” para proporcionar información sobre el proceso, p.ej. al usuario
- Puede tener reloj interno y agenda

  => comportamiento tipo demonio

Formulario DM-1 (Ejemplo)

MODELO DE DISEÑO Decisión arquitectura	DM1 Arquitectura
ESTRUCTURA SUBSISTEMAS	Variación modelo MVC
MODELO DE CONTROL	Control centralizado con “handler” que maneja eventos recibidos, y posibilidad de reloj interno y agenda
DESCOMPOSICIÓN SUBSISTEMAS	El subsistema aplicación de divide en los módulos de la figura siguiendo principios de OO

Pasos en el diseño del sistema



Paso 2: Identificar la plataforma de implementación

- Teóricamente el diseño se puede realizar de forma independiente de la plataforma de implementación.

- Los requisitos específicos del cliente suelen restringir la selección de la plataforma

  = razón para colocarlo en un paso temprano en el proceso

- Elección del software mucho más importante que la del hardware

  a veces no, en el caso de aplicaciones en tiempo real

- Si elección libre, considerar postponer hasta acabar el paso 3

Criterios plataforma (1)

- Librería de clases de objetos "vista" (librería de vistas de objetos)

  puede ser necesario construir muchos uno mismo

- Paradigma de programación

  Si el diseño es O-O, es deseable que el lenguaje de la herramienta lo sea

- Formalismo de representación de conocimiento declarativo

  A veces es un problema en plataformas O- O que no disponen de formalismos de otro tipo, como reglas.

- Interfaces estándar a otro software

  ej. ODBC, CORBA, J2EE, .NET, necesarias a menudo

Criterios plataforma (2)

- Facilidades para la escritura del lenguaje
  - un “tecleado” débil normalmente implica más trabajo en el mapeado del modelo de análisis (ver paso 4a)
- Facilidades de control/protocolos
  - soporte de paso de mensajes
  - posibilidad de multi-threading
- Soporte CommonKADS
  - extensión de plataformas dedicadas (ej. librerías de objetos)
  - link con herramientas CASE soportando CommonKADS

Formulario DM-2

Modelo de Diseño	DM-2: Plataforma Implementación
PAQUETE SOFTWARE	Nombre paquete
HARDWARE POTENCIAL	Nombre del hardware
HARDWARE META	Nombre del hardware
LIBRERÍA VISUALIZACIÓN	Disponibilidad y estructuras
LENGUAJE DE IMPLEMENTACIÓN	O-O? Herencia múltiple?
REPRESENTACIÓN DEL CONOCIMIENTO	Declarativo, Procedimental, Híbrido?
PROTOCOLOS DE INTERACCIÓN	ODBC, CORBA, etc.?
FLUJO DE CONTROL	Protocolo paso mensajes? Multithreading?
SOPORTE CKADS	Arquitectura implementada? Herramienta CASE?

Ejemplos: Aion, Prolog y Java

AionDS 8.0

- Librería de objetos vistas
- Representación de conocimiento (Semi-declarativa)
- Interfaces ODBC/CORBA
- utilidades de escritura O-O (incluyendo relaciones)
- protocolo de paso de mensajes O-O
- soporte CommonKADS
  - framework dedicado

