# 具身智能操作系统技术白皮书

Embodied AI Operating System Technical Whitepaper

版本 v0.1

机构： CCF 泛在操作系统开放社区

协议： MulanPSL- 2.0

时间： 2026 年 1 月 29 日

# 目录

# 插图

# 表格

# 摘要

具身智能正处于快速发展阶段。随着 VLA（视觉－语言－动作） 大模型等关键技术的突破，机器人的通用感知、规划决策与行动执行等能力显著提升，应用前景极为广阔，具身智能有望成为万亿级规模产业。然而， 目前让具身智能机器人稳定、高效地执行真实任务、掌握所需技能，仍然高度依赖大量专业化、定制化的工程开发， 效率低下，成本高昂。其根本原因在于当前具身智能的软件算法模型与硬件物理本体之间存在高度紧耦合关系，导致“硬件难适配、软件难复用”。具身智能产业要迈向万亿级规模， 必然需要类似于通用操作系统之于计算机产业那样的共性基础设施，将机器人硬件与软件解耦，支撑生态化分工、跨平台复用与规模化应用。

本白皮书提出一种面向具身智能的新型操作系统设计——EAIOS （Embodied AI Operating System），试图以操作系统范式重构具身智能的工程基础。EAIOS 采用“原语－服务－技能－任务”对机器人的行动进行抽象， 采用“对象”为基本单元构建映像外部世界的虚拟表征空间，引入世界模型（World Model）与安全内核作为核心运行时支撑， 自底向上统一异构硬件抽象，并向上提供标准化的行动接口。在执行层面， EAIOS 通过世界模型对大模型生成的规划与行为进行推演与验证，在物理执行前确保其满足安全约束与价值伦理要求，提升系统的可靠性与可控性。

这种设计有效实现了具身智能中“身体”（硬件系统） 与“大脑”（模型与软件系统） 的解耦，使大模型提供者、技能开发者与硬件开发商能够独立分工发展，为构建大规模、可持续的具身智能产业生态提供基础支撑。白皮书将系统性阐述 EAIOS 的整体架构、关键设计理念与接口规范。相关参考实现 Robonix 将以开源形式发布。

关键字：具身智能、操作系统、人工智能、智能体、 ROS、VLA、世界模型、泛在操作系统

# 1 引言

## 1.1 具身智能有望形成新的万亿产业

具身智能是新一轮科技革命和产业变革的最新前沿与制高点[1]。2010 年代以来，随着以 Transformer 为代表的深度学习技术取得突破， 以及 GPT 等大语言模型的迅速发展， 人工智能正加速从数字世界向物理世界渗透。这一趋势不仅推动了 AI 深度融入人类社会，更深刻重塑了全球制造业与经济格局。自 2023 年起，具身智能（Embodied AI）产业强势崛起， 展现出广阔的应用前景，有望对经济社会多领域产生变革性影响。

何为具身智能？本质上， 它是指拥有物理硬件“身体”与人工智能软件“大脑”，并能与现实物理环境交互以完成复杂任务的智能系统。智能机器人是具身智能最典型的实现形态，自动驾驶汽车及自主无人飞行器也可被视为具身智能， 而人形智能机器人因具备拟人化设计，在替代人类从事危险或繁重工作方面潜力巨大，因而备受业界关注。

从全球视角来看，具身智能产业目前尚处起步阶段， 但发展势头迅猛。多家机构预测， 人形机器人有望在 2030 年实现规模化生产；全球具身智能市场规模预计将于 2030 年突破 1500 亿美元，并于 2035 年达到 4000 亿美元[1]。全球主要经济体均已展开战略布局： 美国侧重于国防与太空领域的投入；欧洲聚焦于医疗等领域的机器人技术应用； 日本则致力于将机器人融入社会基础设施。与此同时， 英伟达、谷歌、 AMD、Intel、OpenAI、特斯拉等科技巨头也纷纷入局，竞相构建软硬件生态以抢占先机。

我国政府高度重视具身智能发展。2025 年 3 月，十四届全国人大三次会议《政府工作报告》明确提出因地制宜发展新质生产力， 首次将“具身智能”和“智能机器人”写入报告，强调培育相关未来产业及大力发展新一代智能终端。同年 10 月，党的二十届四中全会通过的《中共中央关于制定国民经济和社会发展第十五个五年规划的建议》进一步提出，要前瞻布局未来产业，推动量子科技、脑机接口、 具身智能等成为新的经济增长点。

在产业竞争力方面，我国已跻身全球第一梯队，具备完整的产业链与明显的成本优势。各地政府纷纷积极布局， 推动产业聚集发展。北京以“国家地方共建具身智能机器人创新中心”为引领，集聚企业 300 余家，产值超百亿；上海发布实施方案，计划到 2027 年推动核心产业规模突破 500 亿元；粤港澳大湾区是全球机器人供应链重镇（占据约 24% 份额），深圳已宣布新增 45 亿元专项投资，聚焦人形机器人及核心零部件等领域； 江苏苏州瞄准千亿级产业集群，浙江、安徽、湖北、四川等地也纷纷成立省级创新中心， 推动技术联合攻关。宇树、优必选、智元、云深处、银河通用、优宝特等一批明星具身智能企业涌现，产品已开始服务于经济社会的各个方面。

在人才培养与科研创新方面， 国家战略需求正在加速落地。 2025 年 11 月，教育部公示了北京航空航天大学、北京理工大学等 7 所高校增设“具身智能”新专业的申请。该专业融合了

人工智能、机器人学、控制工程等多学科， 旨在培养复合型人才。同期， 清华大学成立具身智能与机器人研究院，致力于突破“强健本体 + 智慧大脑”的全栈技术原始创新。

展望未来，国务院发展研究中心发布的《中国发展报告 2025》预测，中国具身智能产业规模有望在 2030 年达到 4000 亿元，并在 2035 年突破万亿元大关[1]。报告同时强调，中国具身智能产业仍需在关键共性技术上集中发力，特别是要支持研发安全可控的具身智能操作系统，并鼓励其开源发展，以夯实产业底座。

## 1.2 目前具身智能缺乏能支撑万亿产业生态的共性关键基础设施

具身智能若要成长为万亿级规模的产业，核心前提是形成软件与硬件、上游与下游均能独立发展的成熟生态。而这种生态的形成，离不开业界公认的共性关键基础设施支撑，其中最核心的便是连接与解耦软件生态与硬件生态的操作系统。然而， 当前的具身智能技术仍处于起步阶段，尚缺乏此类关键基础设施。

回顾 20 世纪计算机产业的发展， 其历程极具启示意义。早期的计算机是高度专用化的设备，程序员被迫直接面对“裸机”编程，软件必须为特定硬件量身定制。这种模式导致了严重的后果：

- 软硬高度耦合：更换机器往往意味着软件重写；
- 开发效率低下：程序员必须深刻理解底层硬件的每一个细节；
- 难以复用：无法形成通用的软件产品，阻碍了产业规模化。

随着 Unix 和 Windows 等操作系统的出现， 产业逻辑发生了根本性变革。操作系统构建了一个关键的 “抽象层”：

- 向上（应用层）：提供了一套稳定、统一的虚拟机器抽象接口（系统调用 API）。应用开发者只需遵循这套接口编程，即可让软件运行在所有兼容的硬件上，无需关心底层细节；
- 向下（硬件层）：提供了一套标准的驱动框架与接口，统一管理硬件资源，屏蔽了底层驱动与调度的复杂性。

正是得益于这一“抽象层”，硬件与软件实现了正式解耦， 从而催生了独立的、爆发式增长的软件产业与硬件产业，确立了专业化分工的庞大生态。

21 世纪移动互联网产业的爆发也遵循了同样的逻辑。苹果 iOS 和谷歌 Android 操作系统的出现，成为了产业腾飞的关键使能因素。这两大操作系统成功实现了智能手机软硬件生态的解耦与独立发展，造就了继个人计算机之后的又一个信息技术产业高峰。

当下的具身智能产业， 正处于类似于早期计算机产业的“前操作系统时代”。让机器人完成人类指定的工作极具挑战性：开发者面对的是复杂的全栈技术体系，必须针对特定的机器人硬件进行开发。无论是通过规则编程，还是利用 AI 大模型训练，为了让机器人完成一个动作或掌握一项技能，往往需要付出巨大的定制化成本。这种工作模式缺乏复用性——一旦物理环境、任务细节甚至硬件状态发生微小变化，原有的程序或模型便可能失效。

