Capítulo 4: Arquitecturas deliberativas e híbridas

Capas y modelos del mundo

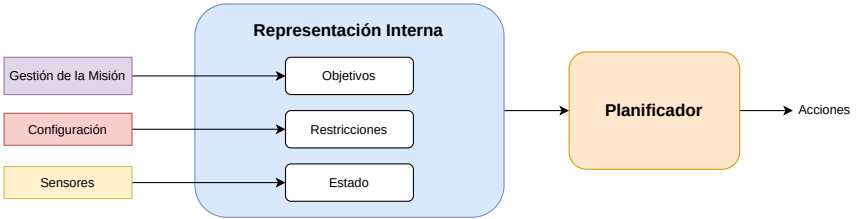
Francisco Martín Rico y Rodrigo Pérez Rodríguez

Intelligent Robotics Lab

De lo reactivo a lo deliberativo e híbrido

Por qué no basta con reaccionar

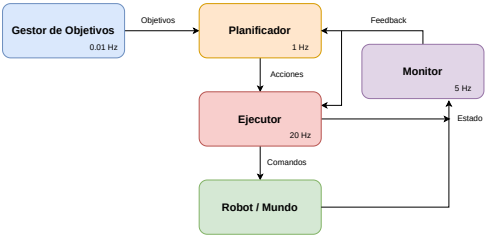
- Lo reactivo funciona bien para reflejos y bucles rápidos: "si ocurre X, haz Y".
- En misiones largas aparecen límites: miopía temporal, oscilaciones y deuda de estado.
- La deliberación introduce representación interna: estado + objetivos + restricciones, y planifica sobre ellos.
- Deliberar obliga a responder: *qué sé* (modelo del mundo), *qué quiero* y *qué puedo hacer*; y a explicitar interfaces, estados y contratos.



De lo reactivo a lo deliberativo e híbrido

Arquitecturas híbridas: combinación práctica (1/2)

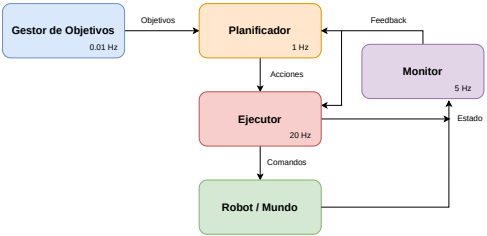
- La deliberación no sustituye a la reactividad: la **complementa** (bucles rápidos + decisión de alto nivel).
- La clave no es "mezclar" módulos, sino **conectar** planificación y reactividad con reglas claras.
- Reglas mínimas: **prioridades**, **flujo de información** y **autoridad** para resolver desacuerdos planificación–reactividad.



De lo reactivo a lo deliberativo e híbrido

Arquitecturas híbridas: combinación práctica (2/2)

- Diseñar por **escalas temporales** y roles: ms (seguridad/control), s (ejecución), min (misión).
- Tratar el **estado de mundo** como activo: modelo del mundo consultable (incertidumbre y vigencia) que cambia decisiones.
- Hacer el sistema operable: **contratos de ejecución** (progreso/cancelación/timeout) + **supervisión** y replanificación por eventos.



De lo reactivo a lo deliberativo e híbrido

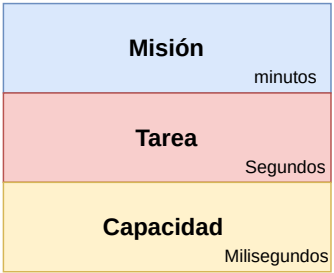
Receta de diseño: roles, ritmos y contratos

- **Fija el horizonte y el ritmo**: qué se decide en ms (reactivo), en s (ejecución) y en min (misión).
- **Define objetivos y restricciones verificables**: éxito, límites de seguridad, recursos y prioridades.
- **Enumera capacidades** como "servicios con contrato": entradas, feedback, resultados y fallos con significado.
- **Decide qué estado debe ser explícito** (modelo del mundo): vigencia e incertidumbre, y qué queda como señal reactiva.
- **Diseña el bucle de supervisión**: qué se monitoriza, umbrales de recuperación y cuándo replanificar.
- **Establece reglas de arbitraje** entre capas: prioridades, cancelación y comportamiento seguro por defecto.