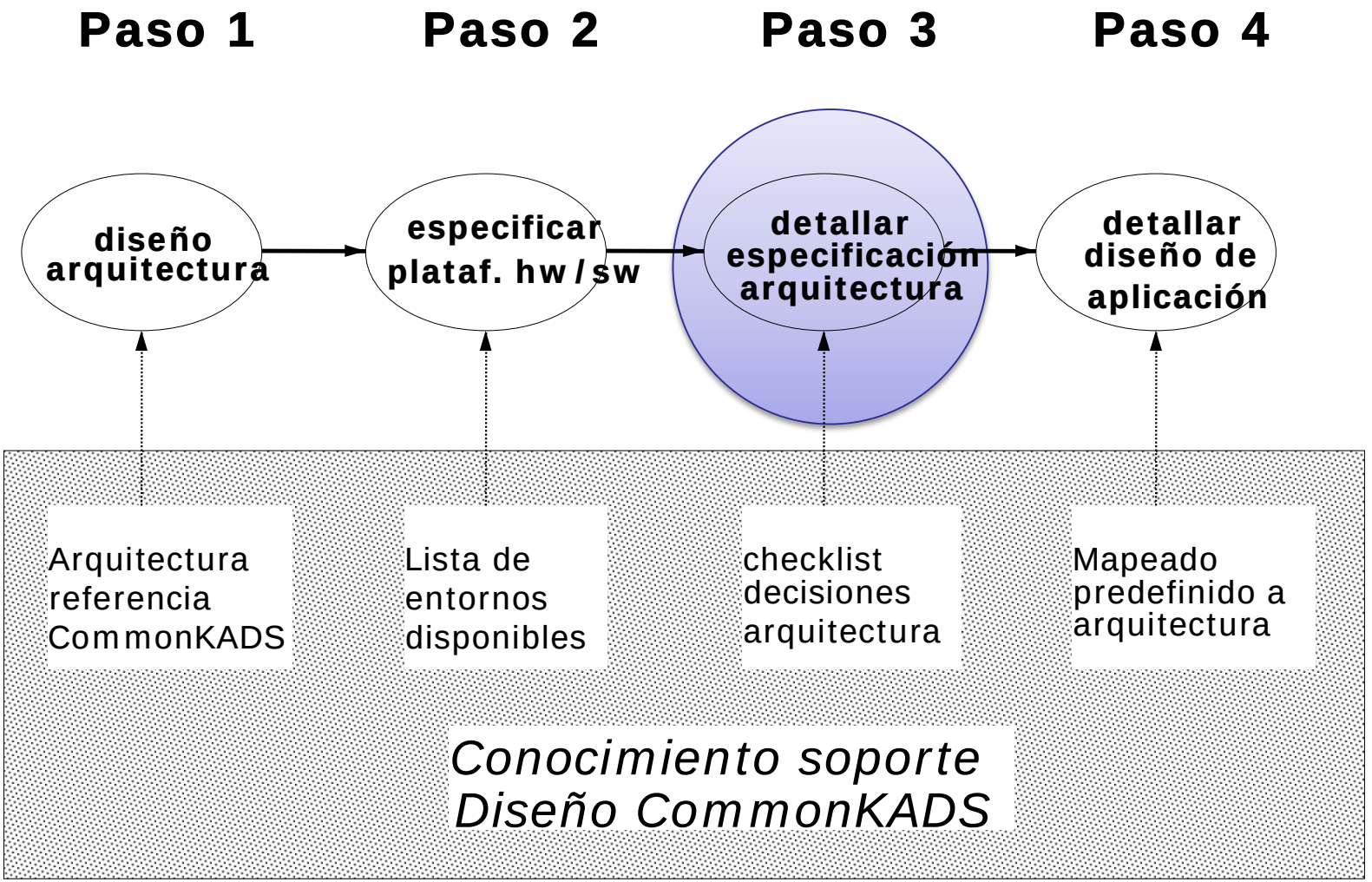
JAVA

- Librería de vistas
- no hay repres. declarativa del conocimiento
- interfaces DB
- utilidades de escritura tipo C++
- utilidades de control: e.j. multi-threading
- actualmente no tiene soporte CommonKADS

PROLOG

- Librería Vistas: depende del vendedor
- Representación de conocimiento declarativa
- Interfaces BBDD: depende del vendedor
- Tecleado “débil” del lenguaje
- No existe un gestor de eventos estándar/ni protocolos de paso de mensaje
- Tiene algunas herramientas de soporte(API)

Pasos en el diseño del sistema



Paso 3: Especificar componentes arquitecturales

- Definir componentes de la arquitectura en más detalle.
  - Definir interfaces entre los subsistemas y/o los módulos de los sistemas
  - Especificar los componentes de las interfaces
- Diseño general de las utilidades de la arquitectura
- Algunas plataformas proporcionan arquitectura CK en las que las decisiones han sido predefinidas
  - ventajas: paso 3 más fácil y rápido
  - desventajas: destruye la capacidad creativa

Utilidades del controlador

- El controlador realiza un control dirigido por eventos y con un componente central de control.