造成这一困境的根本原因，在于目前缺乏能够将具身智能硬件与软件解耦的共性关键基础设施，即缺乏一个提供新型抽象层的具身智能操作系统。

面向未来，具身智能产业要突破当前的“作坊式”开发瓶颈，必须依赖以具身智能操作系统为代表的共性关键基础设施的突破与普及。

## 1.3 白皮书的工作与意义

白皮书立足于操作系统的视角，对当前的具身智能技术体系进行了系统性梳理，并深入剖析了现有技术路径所面临的“硬件难适配”与“软件难复用”两大核心挑战。针对上述挑战，白皮书提出了一种面向具身智能的新型操作系统（EAIOS）设计架构。该系统通过构建“原语—服务—技能—任务”的四层抽象体系，重构了具身智能的运行基座： 底层对硬件进行“原语”级抽象，中间层提供感知、控制、认知等标准化“服务”，上层支持“技能”的开发与“任务”的调度运行。特别地， 系统支持将成功执行的任务转化为可复用的“技能”，实现了技能库的动态积累与演进，使得具身智能体的智能水平能够随着使用过程而持续增强。

通过系统层面的统一抽象，EAIOS 有效实现了具身智能硬件与软件的解耦。这使得大模型提供者、技能开发者、任务开发者以及硬件制造商能够专注于各自领域的独立发展与迭代， 相关制品可以复用， 为构建大规模具身智能产业生态所需的共性关键基础设施提供了设计依据。

白皮书详细阐述了具身智能操作系统的设计原则、核心概念、体系架构及标准接口定义。此外，相关的系统原型实现 **Robonix** 将以开源形式发布于 Gitlink[1] 与 Github[2]，以促进社区的共同探索与发展。

[1] https://www.gitlink.org.cn/syswonder/robonix

[2] https://github.com/syswonder/robonix

# 2 具身智能系统技术现状

## 2.1 具身智能的定义与范式

### 2.1.1 具身智能的机理与内涵

具身智能（Embodied AI） 是基于具身认知（Embodied Cognition） 理念发展而来的一种人工智能范式，认为智能产生于物理身体（Body） 与环境（Environment） 的交互过程。具身智能代表了对智能本质的一种认知。回顾人工智能的早期发展历程，主要经历了符号主义（Symbolism）[2]与连接主义（Connectionism）[3]两个阶段。前者认为智能依赖于显式的知识表征与逻辑推理，其典型代表为早期的专家系统；后者则以神经网络为核心，主张通过从数据中学习特征表示来逼近复杂分布，从而习得智能。这两种范式均隐含了共同的前提假设，即智能可以作为一种独立的信息处理过程，在脱离物理实体与环境的抽象空间中实现。这种将认知过程与物理载体剥离的观点，其哲学基础可追溯至笛卡尔的身心二元论。

相对而言，具身认知理论主张身体结构与环境交互是智能产生的物质基础。Varela 等人[4]指出，认知活动在根本上依赖于智能体具体的身体形态及其与环境的动态交互；知识并非预先存在的抽象符号系统，而是在感知与行动的闭环中被不断建构与更新的产物。在此基础上，Brooks 在其著作《Intelligence without Representation》中提出了“智能无需表征”的论断[5]，强调智能直接源于物理本体与环境的交互行为， 而非依赖于复杂的内部符号模型。 Prem 则在论文标题中正式引入“具身人工智能”（Embodied AI）这一术语，为该领域构建了初步的认识论框架[6]。

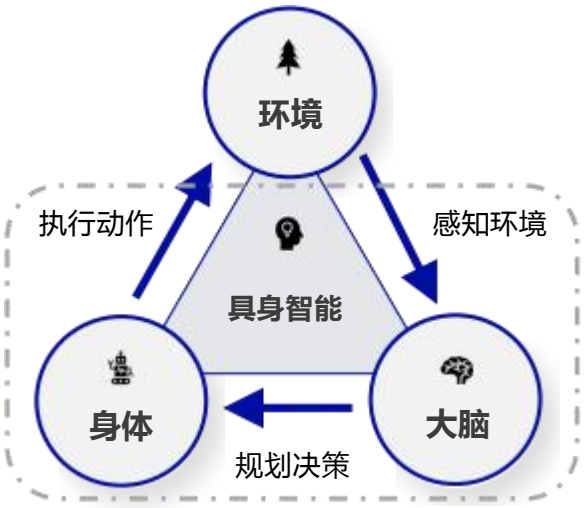


图 1: 具身智能核心要素

随后的学术讨论[7]进一步深化了这一概念，将具身智能视为一个分布在“大脑 - 身体 - 环境”三者之间的动态系统， 而非仅仅局限于孤立的算法或模型层面， 其系统构成关系如图1所

示。

需说明的是，具身智能在本质上区别于仅存在于网络虚拟空间（Cyberspace）中的“离身智能”（Disembodied AI），例如 ChatGPT[8] 等纯软件智能体； 同时也显著不同于传统工业场景中仅能依照预设指令执行固定动作序列的自动化工业机器人或机械臂。具身智能系统的核心特征在于其物理实体与智能系统的有机统一：它必须同时具备物理形态的“身体”与具备认知能力的“大脑”，从而能够在非结构化的物理环境中执行多样化的通用任务。也有观点认为传统工业机器人可被视为具身智能的初级形态。

自 20 世纪 90 年代至 2020 年代初，具身智能领域处于相对缓慢的探索期。这一时期的代表性成果是本田公司于 2000 年推出的人形双足机器人 ASIMO[9]。经过持续的迭代升级，2011 年发布的型号已具备避障、手语交流等高级功能， 至 2014 年更能实现踢球、倒水等复杂的灵巧操作。然而， ASIMO 的控制逻辑主要依赖于复杂的预设规则编程， 导致其研发成本极高， 且缺乏应对未知场景的泛化能力， 难以执行非预设的通用任务。受限于技术范式的瓶颈， ASIMO 难以实际应用，该项目最终于 2019 年终止。

进入 2020 年代，随着深度学习、强化学习以及多模态感知等人工智能技术的突破性进展，具身智能迎来了飞跃式发展的新阶段。特别是通过集成 GPT、Qwen 和 DeepSeek 等大语言模型（LLM），具身智能体在物理环境中的语义理解、逻辑推理及长程规划能力得到了显著提升，标志着具身智能从“基于规则”向“基于学习”和“通用智能”的范式跃迁，具身智能发展迈入新的阶段。

### 2.1.2 泛在计算视角的具身智能

从泛在计算的视角， 具身智能是人工智能技术与物理实体深度融合的典型泛在计算场景。

当前，随着互联网向人类社会和物理世界的全方位延伸， 一个万物互联的“人机物”（人类社会、信息空间和物理世界） 融合泛在计算（Ubiquitous Computing）的时代正在开启。面向未来人机物融合泛在计算的新模式和新场景， 软件定义一切、万物均需互联、 一切皆可编程、人机物自然交互将是其基本特征。除传统计算设备（“机”）和新兴物联设备（“物”）外，“人”作为一种新的重要元素的参与，构成了极其复杂且动态多变的计算环境。所谓泛在计算，是指计算无缝融入物理环境，无处不在又无迹可寻[10]。

泛在计算的环境多变、需求多样、场景复杂， 要求硬件资源、数据资源、软件平台、应用软件具有柔性灵活的软件定义能力、动态适配能力、泛在互联能力和自然交互能力，这对目前主流操作系统提出了严峻挑战。当千亿规模的各类泛在物联终端和新型泛在计算模式不断出现，意味着新型操作系统出现的条件已然具备，这类新型操作系统被称为“泛在操作系统（Ubiquitous Operating System，UOS）”[11]。由于泛在计算场景的领域行业特定性、泛在计算资源的广谱多样性和极端特异性，泛在操作系统的领域性和专用性将会比较突出，有必要面向

不同的应用模式和场景构建不同的泛在操作系统，这些泛在操作系统秉承泛在计算的基本思想，具有泛在感知、泛在互联、轻量计算、轻量认知、反馈控制、自然交互等新特征， 重点关注“感、联、知、控”的共性框架凝练，支持新型计算模式下的泛在应用开发与运行支撑。