Arquitecturas en capas: misión, tarea y capacidad

Separar por nivel de abstracción

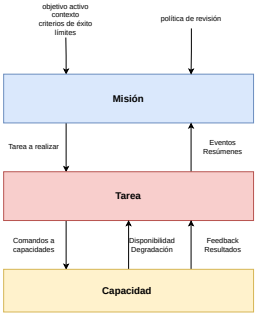
- Tres capas útiles: **misión** (intención/política), **tarea** (orquestación), **capacidad** (acción segura).
- Cada capa decide con horizonte y ritmo distintos; consume estado distinto y produce garantías distintas.
- Fronteras por preguntas: qué (misión), cómo (tarea), con qué (capacidad).
- Hacia abajo bajan objetivos/restricciones; hacia arriba suben progreso/resultado/razón de fallo.
- La tarea hace de puente: misión no baja a sensores y capacidades no conocen el objetivo global.



Arquitecturas en capas: misión, tarea y capacidad

Interfaces y contratos entre capas

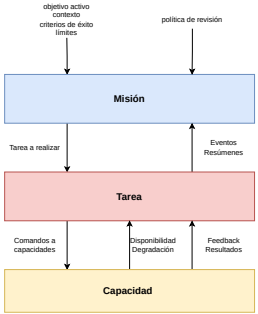
- **Misión:** objetivos, prioridades, restricciones, criterio de éxito y política de revisión.
- **Tarea:** curso de acción con estado de progreso, contingencias y recuperación acotada.
- **Capacidad:** servicio gobernable con precondiciones, parámetros, feedback, cancelación y fallos.
- Contratos explícitos evitan acoplamientos informales y facilitan evolución independiente.
- Anti-oscilación: prioridades + cancelación + histéresis (umbrales, tiempos mínimos, cooldown).



Arquitecturas en capas: misión, tarea y capacidad

Checklist por capa (antes de implementar)

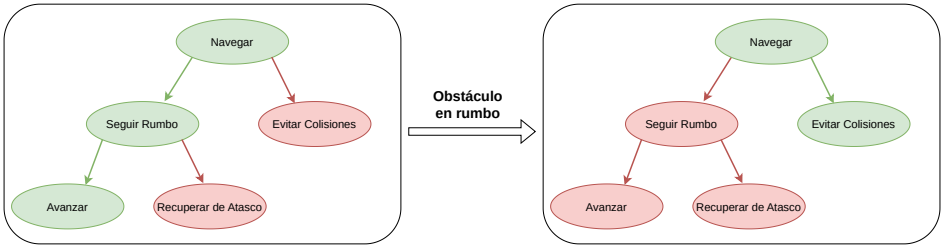
- **Para misión:** objetivos, prioridades, restricciones y disparadores de revisión.
- **Para tarea:** estados de progreso, eventos de recuperación y resumen hacia misión.
- **Para capacidad:** precondiciones, parámetros, feedback, resultado, cancelación y fallos.
- **Para fronteras:** timeouts, invariantes de seguridad y autoridad para pausar/cancelar.
- Resultado: **coherencia**, **reutilización** y **operabilidad** del sistema.



Subsumption y arbitraje reactivo

Prioridad por supresión/inhibición

- El comportamiento emerge de capas reactivas en paralelo que compiten por el control.
- El arbitraje se basa en **supresión** e **inhibición**: capas superiores anulan salidas inferiores.
- Útil para reflejos robustos: seguridad puede imponerse sin esperar deliberación.
- Enfoque incremental: añadir capas sin reescribir las anteriores.
- En híbridos, conviene encapsularlo dentro de capacidades o estratos reactivos con contratos hacia arriba.



Subsumption y arbitraje reactivo

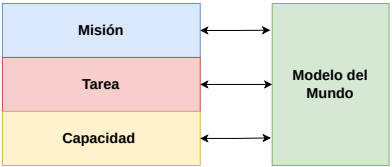
Ventajas, límites e integración sana

- **Ventaja:** responsividad y robustez ante ruido/cambios rápidos.
- **Ventaja:** fuerza a explicitar prioridades y autoridad de control.
- **Límite:** objetivo global y estado suelen quedar implícitos (opacidad, oscilaciones sutiles).
- **Límite:** escala peor en misiones largas sin estado y objetivos explícitos.
- Integración recomendada: reflejos/reactividad en capacidades, y misión/tarea gobiernan intención y cancelación.