- Puede verse como una implementación del modelo de comunicación

- Decisiones típicas:
  - activación/finalización de funciones del modelo de la aplicación
  - decidir posibilidad interuupciones usuario para información de trazado/contexto
  - posibilidad de abortar funciones
  - manejo de funciones de transferencia
  - Necesidad o no de procesado concurrente

Opciones básicas

- Gestores de eventos para activar funciones del modelo de la aplicación requeridas por agentes externos
- Handlers de eventos para las f. transf. Receive del modelo de aplicación
- Handlers de eventos para eventos internos, generados por f. tipo obtain.
  - en este caso, +mecanismo suspensión/reanudación
- Handlers de eventos para proporcionar información del proceso de razonamiento
- Handlers de eventos para abortar ejecuciones de funciones
- Típicamente cada transacción se implementará en un handler de eventos, o como una combinación de ellos.
- El controlador puede definir sus propias vistas de objetos

Utilidades del modelo de aplicación (1)

- Tarea: definir dos operaciones.
- Método de inicialización, para iniciar los valores de las entradas de la tarea
- Método de ejecución, que invoque al método de la tarea
  - Decidir si devuelve valor booleano de éxito

- Método de la tarea: Dos decisiones de operacionalización de estructura de control
- Lenguaje de estructura de control: Suele especificarse en forma imperativa y definirse en pseudócódigo informal
  - El diseñador decide el conjunto de constructos
    - Sequenciación
    - Selección (If..then..else)
    - Iteración (repeat-until, while-do, for-do)
    - Posible procesado concurrente y sincronización
    - Parte del control por invocación de operaciones en otros objetos de la aqruitectura (roles dinámicos, tareas, inferencias, y funciones de transferencia)
- Lugar donde se define la estructura de control.
  - En O-O dentro de un método ejecutar

Utilidades del modelo de aplicación (2)

- Inferencia
  - Memoria interna para almacenar las soluciones
  - Tres operaciones: execute, new-solution?, has-solution?
  - Unión a los métodos de la inferencia
- Método de la Inferencia (satisfacción de restricciones, encadenamiento progresivo,regresivo, generalización, etc.)
  - Permitir relaciones muchos-a-muchos entre método e inferencia
  - librería de métodos
- Función de transferencia
  - implementada via patrones de paso-de-mensajes
- Roles dinámicos
  - tipos de datos permitidos: “elemento”, “conjunto”, “lista” ?!
  - Operaciones de acceso/actualización: select, add, substract, empty?,...
  - Mantenimiento de verdad, comprobaciones new-solution, has-solution.

Utilidades del modelo de aplicación (3)

- Roles estáticos
  - funciones de acceso/query
    - Proporcionar todas las instancias asociadas al rol
    - Proporcionar una única instancia del rol
    - Responder a si existe una instancia determinada
- Bases de conocimiento
  - formato representational
  - funciones de acceso/query
  - funciones de modificación/análisis (si es necesario)
- Constructos del dominio
  - (sólo inspeccionar)

Utilidades de vistas

- La vistas realizan la presentación del sistema a los agentes externos. Aspectos de diseño:
  - Tipos de vistas que vamos a usar (ventanas, menús,…)
  - Utilidades de la arquitectura que permitan asociar objetos del modelo de la aplicación a una o más vistas, garantizando integridad
    - Utilidades arquitecturales de actualización de vistas
      - tablas de mapeado
      - protocolos de mensajes

- Para interfaces de usuarios, los tipos de vistas suelen ser Visualizaciones gráficas estándar.

- Para presentación a otros sistemas software suele ser estándar
  - e.g. query SQL
  - tablas de mapeado

Interfaces de usuario

- Dos tipos de interfaces:
  - usuarios finales y expertos

- Interfaz usuario final
  - Decisión principal de diseño: si el entorno meta da suficiente capacidad de "visión".
  - Estado del arte: Interfaz gráfica de manipulación directa
  - Posibilidad de considerar utilidades especiales: generación lenguaje natural, ….
  - Uso de visualizaciones específicas del dominio=> depende del diseño de la aplicación.