具身智能系统实现了人工智能算法与物理硬件的深度耦合，其计算过程无缝融入了本体与环境的感知、认知及控制闭环中， 是典型的泛在计算范式。基于泛在计算与泛在操作系统的理论框架，具身智能的发展亟需一种面向该领域特性的新型泛在操作系统提供底层支撑，通过屏蔽底层异构硬件的复杂性， 为上层具身智能应用与任务提供标准化的“感、联、知、控”共性服务。

## 2.2 具身智能系统

具身智能强调智能体必须依托物理实体，通过持续运行“感知—规划决策—执行”的闭环，与真实物理环境进行动态交互， 进而实现智能的演进与适应。为了将这一理念推向工程落地，需要构建一套能够承载具身智能运行机制的系统， 即具身智能系统（**Embodied Artificial Intelligence System**）。从系统构成来看，具身智能系统主要由具身智能硬件（作为系统的物理载体，即“身体”）与具身智能软件（作为系统的认知与控制核心， 即“大脑”与“小脑”，简称“大小脑”）两部分组成。

具身智能硬件为智能体提供物理载体与运动执行能力， 是其与真实世界进行交互的基础。其核心组成包括多模态传感器、运动控制器、边缘计算设备、移动底盘以及机械臂等关键模块。其中， 多模态传感器负责采集视觉、力觉、位姿等环境状态数据； 运动控制器执行电机的实时伺服控制；计算设备承担本地感知计算与决策推理任务；移动底盘赋予系统自主移动能力；而机械臂则用于执行高精度的操作与物理交互任务。移动底盘与机械臂等共同构成了具身智能的身体形态，身体形态决定了系统与外界环境的交互方式。需要说明的是，机器人学通常使用“本体”这一术语来描述机器人的机械载体，侧重于其物理属性与工程本质；而具身智能则从认知科学的视角出发， 强调“身体”在塑造智能过程中的基础性作用， 与“心智”或“大脑”相对应。为行文方便， 本文在后续论述中不严格区分“本体”与“身体”，视两者为同一概念。

具身智能软件为智能体赋予了感知理解、规划决策与任务执行等能力。典型的具身智能软件包括机器人中间件、智能模型与算法等。其中， 机器人中间件（如 ROS 2[12]、dora-rs[13] 等）主要负责系统级通信与协同； 智能模型与算法（如 GPT[14]、RT-2[15]、RoboBrain2.0[16] 等）则主要实现高层语义理解、复杂任务的逻辑推理以及动作序列的生成等。

在实际运行中，具身智能的软硬件系统必须实现高度的一体化协同。各功能模块之间存在紧密的依赖关系与强耦合特性，图2展示了一种具身智能软硬件系统的详细模块组成关系。

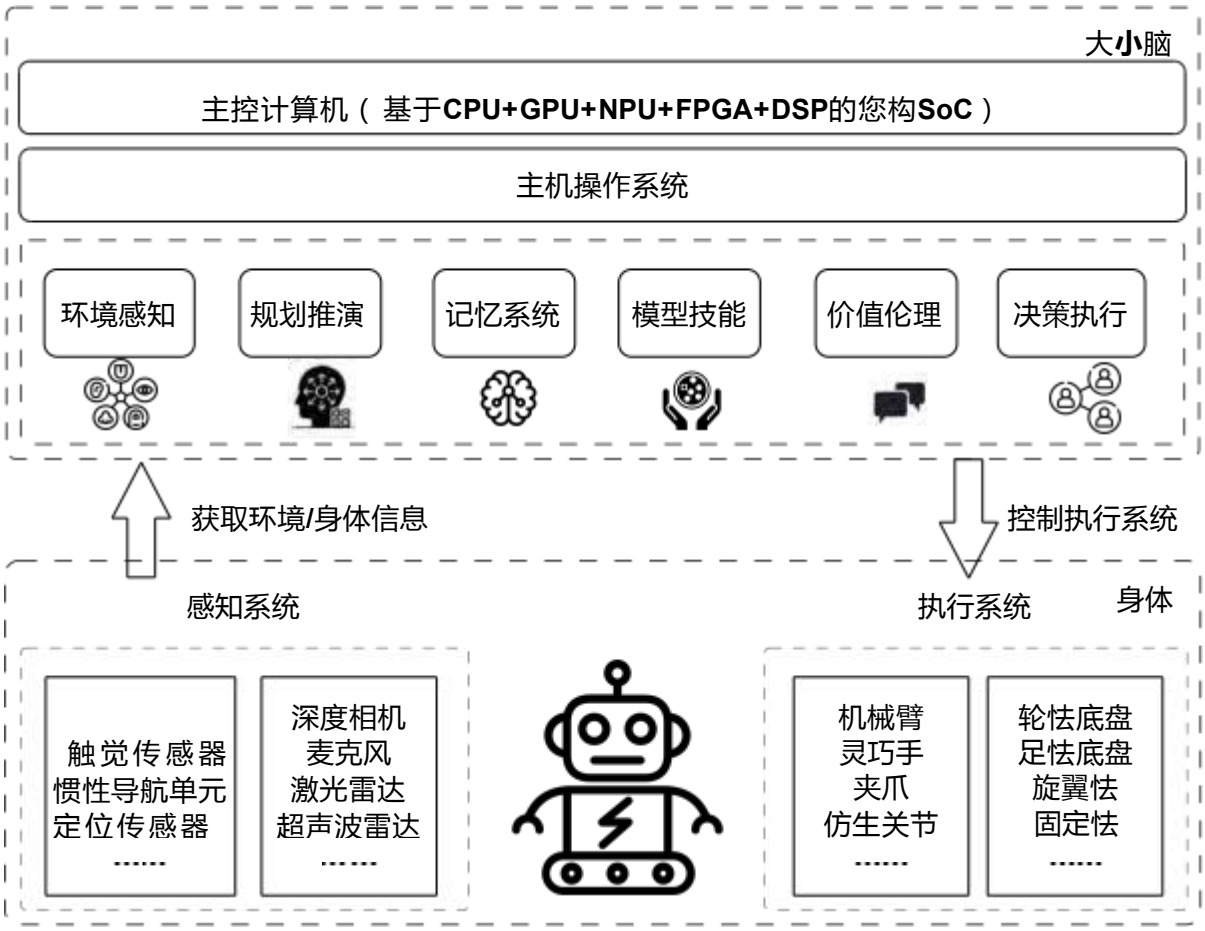


图 2: 具身智能系统构成概要

### 2.2.1 具身智能硬件系统（身体）

具身智能系统的硬件形态呈现出显著的多样性，涵盖轮式机器人、四足机器人及人形机器人等多种构型。尽管不同形态在运动学结构、传感器配置及控制拓扑上存在差异，但在系统架构层面，其基本控制体系普遍遵循分层设计的原则，主要由主控计算平台、中间控制模块以及关节驱动器构成。本节以四足机器人与人形机器人为例， 阐述其典型的硬件架构设计。

**1.** 四足机器人硬件架构

典型四足机器人的硬件架构如图3所示。该类系统通常由机身载体、腿部执行机构、多模态感知模组、运动控制主机以及关节驱动单元构成。在执行层面， 腿部包含多个关节自由度，每个关节由独立的电机与微控制器单元（Microcontroller Unit，MCU）驱动，内部运行实时操作系统（Real- Time Operating System, RTOS） 以保障高速伺服控制的实时性。在感知层面，机身集成了惯性测量单元（Inertial Measurement Unit，IMU）、深度相机等传感器， 用于高精度的状态估计与环境感知。运动控制主机通常运行 Linux 或 Windows 操作系统，负责感知计算与运动控制。主机通过 CAN/CAN- FD 总线[17]等工业级通信协议与关节驱动节点互联。CAN/CAN- FD是一类广泛应用的工业实时通信协议，因其高可靠性与低时延特性，被广泛应用于关节状态反馈与控制指令下发。