Modelos del mundo

Frontera entre percepción y decisión

- Un modelo del mundo integra percepciones en **conocimiento consultable** para decidir.
- Actúa como frontera: ingiere evidencia (mediciones/detecciones/estimaciones) y publica estado.
- Incluye asociación de datos, estimación (filtrado/seguimiento) y gestión temporal (caducidad).
- Debe representar incertidumbre, confianza y vigencia: condicionan seguridad y replanificación.



Modelos del mundo

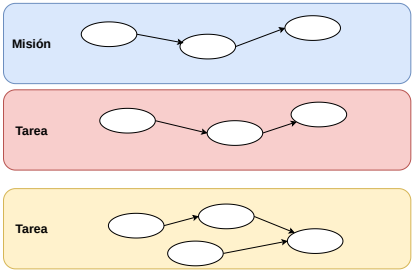
Consultas orientadas a decisiones

- Publicar datos no basta: interesan **interfaces de consulta** (dónde estoy, obstáculos cerca, estado de entidades).
- Diseñar consultas reduce acoplamiento a formatos/tasas/fuentes perceptivas concretas.
- Consultas y estado permiten trazabilidad: explicar qué creía el robot y por qué decidió.
- La deliberación se estabiliza con disparadores y presupuestos: replanificar por eventos relevantes.
- Regla práctica: modelar explícitamente lo que *cambia decisiones* (sin intentar modelarlo todo).

Mapeo de capas a un grafo de ROS 2

Topología: roles y contratos

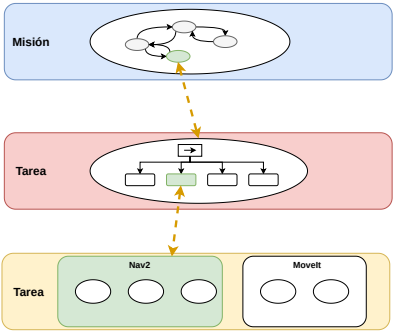
- En ROS 2, la arquitectura se materializa como **grafo** de nodos y comunicaciones.
- Diseñar el grafo por roles: misión (supervisor), tareas (orquestador), capacidades (proveedores).
- Conexiones representan contratos: órdenes, feedback, estado y eventos.
- Acciones para actividades largas y cancelables; topics para estado continuo; servicios para consultas puntuales.
- Namespaces, parámetros y composición ayudan a mantener modularidad sin perder eficiencia.



Mapeo de capas a un grafo de ROS 2

Capacidades como subsistemas gobernables

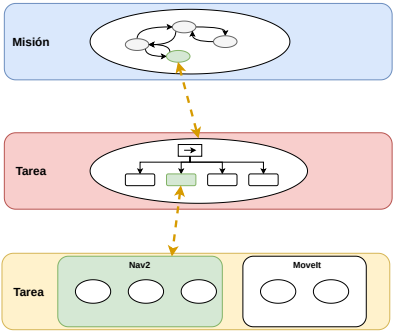
- Una capacidad no es "un mensaje": es una **actividad** de larga duración.
- Contrato típico: **progreso y cancelación** (acciones) + **estado/telemetría** (topics).
- Publicar **diagnósticos** y estado facilita supervisión y recuperación.
- Activación y degradación sin reinicios: nodos *managed* (ciclo de vida) cuando aplica.
- Ejemplos: Nav2 y MoveIt 2 se consumen desde arriba como servidores con estado y preempción.