- Interfaz expertos
  - interfaz de trazados
  - interfaz de edición/ refinado para bases de conocimiento

Formatos típicos de interfaz de trazado

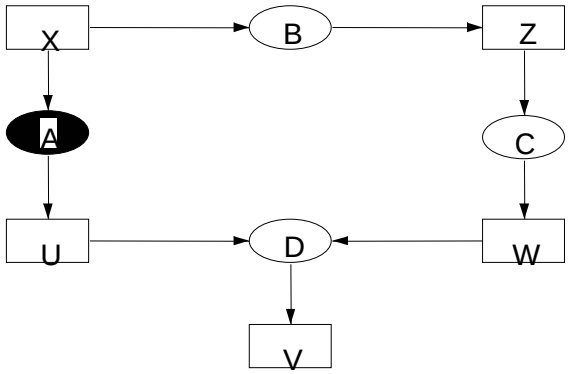
Application tr

Control Tarea

EXECUTING task t2

REPEAT
NEW-SOLUTION(inference B)
inference
inference B
UNTIL HAS-SOLUTION(inference D)

Estructura inferencial



Uniones de roles dinámicos

Role U object 1	**Role V** object 1 object 2	**Role W** object 3
Role X object 4 object 5	**Role Y** object 6	**Role Z** object 7

Ligaduras de roles estáticos

Domain knowledge used by inference A

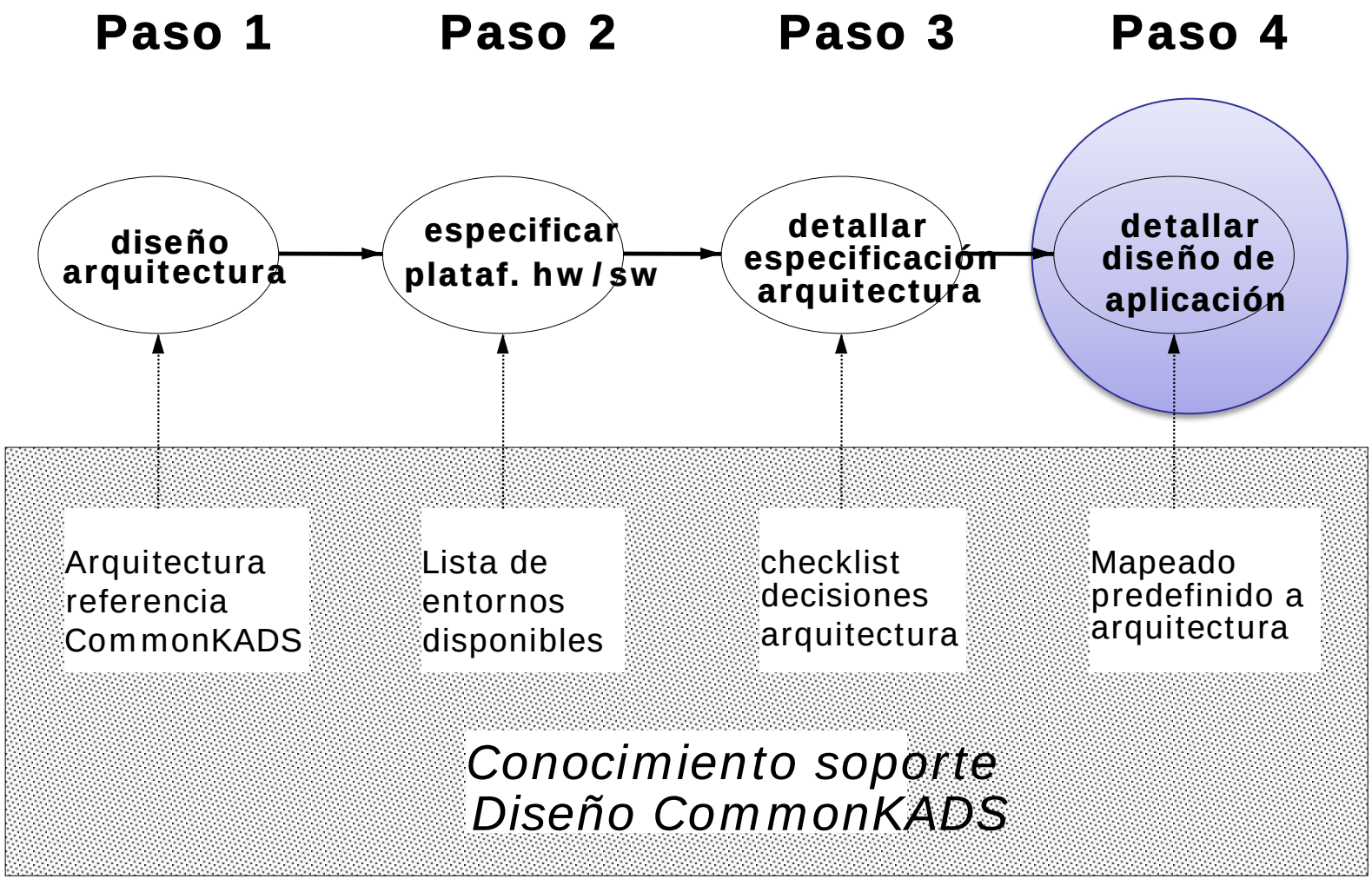
IF obj.a1 > 2 AND obj.a2 < 4
THEN obj.a3 = "xyz"

