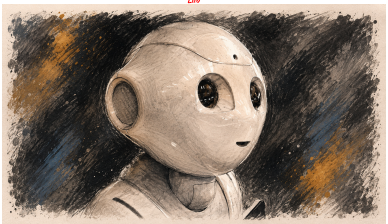
Capítulo 2: Ejecución y comunicación básica

Nodos, datos y eventos

Francisco Martín Rico y Rodrigo Pérez Rodríguez

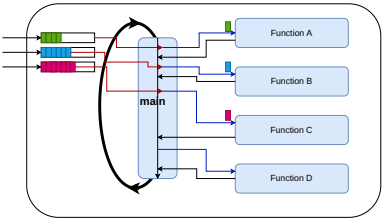
Intelligent Robotics Lab



Modelos de ejecución reactivos

Secuencial vs. reactivo

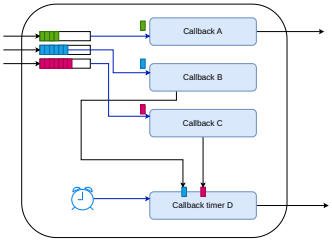
- La ejecución reactiva hace que el comportamiento emerja de **eventos** (estímulos externos e internos).
- La asincronía y la concurrencia rompen el control secuencial: el **orden de ejecución** no se puede predecir con exactitud.
- El robot **adapta su comportamiento** continuamente según la información que recibe.
- Los lazos periódicos pueden entenderse como **eventos temporales** dentro del modelo reactivo.



Modelos de ejecución reactivos

Arquitectura dirigida por eventos

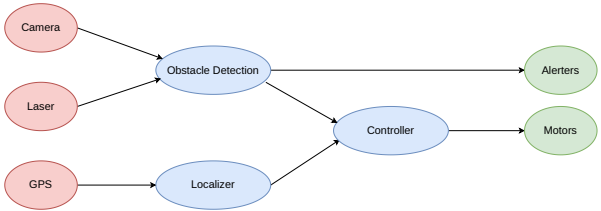
- Los componentes se activan solo ante **estímulos relevantes** y el comportamiento global surge de reacciones locales.
- El modelo reactivo reduce latencia ante cambios del entorno (dinámico e impredecible).
- La modularidad aumenta, pero exige **coordinar reacciones** y evitar emergencias no deseadas.
- Se diseña por **condiciones de reacción**, no por pasos; encaja con incertidumbre, asincronía y concurrencia propias de la robótica.



Flujo de datos, eventos y callbacks

De flujos a eventos (visión arquitectónica)

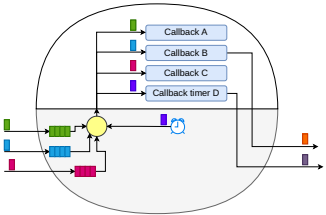
- La arquitectura se entiende como **productores y consumidores** conectados por flujos con contratos explícitos.
- Cada flujo impone requisitos (frecuencia/latencia) y hace visibles dependencias, puntos críticos y posibilidades de reutilización.
- En implementación, los flujos suelen materializarse como **eventos** que activan callbacks: asincronía y concurrencia son inherentes.
- Para mantener baja latencia y coherencia: callbacks cortos/no bloqueantes, responsabilidades claras y mínimo estado compartido.



Nodos

Unidad básica de ejecución en ROS 2

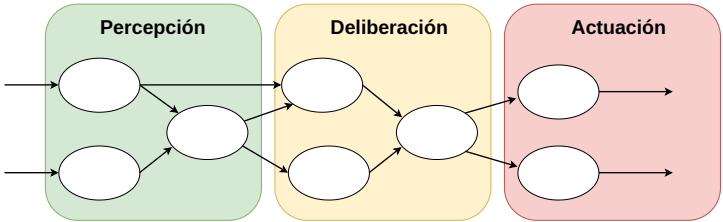
- Los nodos son procesos reactivos que ejecutan callbacks.
- Cada nodo encapsula funcionalidad y expone interfaces explícitas.



Nodos

Responsabilidades y cohesión

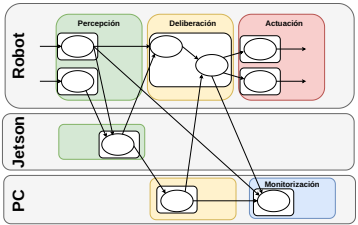
- La cohesión de cada nodo evita monolitos difíciles de mantener.
- La lógica interna debe tolerar asincronía y concurrencia.