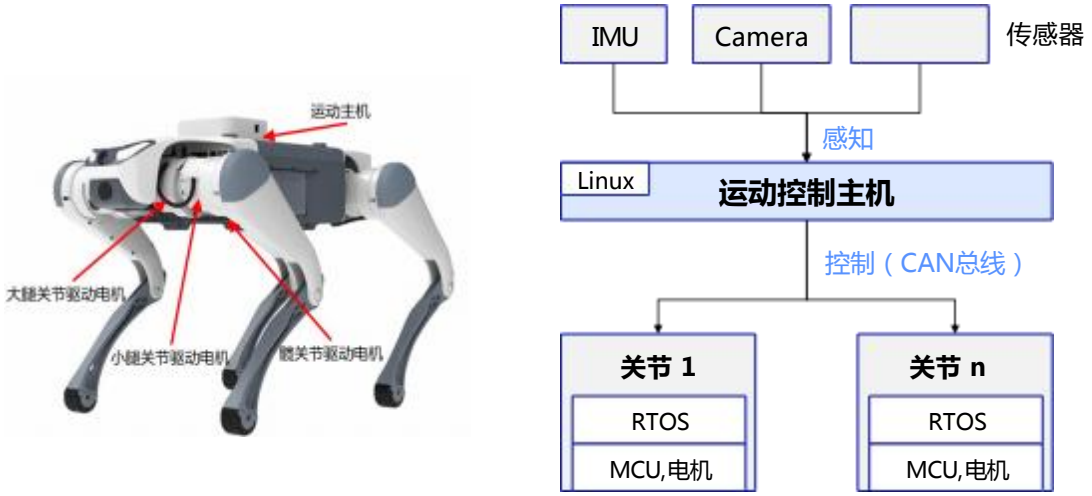


图 3: 一种四足机器人的硬件架构

**2.** 人形机器人硬件架构

图4展示了一种典型的人形机器人硬件架构。由于人形机器人具有高自由度（DoF）且关节分布密集的特征，其控制体系通常采用分层架构设计：

- 最上层为主控计算机（通常运行 Linux 或 Windows），承担多模态感知、任务规划与全身动作协调等任务。
- 中间层包含上身与下身控制模块，负责区域级的动作管理与协调。

- 底层为运行 RTOS 的关节驱动器，负责执行局部的微秒级高速闭环伺服控制。

主控平台与中间控制模块之间通常采用 EtherCAT[18] 等工业以太网协议进行通信， 以保证大数据量传输的实时性； 中间模块与关节驱动器之间则多采用 CAN/CAN- FD 总线进行通信。此外，为了应对复杂的信号处理需求并支持功能扩展，现代人形机器人常采用多计算单元协同的架构，除主控计算机外，还可能集成用于信号处理的高算力第二主机或支持用户定制开发的第三主机，从而构成复杂的多主机协同系统。

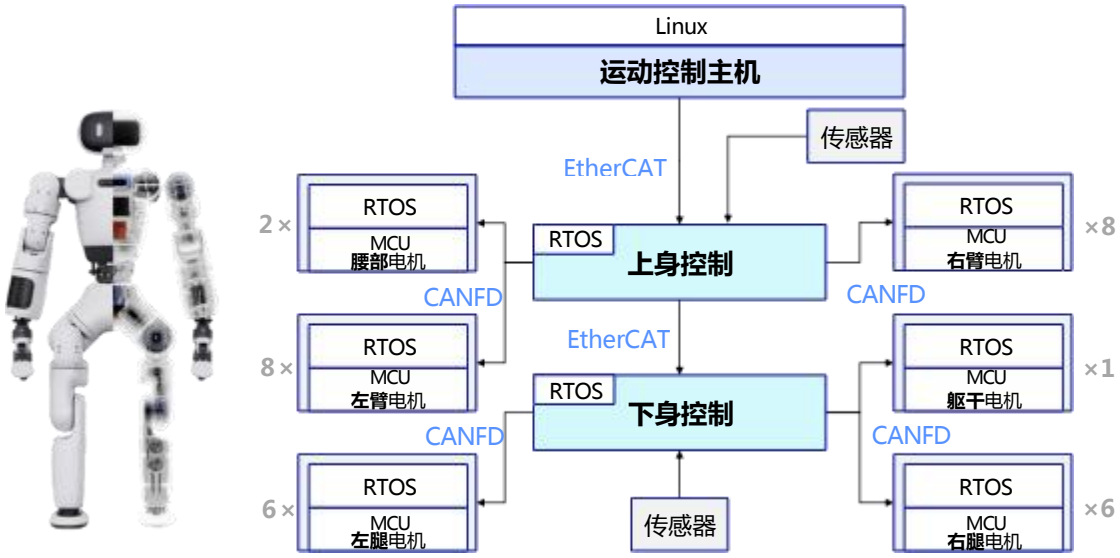


图 4: 一种人形机器人的硬件架构

### 2.2.2 具身智能软件系统（大小脑）

具身智能系统的核心目标是要赋予智能体在真实物理环境中自主执行多样化通用任务的能力。与仅运行于数字虚拟空间的传统人工智能系统（如 ChatGPT[8] 等）不同，具身智能系统在执行任务的同时，还必须能同时应对物理世界的不确定性、多样性以及突发状况， 因此，负责指挥物理硬件协同工作的具身智能软件系统（即“大小脑”）如何设计和如何运行至关重要。

具身智能软件系统需解决如下问题：

- 如何构建具备高鲁棒性的具身“大小脑”，以在复杂物理环境的约束下，实现多模态环境感知、高层任务语义规划决策与底层实时运动控制的有机统一；
- 如何建立高效的“感知—规划决策—执行”闭环反馈机制，驱动物理本体与环境进行持续、有效的动态交互，从而确保各类目标任务的精准达成；
- 如何实现任务的高效定义与低成本构建（例如降低对大规模示教数据的依赖），并提升系统在开放世界中的自主执行能力与泛化适应性。

上述问题共同构成了当前具身智能系统架构设计与核心算法研究面临的主要挑战。

## 2.3 具身智能软件系统的实现路径

现有的具身智能软件系统（即具身智能“大小脑”）在实现架构上，通常采用应用层软件直接驱动底层硬件执行动作的模式。根据控制机理与实现范式的差异，其技术路径主要可划分为两大类：一是基于确定性模型（Deterministic Model）的传统机器人控制范式，该路径主要依赖于人工设计的规则与算法逻辑；二是基于概率性模型（Probabilistic Model）的大模型驱动范式，该路径由大模型直接推理输出控制指令，虽然显著降低了对显式规则编程的依赖，但对模型的训练数据规模与微调工艺提出了很高要求。

具体而言，确定性模型是指在既定的初始条件与控制输入下，系统的状态演化轨迹与输出结果具有唯一确定性。这一类别的典型代表包括经典的机器人运动学模型、动力学模型以及最优控制模型等。与之相对，概率性模型则将环境状态、感知不确定性及行为结果建模为随机变量，侧重于利用统计学习方法来刻画变量的分布特性，基于深度学习的感知、决策与动作生成模型是此类路径的典型代表。

### 2.3.1 基于确定性模型

工业机器人、仓储机器人以及早期的服务机器人系统普遍采用基于确定性模型的控制范式。此类系统建立在经典机器人运动学与现代控制理论基础之上，侧重于针对特定任务场景进行高度定制化的规则设计与算法编程，以确保在结构化环境中实现高稳定、高精度的自动化作业。在软件架构层面，此类系统广泛采用机器人操作系统（**ROS**）[12]作为核心中间件。如图5所示，其运行时的软件栈由底层 Linux 内核、中间层数据分发服务（Data Distribution Service, DDS）[19] 以及上层 ROS 客户端库共同构成，业务逻辑则以 ROS 节点（Node）的形式运行。

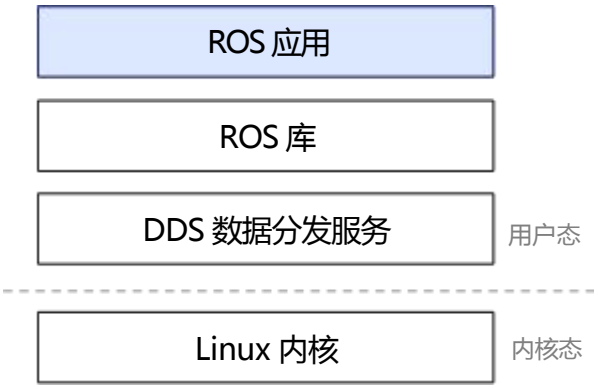


图 5: ROS 架构

在工程实践中，确定性模型在 ROS 框架内的实现路径主要包含以下三个维度：

**1.** 运动学与动力学建模

该维度通过统一机器人描述格式（URDF）[20] 与运动学与动力学库（KDL）[21] 等工具，对机

器人关节拓扑、连杆参数及动力学约束进行精确数学建模。这为上层的运动规划与控制组件提供了必要的解析求解基础。例如， MoveIt2[22] 框架即高度依赖这些确定性模型来执行逆运动学求解、碰撞检测与工作空间可达性分析。