IF obj.a1 =< 2
THEN obj.a3 = "abc"

Formulario DM-3

Componente arquitectura	Puntos típicos de decisión
CONTROLADOR	Mecanismos internos/externos para manejo de eventos? Concurrencia?Interrupciones? Escenarios what-if? Control del usuario sobre la estrategia de razonamiento?
TAREA	Método de inicialización. Permisión de fallos
MÉTODO TAREA	Lenguaje para estructura de control. Método declarativo o procesal
INFERENCIA	Definir variables de estado interno, cuándo debe resetearse la variable (ej. después de completar tarea) Definir operaciones de ejecución y test de prueba (has-solution? New-solution?)
MÉTODO INFERENCIA	Mapeado muchos-a-muchos de inferencia a método de inferencia. Algoritmo elegido. Catálogo de métodos?
ROL DINÁMICO	Tipos de datos disponibles. Operaciones de acceso/modificación para cada tipo
ROL ESTÁTICO	Definir operaciones acceso: dar-todos? Dar-uno? Existe –uno?
BASE DE CONOCIMIENTO	Decidir representación de las reglas. Definir acceso y métodos de modificación/análisis, i.e. interfaz de expertos
VISTAS	Interfaz gráfica estándar de manipulación directa? Otras utilidades especiales. Diferentes interfaces (usuarios finales y expertos). Utilidades de entrenamiento genéricas

Pasos en el diseño del sistema



Paso 4: Especificar la aplicación en la arquitectura

- Paso 4a: “mapear la información de análisis en la arquitectura del paso previo”
  - asegura la aproximación de mantenimiento de la estructura
  - el mapeado puede ser manual (incómodo) o automático (preferible)
  - Uno de los criterios de selección del entorno de implementación es si posee herramientas de soporte para el mapeado
    - Creación automática de especificaciones de la estructura de control
    - Creación automática de instancias de reglas
    - Creación automática de de objetos controlador

- Paso 4b: “añadir detalles de diseño necesarios para el diseño de la aplicación”
  - lista de detalles de diseño que necesitamos añadir para completar la operacionalización de un modelo de análisis

Paso 4b. Añadir detalles específicos de diseño

- Diseño de la aplicación para el controlador
  - Entrada principal: modelo de comunicación
  - Mínimo: Proceso de inicialización del Sistema y Gestores de eventos, para obtener la información de los usuarios
  - Otras funciones:
    - Interacción externa compleja (ej. Homebots)
    - manejo de requisitos de explicación
    - control del usuario sobre el proceso de razonamiento
    - interrupción de razonamiento/control estratégico
    - permitir escenarios “what-if”
    - permitir demonios

Paso 4b. Añadir detalles específicos de diseño

- Diseño del modelo de aplicación

Conjunto mínimo de actividades de diseño de aplicación:

- Para cada método de la tarea:
  - Formalizar el método estructura de control en el lenguaje de control que proporcione la arquitectura.
- Para cada inferencia
  - Escribir la especificación de la invocación de un método inferencial
  - El método debe mostrar cómo los roles estáticos y dinámicos se mapean a argumentos del método inferencial
  - El método inferencial puede ser un método de razonamiento de IA, o un algoritmo estándar (sorting, selección de subconjuntos, etc.)
- Para cada rol dinámico:
  - elegir un tipo de dato, en función de los que proporcione la arquitectura

Paso 4b. Añadir detalles específicos de diseño

- Diseño de vistas para aplicación
  - Elegir una vista para cada objeto de la aplicación, si es necesario
  - Guía: para la interfaz de usuario usar vistas lo más cercanas posible a los formatos específicos del dominio
    - con demasiada frecuencia los diseñadores del sistema imponemos a los usuarios lo que nos gusta
    - cada dominio tiene su propia “tradición” en la representación de la información (y normalmente hay una buena razón para ello)

Formulario DM-4: Diseño aplicación

ELEMENTO	DECISIÓN DE DISEÑO	COMENTARIOS
CONTROLADOR	Traducir modelo de comunicación, con plan de control y transacciones a gestores de eventos	Necesario comportamiento en tiempo real? Necesario control del usuario sobre razonamiento?
MÉTODOS DE TAREA	Formalizar estructura de control	Fuertemente restringida por el lenguaje de control que proporcione la arquitectura Deseable mapeado automático
ROLES DINÁMICOS	Elegir un tipo de dato para cada rol	Limitado a los tipos de datos que proporcione la arquitectura. Usar conjuntos de roles reales (en vez de listas) en lo posible, para razonamiento más natural.
INFERENCIAS	Escribir especificación de la invocación del método(s) inferencial(es)	El método de invocación de be mostrar mapeado entre roles y argumentos del método. Transformación frecuente de las entradas
MÉTODOS INFERENCIAS	Especificar o elegir métodos	Elegir técnica de razonamiento apropiada o algoritmo. Limitar el nº de métodos , intentando usarlos para más de una inferencia.
BASES DE CONOCIMIENTO	Traducir las instancias de las bases de conocimiento al formato representacional de la arquitectura	Deseable usar herramientas de mapeado automático
OBJETOS VISTAS	Elegir vistas apropiadas para objetos del modelo de la aplicación y controlador	Para interfaz de usuario final, usar en lo posible representaciones específicas del dominio.

Diseño de prototipos. Prototipado sistema razonamiento

- Cuando es útil?
  - El modelo de conocimiento no se basa en una plantilla existente
  - Agujeros o lagunas en el conocimiento del dominio, sin definir claramente
  - El prototipo debe permitir el trazado del proceso de razonamiento en términos del modelo de conocimiento→ necesaria una interfaz
  - En general: V&V del modelo de conocimiento
    - verificación: “is the system right”
    - validación: “is it the right system”

- Debe soportarlo la plataforma de implementación
  - la construcción del prototipo debe ser cuestión de días

Diseño de prototipos. Prototipado Interfaz

- Prototipado que contenga objetos de controlador y vistas
- Comprobar la interfaz sin necesidad de desarrollar toda su funcionalidad

- Puede ser necesario:
  - interacción externa compleja (e.g.; HOMEBOTS)
  - formatos de vistas complejos
  - agregaciones de vistas complejas

Bibliografía

- A. Alonso Betanzos, B. Guijarro Berdiñas, A. Lozano Tello, J.T. Palma Méndez, M.J. Taboada Iglesias. Ingeniería del Conocimiento. Aspectos metodológicos. Pearson Educación, 2004.

- G. Schreiber, H. Akkermans, A. Anjerwierden, R. de Hoog, N. Shadbolt, W. van der Belde and B. Wielinga. Knowledge engineering and management. The CommonKADS methodology. MIT Press, 2000.

UNIVERSIDADE DA CORUÑA

fic
facultade de
informática
da coruña

TEMA 4:
MODELO DE DISEÑO en CommonKADS