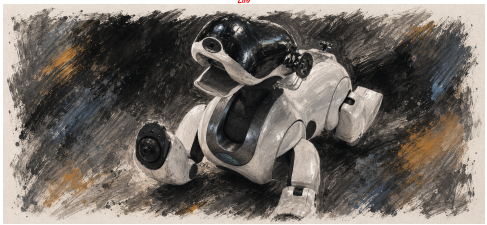
Arquitectura software en robótica

Arquitecturas Software para Robots, Grado en Ingeniera de Robótica Software

Francisco Martín Rico y Rodrigo Pérez Rodríguez

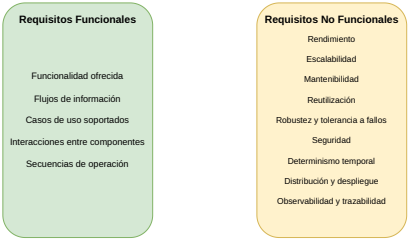
Intelligent Robotics Lab



Qué es arquitectura software

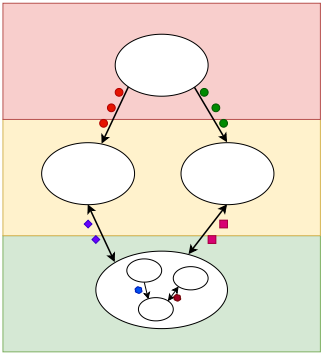
- **Definición:** la arquitectura software describe la **estructura fundamental** de un sistema (componentes, responsabilidades y relaciones) y los **principios** que guían sus decisiones de diseño.

- Puente entre requisitos y código: traduce requisitos funcionales y no funcionales en decisiones estructurales.



Qué es arquitectura software

- Define elementos arquitectónicos: módulos, interfaces, mecanismos de comunicación y patrones (desacoplamiento, modularidad, jerarquía).
- Condiciona la evolución: integración de nuevos componentes, reutilización, testabilidad y contención de fallos.
- No existe una arquitectura universal: se diseña por compromisos y se valida contra el contexto y los requisitos.

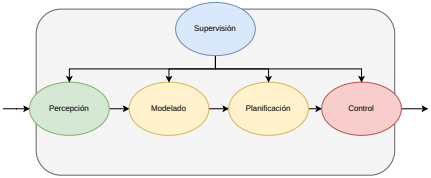


Qué es arquitectura software en robótica

- Organización del software que permite al robot percibir, decidir y actuar en el mundo físico.
- Particularidades: entorno dinámico e incierto, datos sensoriales con caducidad, y actuación con consecuencias físicas.
- La arquitectura debe garantizar un comportamiento **coherente, robusto y seguro** ante incertidumbre, degradaciones y fallos.
- Adapta principios clásicos (modularidad, desacoplamiento, reutilización) a un dominio con requisitos **físicos y temporales**.
- Exige tratar explícitamente tiempo y concurrencia: latencia, sincronización, priorización y tolerancia a fallos.

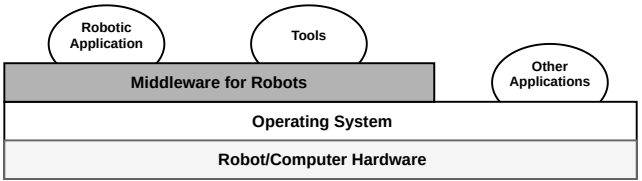
Qué es arquitectura software en robótica

- Estructura funcional típica: percepción, modelado, planificación, control y supervisión, con dependencias fuertes.
- Define cómo fluye la información: **representaciones compartidas**, coordinación de comportamientos y decisiones **reactivas** vs. **deliberativas**.
- Debe ser **adaptable y extensible**: integrar nuevos sensores, sustituir algoritmos e incorporar capacidades sin reescribir el sistema.
- Requiere monitorización y trazabilidad: exponer estado interno y reconstruir decisiones para depurar, validar y asegurar.



Middlewares de programación de robot: idea principal

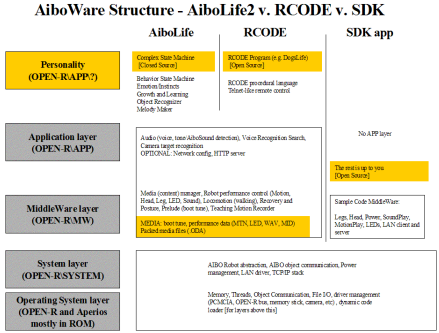
- Un middleware evita rehacer infraestructura: comunicación, abstracción de hardware y modelos de ejecución.
- Materializa la arquitectura: convierte interacciones y componentes definidos en diseño en comportamiento en ejecución.
- Condiciona decisiones arquitectónicas: acoplamiento posible, modelo de concurrencia, distribución y garantías temporales.
- Suele incluir: transporte de mensajes, descubrimiento, serialización, *runtime*, herramientas de despliegue e introspección.



Middlewares: ejemplos y rasgos arquitectónicos

Open-R (Aibo)

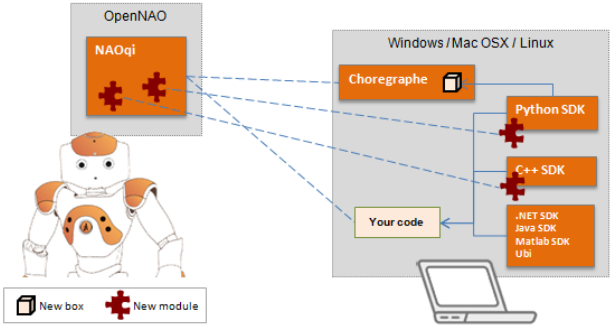
- **Open-R (Aibo):** tiempo real fuerte; arquitectura de objetos concurrentes (un *thread* por objeto) y comunicación por mensajes tipados.