**2.** 确定性规划

此类模型主要由 Nav2[23] 与 MoveIt2[22] 等规划框架支撑。Nav2 基于代价地图（Cost Map）实现全局路径搜索与局部轨迹优化， 广泛应用于移动底盘的自主导航； MoveIt2 则结合环境几何约束与逆运动学算法，生成满足避障要求的机械臂操作轨迹。

**3.** 确定性控制

底层的闭环控制通常由 ros2_control 框架[24] 管理。该框架定义了标准化的硬件抽象接口（Hardware Interface），支持开发者以插件形式加载各类控制器。具体的控制算法——如比例－积分－微分控制（PID）[25]、模型预测控制（MPC）[26] 等——封装于控制器插件中， 由 ros2_control 负责实时调度控制、处理传感器反馈并下发执行指令。

尽管基于确定性模型的系统在预设任务与结构化环境中表现出优异的稳定性、实时性与执行效率，但其局限性亦十分显著：

- 开发成本高昂： 系统功能实现高度依赖于专家知识与繁琐的人工规则编码， 导致开发周期长、工程投入大；
- 泛化能力缺失： 系统局限于执行预先定义的行为序列， 缺乏跨任务与跨场景的自适应能力；
- 软硬耦合严重：控制程序与特定的硬件构型深度绑定， 严重制约了算法的可移植性与软件复用效率；
- 语义理解匮乏：此类系统无法直接处理自然语言指令， 所有任务需求必须经由人工转化为精确的控制逻辑代码。

本田 ASIMO[9] 项目是这一路径的典型案例。尽管其具备卓越的运动性能，但由于单机造价极高（百万美元级），且所有复杂动作与交互行为均依赖精细化的规则编程， 导致后续的软件研发迭代与维护成本难以为继，该项目最终于 2019 年终止。

### 2.3.2 基于概率性模型

2023 年，Google DeepMind 发布的 RT- 2 (Robotics Transformer 2)[15] 标志着视觉- 语言- 动作模型（Vision- Language- Action Model，VLA） 的诞生。该模型创新性地将机器人的物理动作离散化为语言 Token，并与视觉及文本数据进行联合训练，从而实现了从自然语言指令及视觉输入到动作控制指令的端到端生成。这一突破显著增强了机器人的场景泛化能力与复杂任务

执行水平，有效降低了系统实现的复杂度。随着 RT_2 的问世，各类 VLA 大模型相继涌现（如图 6所示），推动具身智能技术迈入了崭新的发展阶段。

以 RT_2 为代表的一系列研究工作表明，利用大模型的跨模态感知、语义理解及逻辑推理能力，构建以概率性模型为核心的具身智能系统已成为可行路径。该范式的核心目标在于使机器人能够利用互联网级规模的海量数据习得“技能（Skill）”，进而直接输出动作指令以完成特定任务。这一数据驱动的路径彻底摒弃了对人工规则编程的依赖，构成了与基于确定性模型截然不同的技术范式。

从系统架构视角分析，当前基于概率性模型的具身智能系统已演化出两种代表性的技术架构范式：视觉- 语言- 动作模型（**VLA**）与加入长程复杂任务考虑的分层行动模型（**H- VLA**）。

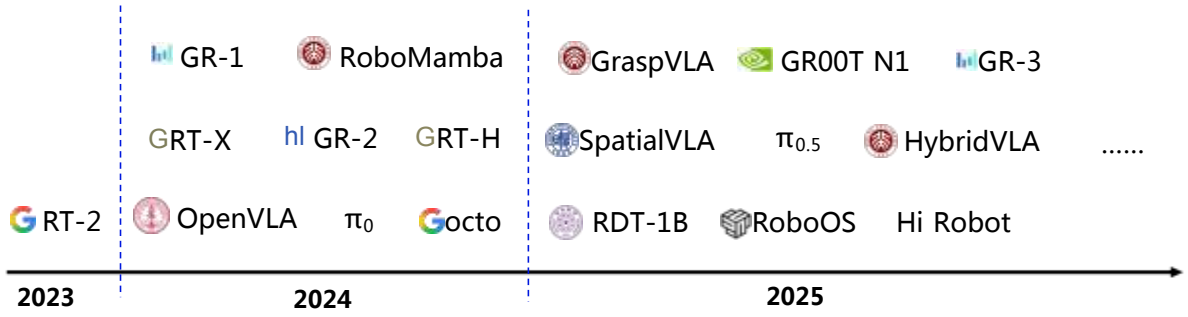


图 6: VLA 发展时间线

### 2.3.3 基于视觉- 语言- 动作模型 VLA


图 7: 视觉_语言_动作模型 VLA

视觉_语言_动作模型（Vision_Language_Action Model，VLA）代表了一种新兴的端到端控制范式，其典型代表包括 RT_2[15]、OpenVLA[27]、Octo[28]、RDT_1B[29] 以及 $\pi_0$[30]等。该类模型旨在统一的参数空间内实现“视觉特征—语言语义—低层动作控制”三种模态表征的对齐。通过输入多模态感知数据与自然语言指令，VLA 模型能够直接映射并输出可执行的低层级机器人动作序列，从而高效完成抓取、开关门、倒水等短周期的原子化操作任务。 VLA 模型一方面继承了视觉 - 语言模型（VLM）强大的开放域语义理解能力，使其能够解析高度非结构化的自然语言指令；另一方面，通过引入大规模“视觉—语言—行动”对齐数据，实现了感控一体化的端到端控制。

尽管 VLA 模型在短程技能泛化与语义交互方面展现出显著优势，但其局限性也很突出：

- 可解释性与安全性不足： VLA 模型本质上属于概率生成模型， 存在固有的“幻觉”风险。在物理交互中，不确定的输出可能导致执行错误甚至引发安全事故；
- 分布外（**OOD**）泛化能力受限： 模型性能高度依赖训练数据的分布特征， 对于长尾场景（如罕见物体、极端光照条件） 或涉及物理常识推理的任务（如判断“推开虚掩门所需的精确力度”），模型往往力不从心；
- 实时控制瓶颈：大参数量模型的推理延迟通常较高， 难以满足高频闭环控制对实时性的严苛要求；
- 长程规划与逻辑推理能力薄弱： 端到端模型倾向于解决单阶段、短时序的操作任务， 在处理涉及多目标协同、多步骤决策及长时序依赖的复杂任务时，其能力明显不足。

此外，尽管 VLA 范式规避了传统的人工规则编码，但要进行针对性的模型训练，工程代价非常高：

- 高质量数据获取成本高昂：模型性能高度依赖大规模且带有动作标注的示教数据，而物理世界的数据采集效率受限于机械运动与人工操作速度，效率低成本高。例如，Ra- dosavovic 等人的研究指出，采集 2 万条机械臂轨迹耗时达 9 个月（平均采集效率仅为3 条/小时）[31]；DexWild[32] 论文中亦提及，传统远程控制灵巧手的数据采集效率约为 43条/小时。
- 训练算力消耗巨大： 多模态大模型的预训练与微调严重依赖高端 GPU 算力集群。针对特定机器人平台或任务的单轮训练成本往往高达数千至数万美元（详见表 1）。
- 模型与硬件本体强耦合： 现有训练模式使得模型参数与特定的传感器配置、控制接口及动力学特性形成深度绑定。 一旦更换机器人本体或运行环境发生变化， 原有模型往往无法直接迁移， 必须重新采集数据并进行微调。这种“模型—本体”强绑定的特性， 严重制约了具身智能的规模化部署与应用复用。

表 1: 视觉- 语言- 动作模型 VLA 训练成本

| 模型 | 参数量 | GPU 小时 | 硬件规模 | 训练时间 | 成本[33] |
|---|---|---|---|---|---|
| OpenVLA[27] | 7B | 21500 | 64 × A100 | 14 天 | $30,000 |
| NORA- 1.5[34] | 1B | 960 | 1 × H100 | 5 天 | $2,000 |
| GR00T N1[35] | 2B | 50000 | 1024 × H100 | - | $100,000 |

### 2.3.4 基于分层行动模型 H- VLA

分层行动模型（Hierarchical VLA，本文简称为 H- VLA）是具有高层任务规划与低层动作执行两层结构的一种行动模型，其目标是赋予系统处理长时序、多阶段复杂任务的能力。实现该目标的途径主要是引入基于视觉－语言模型（VLM）的高层任务规划层， 负责任务行动方案。