Mapeo de capas a un grafo de ROS 2

Tarea como orquestador (Behavior Trees)

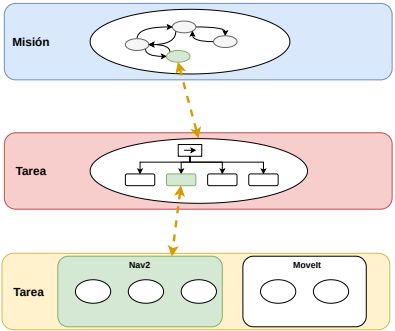
- La tarea integra capacidades, decide el orden y mantiene **contexto** de ejecución.
- Un BT "tiquea" condiciones y acciones: consulta estado, lanza acciones y decide reintentos.
- Ventaja operativa: **progreso observable**, **timeouts** por rama y **recuperación local**.
- Preempción sin bloquear: cancelar una rama no debe congelar el resto del sistema.
- En el grafo: cliente de varias acciones y productor de eventos/resúmenes hacia misión.



Mapeo de capas a un grafo de ROS 2

Misión como FSM

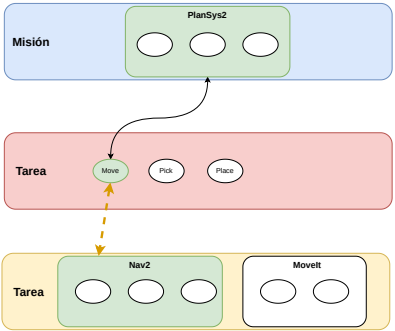
- Útil cuando las situaciones son **acotadas** y el comportamiento es **enumerable**.
- La misión opera a **bajo ritmo** y es soberana: cambia objetivos, preempta/cancela y registra razones.
- La tarea (p. ej., BT) sostiene la ejecución: progreso, timeouts y recuperación local.
- Contratos claros hacia abajo: acciones en capacidades y eventos/resúmenes hacia arriba.



Mapeo de capas a un grafo de ROS 2

Misión como planificador (PlanSys2)

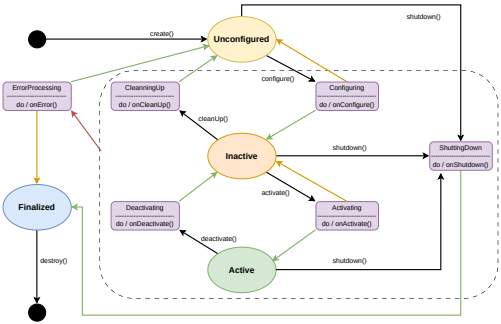
- Conveniente si hay que razonar sobre **objetivos, recursos y restricciones**.
- Flujo típico: **objetivos** → **plan** → **tareas** que ejecutan pasos.
- La misión mantiene política y revisión: cuándo replanificar y cómo priorizar bajo incertidumbre.
- Requiere trazabilidad: registrar **razones** (qué hecho/condición motivó la decisión).



Lifecycle Nodes y subsumption

Lifecycle Nodes: qué son y cómo funcionan

- Un **Lifecycle Node** separa *crear* el nodo de *ponerlo a trabajar*: puede arrancar y quedar *inactivo*.
- Implementa una **máquina de estados**: *unconfigured* → *inactive* → *active* (y *finalized*).
- Las **transiciones** (configure/activate/deactivate/cleanup/shutdown) disparan callbacks de init/liberación.
- Permite orquestar **arranques ordenados**, **activaciones controladas** y **paradas seguras**.
- Aporta operabilidad: comprobar preparación, degradar por desactivación y recuperar sin reiniciar todo.



Lifecycle Nodes y subsumption

Subsumption con Cascade Lifecycle Node

- **Subsumption** organiza comportamientos en capas con prioridad: unas capas *suprimen/inhiben* a otras.
- Idea práctica en ROS 2: modelar cada capa como **nodos gobernables** por ciclo de vida (activar/desactivar).
- **cascade_lifecycle**: un Cascade Lifecycle Node declara qué otros lifecycle nodes deben estar activos si él lo está.
- Al activar una capa de alta prioridad, se activan sus dependencias y se **suprimen** capas inferiores desactivando su control.
- La histéresis se puede reflejar en la activación: tiempos mínimos en *active* y *cooldown* antes de volver a bajar.