Nodos

Despliegue distribuido y arquitectura

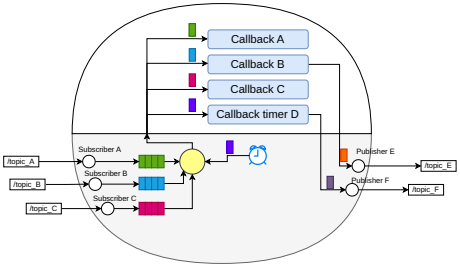
- Interfaces limpias facilitan despliegue distribuido y escalado.
- Los nodos concretan eventos, datos y desacoplamiento en ROS 2.



Publishers y subscribers

Comunicación por flujo de datos (pub/sub)

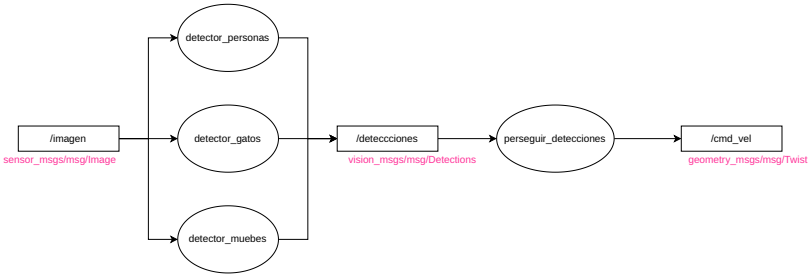
- Pub/sub materializa el **flujo de datos** en ROS 2: sensórica, estado y comandos frecuentes.
- El **desacoplamiento** se logra mediante contratos de mensajes (tipo + semántica) que estabilizan interfaces.
- Granularidad/frecuencia son decisiones arquitectónicas: tolerar latencias/pérdidas y diseñar flujos observables y depurables.



Topics

Diseño de canales e interfaces públicas

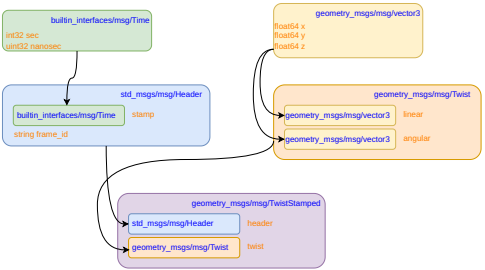
- Un topic es un **canal lógico tipado** y una interfaz pública: condiciona la evolución del sistema.
- Tipo + nombre fijan el **contrato** y sirven de documentación viva (reutilización/interoperabilidad).
- Frecuencia/contenido determinan latencia y flexibilidad; evitar proliferación y semánticas implícitas reduce complejidad.



Primeros mensajes

Mensajes como contrato de datos

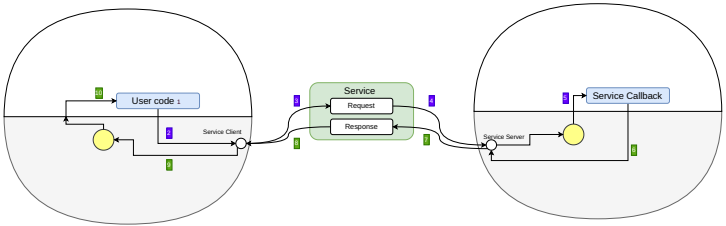
- El mensaje es el **contrato de datos**: su contenido condiciona integración y evolución.
- Semántica explícita (unidades, *timestamps*, referencia espacial) aporta contexto y coherencia.
- Claridad evita acoplamientos/deuda; mensajes estándar mejoran interoperabilidad y los propios se diseñarán en prácticas.



Servicios y acciones

Servicios (petición–respuesta)

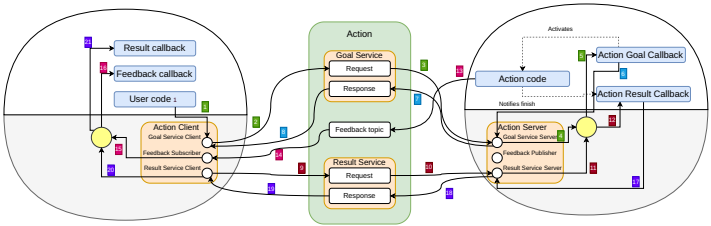
- Servicios y acciones complementan a los flujos continuos.
- Los servicios son adecuados para operaciones puntuales con respuesta.
- Elegir servicio o topic depende de la semántica de la información.



Servicios y acciones

Acciones (tareas largas) y uso equilibrado

- Las acciones modelan tareas largas con feedback y cancelación.
- Las acciones exponen capacidades a la lógica de alto nivel.
- Servicios y acciones también se implementan mediante eventos.
- Un uso equilibrado evita acoplamientos rígidos o diseños confusos.
- Topic, servicio y acción se eligen según la interacción deseada.



Preguntas