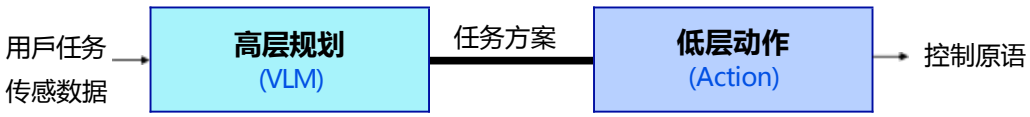


图 8: 分层行动模型 H- VLA

视觉－语言模型（**Vision－Language Model, VLM**）具备同时处理与理解图像及文本信息的能力，并能输出语义表征。其输出通常采用自然语言或结构化文本的形式，广泛应用于图像描述、物体识别及场景理解等任务。基于VLM 的分层行动模型（**H- VLA**）采用了显式的双层架构设计：

- 高层（任务规划层）：利用 VLM 大模型进行宏观的任务理解与路径规划，生成抽象的任务方案或子任务序列；
- 底层（动作执行层）：接收高层指令，利用传统控制算法或端到端 VLA 模型将其转化为具体的控制原语并执行。

该领域的代表性工作包括 RoboOS[36]、Hi Robot[37]、$\pi_{0.5}$[38]等。

分层行动模型 H- VLA 目前是具身智能系统实现复杂任务与工程落地的主流路线，在安全性方面，H- VLA 能够在规划层引入规则约束和安全过滤等机制提升系统动作的可控性， 而端到端 VLA 模型仅能依赖模型训练时的数据分布和训练方式来实现隐式而有限的安全约束；在可解释性方面，H- VLA 由于显式引入中间的任务方案表示， 使系统决策过程在任务级别上具有一定的可追溯性，相比之下，端到端VLA 模型缺乏显式的中间决策结构， 可解释性与 H- VLA 相比较差；在工程实现层面， H- VLA 通过分层设计提升了系统的可扩展性， 但相比单个端到端 VLA模型，其引入了多模块协同、接口设计与系统集成等额外复杂度。分层行动模型 H- VLA 有效提升了系统在长程多阶段复杂任务中的表现，但其仍存在显著局限：

- 训练成本叠加与泛化性受限： H- VLA 在底层动作执行上继承了端到端 VLA 模型的高昂训练成本与泛化难题， 此外，为了适配特定任务， 往往还需要对高层的VLM 规划模型进行额外的训练或微调，进一步增加了工程复杂度。

- 接口标准缺失导致的紧耦合： 目前，高层规划层生成的控制指令缺乏统一的规范与标准，不同系统间的动作接口定义差异显著。这使得高层规划与底层执行之间形成强耦合关系，严重阻碍了跨平台复用与生态构建。

### 2.3.5 引入世界模型（World Model）

世界模型（World Model） 旨在构建具身智能系统内部对物理世界及其演化规律的认知表征。本质上，它是描述在既定环境状态与动作输入条件下，环境状态与本体状态随时间推移而演化的动力学模型（通常基于神经网络构建）。根据 LeCun 的理论框架，世界模型在智能系统中主要承担两项核心功能：一是状态补全，即推断感知系统未能直接观测到的潜在环境状态；二是未来预测，即基于当前状态与动作序列，推演物理世界的未来演化轨迹[39]。图 9展示了世界模型在具身智能系统中的功能定位。

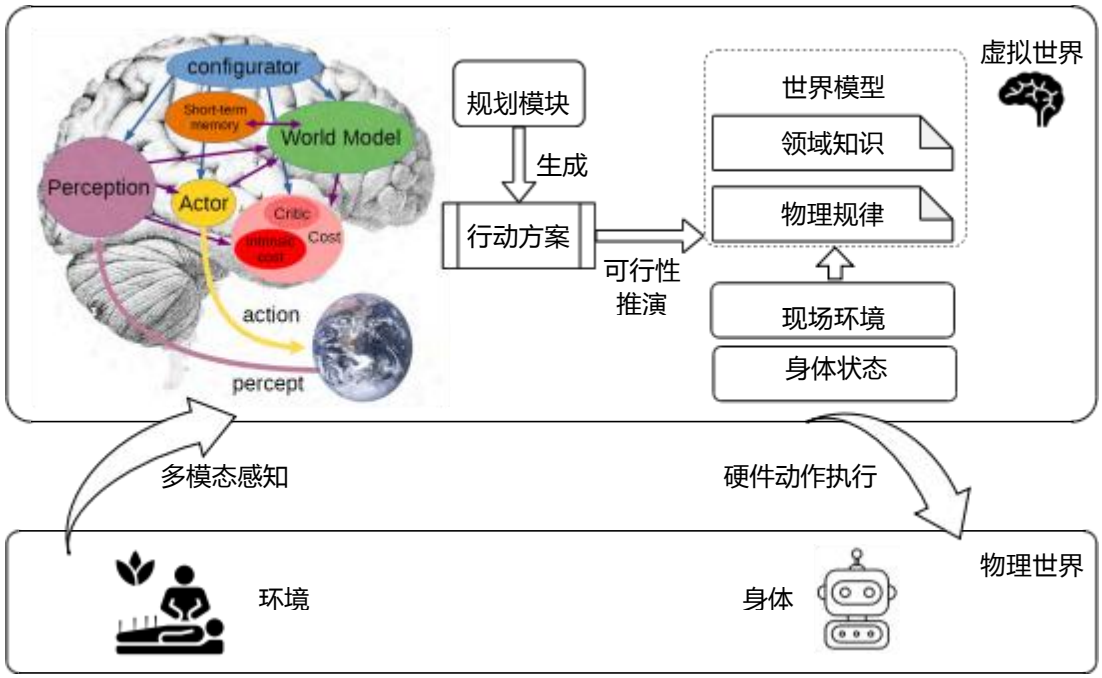


图 9: 世界模型

在具身智能系统中，世界模型的功能是提供一种服务于决策与行动的内部预测机制，而非通用智能模块，其核心价值在于为系统提供对未来状态演化的预测能力，使得智能体能够在执行前进行安全推演、风险评估与策略筛选。

当前，世界模型的主流技术路线可归纳为以下四类：

1. 潜在动力学与模型强化学习路线： 以 Dreamer 系列工作[40-42]为代表，侧重于在潜在空间中学习环境动力学并辅助策略优化；
2. 预测式表征学习路线： 以 JEPA 架构[43-45]为代表，强调通过预测被遮蔽的特征来学习抽象表征；

3. 视频生成与神经仿真器路线： 基于 Transformer 或扩散模型， 其中 Sora 等近期工作显著推动了关于“以视频生成表征未来预测能力”的学术探讨[46- 47]；

4. 空间智能与三维生成路线： 以 World Labs 的 Marble 等为代表，聚焦于从文本、图像或视频生成可编辑、可交互的三维世界， 被视为面向具身智能与虚实融合应用的新一代表征 - 生成式世界模型实践[48- 49]。

尽管世界模型的研究已取得初步进展， 但整体尚处于早期探索阶段，仍面临诸多严峻的技术挑战：模型的可控性与可验证性尚显不足；物理仿真精度与计算成本之间存在难以调和的结构性矛盾；此外，在跨本体迁移方面，仍缺乏统一的状态抽象机制与标准化的技能接口[50- 51]。

具身智能系统的核心关注点在于如何实现世界模型的有机集成，使其有效服务于各类具身智能任务。世界模型范式的引入， 推动了具身智能从单纯的“端到端动作生成”向“内部世界可预测与可验证”的认知层面跃迁。这为构建能够应对复杂长程任务的具身“大小脑”提供了更为系统的理论与工程支撑，因此，世界模型应成为具身智能系统的核心组成部分。

## 2.4 具身智能软件系统现状

当前具身智能软件系统的研发重点主要聚焦于感知、定位、规划与控制等核心功能模块的集成与优化。主流的代表性系统包括 NVIDIA Isaac ROS[52]、Intel Embodied Intelligence SDK[53]、OpenLoongOS[54]、AirOS[55] 以及 M- Robots OS[56]等。