Construcción del modelo del mundo en ROS 2

Conceptos: subsistema de ingestión y publicación

- Organizar como subsistema: **ingiere** evidencia de múltiples fuentes y **publica** una representación integrada.
- Evitar el "nodo monolítico": separar estimación (localización/tracking) de conocimiento (entidades, mapas).
- Coherencia espacio-temporal: marcas de tiempo y TF consistentes para fusionar y consultar.
- Gestión temporal del conocimiento: frescura, caducidad y nivel de confianza/incertidumbre.
- Objetivo: que misión/tarea pregunten *estado* en lugar de reconstruirlo ad hoc desde streams.

Construcción del modelo del mundo en ROS 2

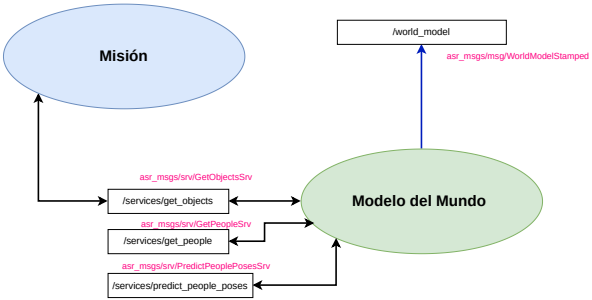
Conceptos: estado publicado + consultas

- Dos caras del modelo: **estado publicado** (monitorización/visualización) y **consultas** (deliberación).
- Consultas orientadas a decisiones: "dónde estoy", "qué hay cerca", "qué entidad está activa".
- Separar interfaces por coste: servicios para consultas rápidas y acciones para consultas costosas (con feedback).
- Disparadores para replanificar: eventos relevantes, cambios de confianza/frescura o fallos de ejecución.
- Operabilidad: diagnósticos de fuentes, frescura, consistencia TF y resumen de incertidumbre.

Construcción del modelo del mundo en ROS 2

Diseño 1: servidor central + interfaces por coste

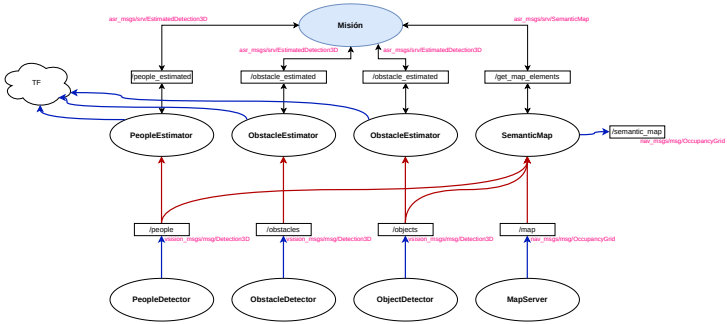
- Un nodo/servidor central mantiene estado integrado y atiende consultas.
- API separada por coste: servicios (rápido) y acciones (caro/cancelable) con progreso.
- Simplifica consistencia y trazabilidad: un punto claro de verdad y diagnóstico.
- Coste: escalado y punto único de fallo si no se replica o degrada con cuidado.
- Útil cuando el dominio es acotado y se busca operabilidad fuerte.



Construcción del modelo del mundo en ROS 2

Diseño 2: vistas federadas

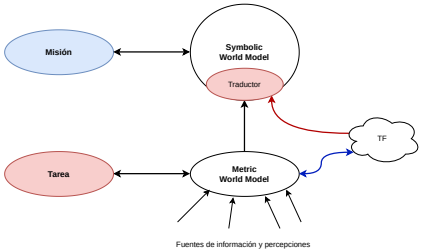
- Cada subsistema publica su mejor estimación con semántica clara (sin "base central" única).
- Vistas/adaptadores sincronizan, transforman (TF) y cachean para responder consultas deliberativas.



Construcción del modelo del mundo en ROS 2

Diseño 3: modelo dual (geométrico + simbólico)

- Separar mundo geométrico (poses, mapas, accesibilidad) y mundo simbólico (hechos/predicados).
- Un traductor discretiza evidencia continua en hechos con umbrales, confianza y caducidad.
- Encaja con planificación: misión consume símbolos; tarea/capacidades consumen geometría.
- Ventaja: responsabilidades claras y explicabilidad (qué hecho habilitó qué acción).
- Riesgo: discretización puede oscilar sin histéresis y gestión temporal explícita.



¿Preguntas?