Middlewares: ejemplos y rasgos arquitectónicos

NaoQi (Nao/Pepper)

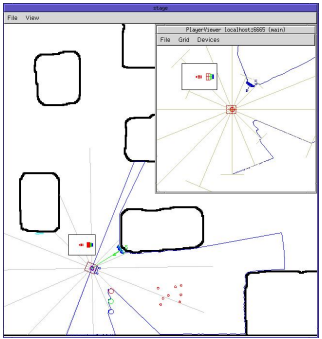
- **NaoQi (Nao/Pepper):** módulos preexistentes accesibles por *proxies*; *blackboard* interna y modelo multihilo gestionado por el middleware.



Middlewares: ejemplos y rasgos arquitectónicos

Player/Stage

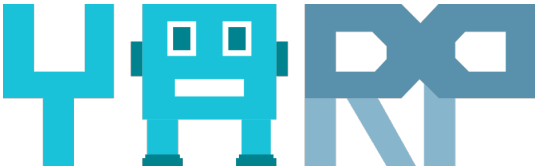
- **Player/Stage:** enfoque cliente–servidor; separación clara hardware–algoritmos y simulación 2D para reutilizar el mismo software.



Middlewares: ejemplos y rasgos arquitectónicos

YARP

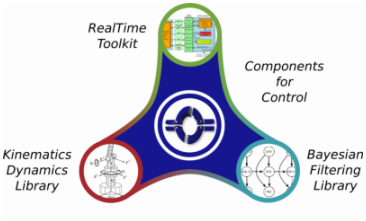
- **YARP:** comunicación por *ports* tipados y conexiones dinámicas; favorece sistemas distribuidos y débilmente acoplados.



Middlewares: ejemplos y rasgos arquitectónicos

Orocos

- **Orocos:** orientación a control en tiempo real; componentes con *ports* y ejecución determinista para capas de bajo nivel.

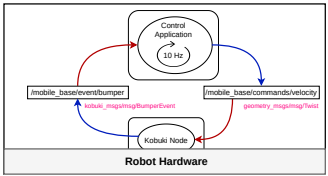


ROS y ROS 2 como *middleware* de facto

- ROS/ROS 2 es el estándar de facto: comunidad muy amplia y ecosistema de paquetes reutilizables (navegación, manipulación, percepción, simulación).
- ROS 2 amplía ROS para sistemas reales: distribución nativa, multi-robot y soporte de tiempo real sobre DDS.
- Impacto arquitectónico: reutilización de paquetes y convenciones de la comunidad condicionan estructura y decisiones de integración.
- Aplicaciones: desde prototipos de investigación hasta robots industriales, móviles, de servicio y vehículos autónomos.

ROS 2 como materialización de una arquitectura software robótica

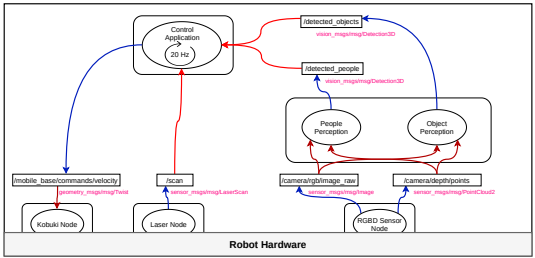
- ROS 2 no impone una arquitectura funcional: proporciona primitivas para implementar la arquitectura que el ingeniero diseña.
- Modelo arquitectónico: nodos desacoplados que interactúan mediante topics, servicios y acciones.
- Integra requisitos robóticos como elementos centrales: tiempo, concurrencia y distribución desde el diseño del *runtime*.
- Incorpora observabilidad y trazabilidad: introspección del sistema y registro de eventos para cerrar el ciclo diseño–ejecución.

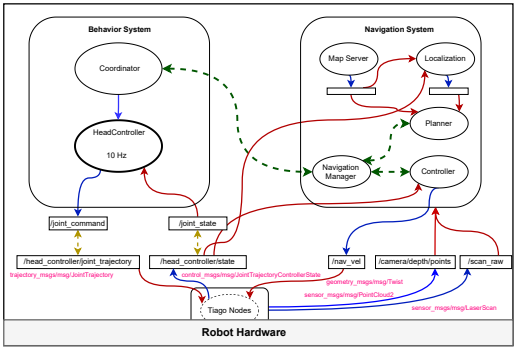


Componentes, procesos y comunicación en ROS 2

- **Nodo** como unidad arquitectónica: encapsula responsabilidades y delimita interfaces.
- **Proceso** como decisión de despliegue: aislamiento y robustez frente a latencia y consumo de recursos.
- **Topics** para datos asíncronos (desacoplamiento productor–consumidor) y flujos continuos.
- **Servicios** para peticiones puntuales síncronas; **acciones** para tareas largas con *feedback*.
- Elegir mecanismo de comunicación es una decisión arquitectónica: define dependencias, acoplamiento y ritmo del sistema.

Componentes, procesos y comunicación en ROS 2





Componentes, procesos y comunicación en ROS 2

- Modelo orientado a eventos: los componentes reaccionan a datos, temporizadores y peticiones externas.
- Concurrencia de actividades: percepción, procesamiento, control y supervisión progresan de forma solapada.
- Diseño distribuido por naturaleza: nodos en procesos y máquinas distintas con comunicación transparente en red.
- **Introspección** del grafo y **monitorización**: nodos, interfaces, conexiones y salud del sistema en ejecución.
- **Logging** y **trazas** del *runtime* para reconstruir decisiones, validar comportamiento y facilitar depuración.

Preguntas