在硬件厂商主导的生态中， NVIDIA Isaac ROS[52]（图 10）提供了涵盖视觉感知、三维建图及定位的 ROS 功能包，其核心优势在于与 Jetson 平台[57]及 CUDA[58] 等底层软硬件生态的深度耦合与优化。类似地， Intel Embodied Intelligence SDK[53]（图 11）集成了视觉处理、 SLAM、机械臂操作、路径规划及大模型接口等功能， 并通过对 Intel oneAPI[59] 与 RealSense[60] 传感器的原生支持，实现了内核级的配置优化与高效驱动适配。

OpenLoongOS[54] 构建了一种具身智能系统架构，涵盖驱动开发 SDK、中间件业务组件、控制算法库以及仿真环境。 AirOS[55] 与 M- Robots OS[56]均基于 OpenHarmony[61]构建，AirOS 通过在系统服务层引入任务管理、分布式通信、异构算力调度等组件，统一管理具身软件与硬件资源；M- Robots OS 采用实时内核与智能内核并行的双内核架构， 结合分布式通信机制和标准化功能组件， 为智能化框架构建和多机协作提供系统级支持。此外， insightOS[62] 作为面向人形机器人场景的专用系统，通过语义级任务管理与动态软硬整合机制，提供了支持端云协同的系统级服务能力。

总体而言，上述技术栈在视觉感知、运动控制、路径规划及人机交互等特定功能域提供了丰富的软件能力，但其本质上多属于基于现有中间件或特定硬件平台的工具集（Toolset） 整合与性能优化方案。当前主流方案在跨平台架构的通用性、异构硬件的统一抽象管理以及高

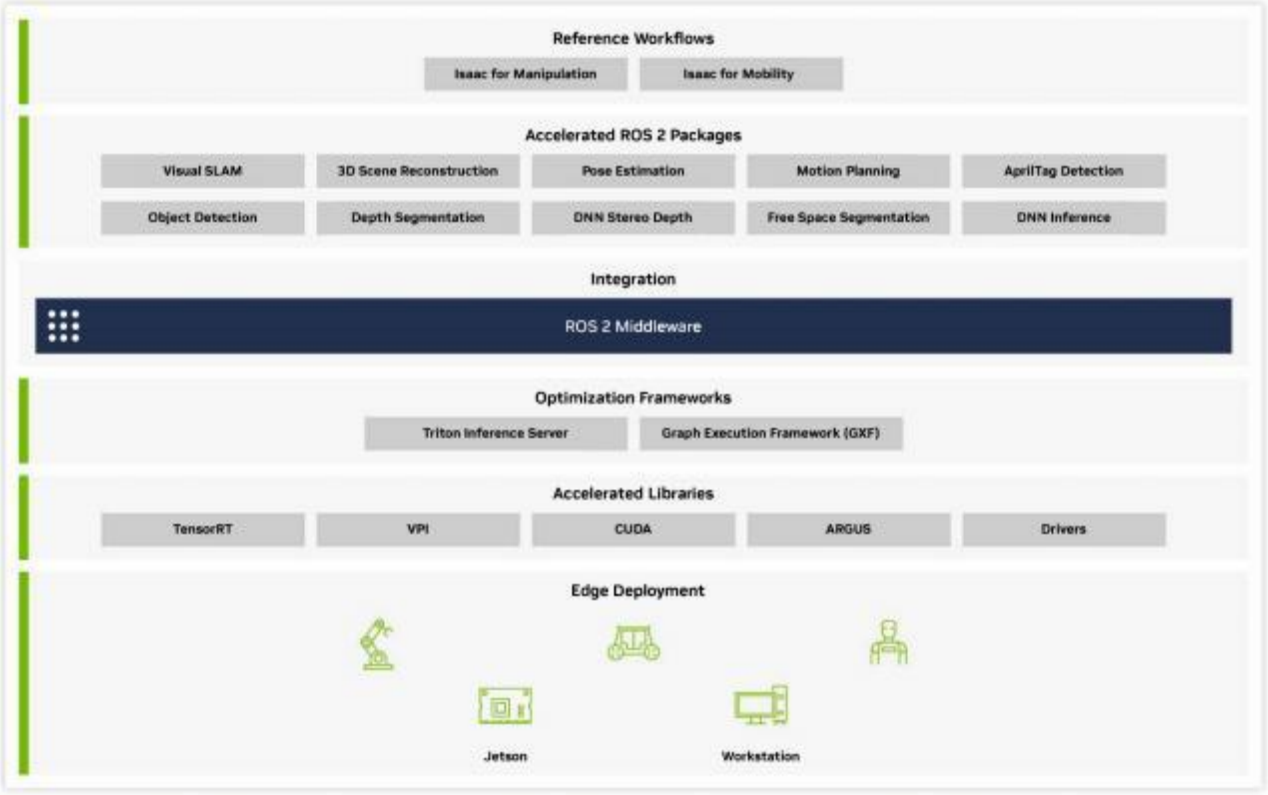


图 10: NVIDIA Isaac ROS 系统架构[52]

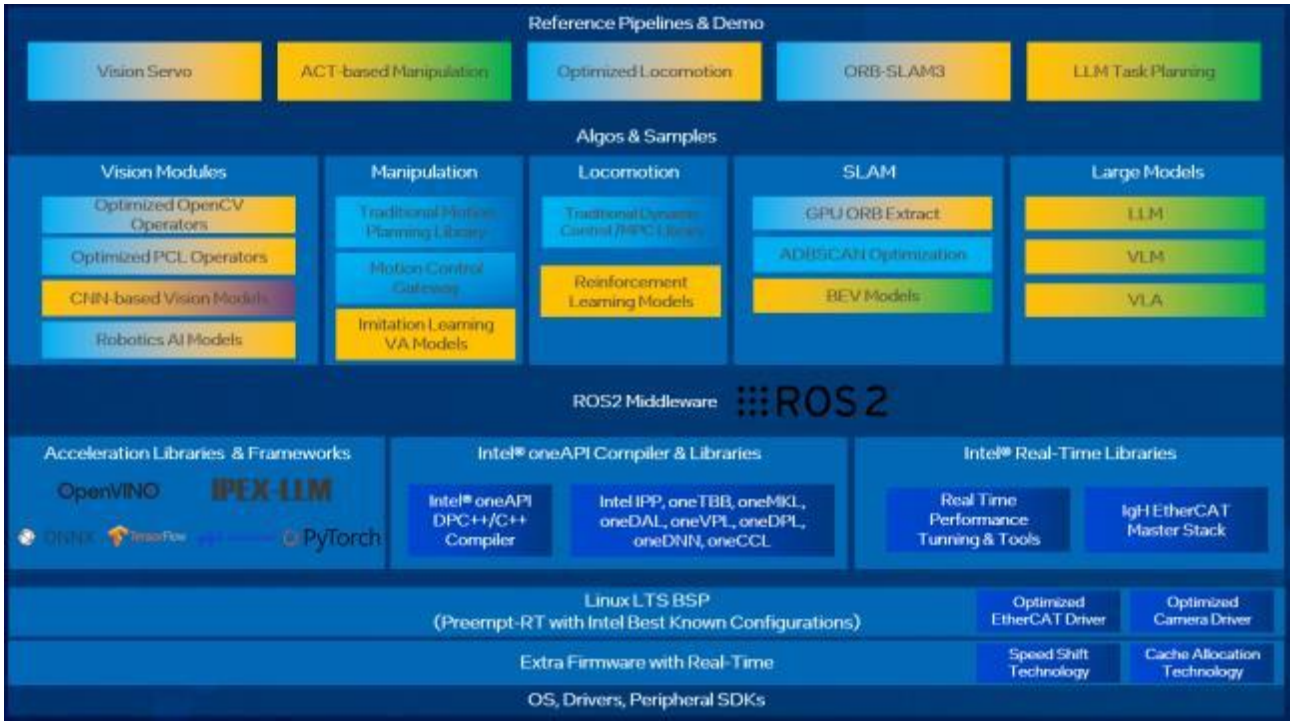


图 11: Intel Embodied Intelligence SDK 架构[53]

层技能复用等方面仍存在局限性，对于具身智能系统层面的标准化抽象与解耦设计尚处于探索阶段。

## 2.5 具身智能系统对新型操作系统的迫切需求

无论是采用端到端的 VLA 模型范式，还是基于分层架构的 H_VLA 行动范式， 当前具身智能系统在工程落地与规模化部署层面均面临显著的系统性约束， 表现为两个核心维度的挑战：

1. 硬件难适配：在端到端VLA 模型范式中，模型在训练阶段即与特定的机器人硬件本体（包括传感器参数、动力学特性及控制接口） 形成了深度耦合。这种强绑定关系导致模型在不同机器人平台间的迁移能力极低， 硬件适配成本高昂。即便模型在某一特定硬件平台上完成了训练，若需迁移至异构硬件平台， 通常需要重新采集大规模数据并进行全量微调或重训练。

2. 软件难复用： 即使是在分层行动模型 H_VLA 中，高层的语义规划模型与底层的动作执行模型之间也往往缺乏标准化的交互接口， 导致两者形成紧密的耦合关系。这使得高层规划算法与底层控制策略无法独立解耦、迭代与复用，严重阻碍了软件生态的形成。

除上述问题外，具身智能系统在运行时还面临若干挑战：

首先，概率模型的安全风险与验证需求。VLA/VLM 本质上属于概率生成模型， 存在固有的“幻觉”现象。在物理世界中， 模型的错误输出不仅是信息层面的偏差， 更可能导致严重的物理碰撞或安全事故。因此，系统亟需引入一种确定性的推演与评估机制，在动作执行前对模型的输出进行预测与验证。

其次，多任务并发与资源调度难题。在实际应用场景中，机器人需要处理高度复杂的动态任务流。例如， 在执行“前往指定地点配送”这一主任务的同时，机器人必须实时响应“电池电量过低”、“环境突发异常（如烧水壶沸腾）”或“用户临时插入的高优先级指令”等异步事件。现有的 VLA/VLM 模型难以独立承担此类涉及优先级管理与资源抢占的系统级调度任务，必须依赖一个中间层运行时环境（Runtime）来提供确定性的调度与管理服务。然而， 现有的具身智能软件栈缺乏足够的抽象能力来应对上述问题。

上述种种问题导致当前具身智能产业仍处于“作坊式”的初级发展阶段：开发模式主要表现为“模型即应用”的垂直整合形态，缺乏有效的产业分工。由此引发了机器人技能开发效率低、模型训练边际成本高、任务适应范围窄等一系列问题，严重制约了具身智能的大规模商业化落地。

从计算产业演进的视角来看， 当前具身智能的处境与计算机发展的早期阶段极为相似。在通用操作系统出现之前，程序员必须针对特定硬件编写“裸机”程序，导致生产效率极低且软件无法移植。随着通用操作系统的问世，硬件控制权被收归于标准的虚拟机器中间层，应

用程序不再直接依赖底层硬件。这一变革确立了软件与硬件的解耦机制，促使两者分离成为独立的产业并高速发展，最终推动了计算机从专用设备走向普及。

综上所述，具身智能产业的发展迫切需要构建属于自己的具身智能操作系统。通过建立统一的硬件抽象层并提供标准化的动作接口与确定性运行时服务，该操作系统将解决“硬件难适配”与“软件难复用”的结构性难题，实现具身智能“大脑”（认知模型）与“身体”（物理硬件）的彻底解耦。这将支持通用化具身任务与可复用技能的高效开发，促进模型、算法、硬件等各参与方形成独立发展的良性生态，为具身智能产业的壮大提供共性关键基础设施支撑。

# 3 具身智能操作系统的架构设计

## 3.1 设计目标与核心定义

为从系统工程层面解决具身智能在实现过程中的结构性约束问题，需引入操作系统来统筹管理具身智能系统的软硬件资源。遵循操作系统的核心设计思想，具身智能操作系统在运行时应满足以下关键技术要求：

- 异构硬件屏蔽与抽象： 建立统一的硬件抽象层， 屏蔽底层硬件实现的异构性及不同厂商 SDK 的接口差异；
- 标准化运行时支撑： 提供标准化的运行时环境（Runtime），实现对具身智能计算、感知及执行资源的统一调度与管理；
- 规划与执行解耦： 定义标准化的行动接口，实现高层任务规划与底层动作执行的解耦；
- 动态安全保障： 建立面向非结构化动态环境的安全机制，保障系统的可靠运行。

基于上述需求， 定义具身智能操作系统（**Embodied AI Operating System**，**EAIOS**）概念如下：

**具身智能操作系统（EAIOS）的定义**

具身智能操作系统是一种旨在统一具身智能硬件抽象， 定义标准化行动接口， 并提供感知、规划、决策、控制等运行时服务的系统软件。其核心目标是构建软硬件解耦的共性基础设施，为具身智能任务的开发、部署与运行提供支撑。

从泛在操作系统的视角，具身智能操作系统属于一类典型的面向具身场景的泛在操作系统。它需要构建满足具身智能特性的“感、联、知、控”应用框架，并提供领域特定的系统服务，如图 12所示。

具身智能操作系统通过以下四个维度构建其核心能力：

首先，通过构建硬件抽象层实现异构资源的统一接入。该层定义了“抽象机器人”概念，并将其能力映射为标准化的“感知原语”和“动作原语”。上层应用与智能模型通过调用这些“原语”与抽象机器人交互， 而非直接操作物理硬件， 从而实现了上层算法与底层硬件的有效解耦。

其次，通过技能（**Skill**）机制支撑任务执行。技能被定义为机器人完成特定子任务所需的、具有明确语义的可复用行动单元， 其地位类似于传统操作系统中的“库函数”。机器人可通过外部技能商店动态获取并加载各类技能，以适应多样化的任务需求。

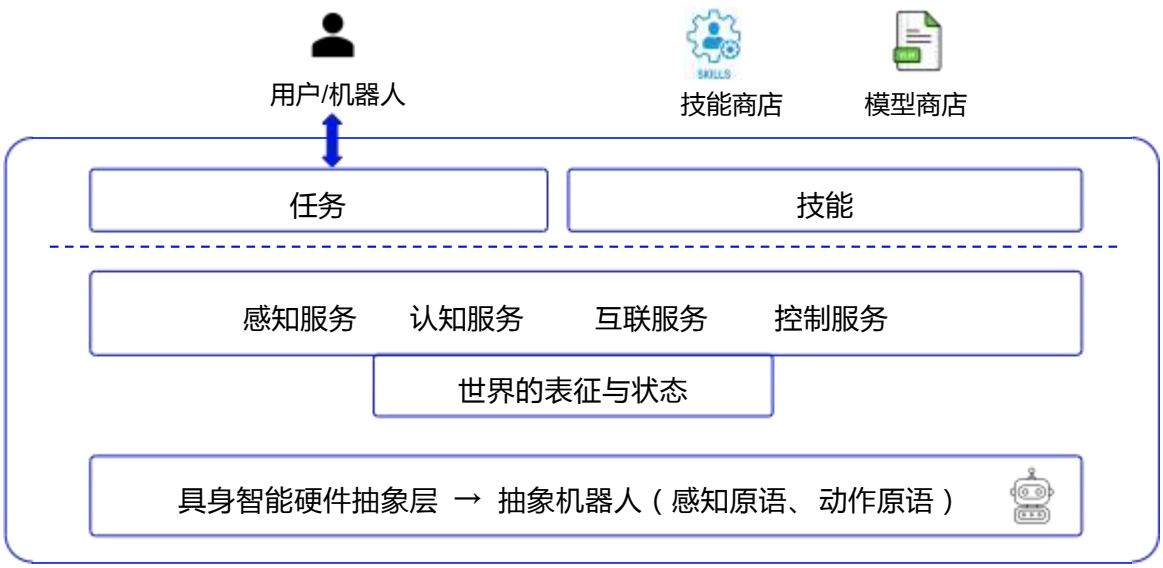


图 12: 具身智能操作系统概览

再次，提供具身智能特定的服务支撑。这涵盖了获取环境与本体状态的感知服务，执行分析、规划、推演及决策的认知服务，驱动本体执行的控制服务，以及支持多机协同与外部通信的互联服务。

最后，围绕“感知—规划决策—执行”闭环进行数据流管理。系统对多模态感知数据进行标准化抽象，并在内部维护物理世界与本体状态的表征，以此支撑复杂的认知活动与闭环控制。

图 13从数据流处理的视角展示了具身智能操作系统的逻辑架构。作为智能机器人的“大脑运行时”，其逻辑结构可划分为感知空间（Perception Space）、动作空间（Action Space）与认知空间（Cognition Space）三大核心域。

感知空间专注于环境与本体的数据获取与建模。其核心任务是处理多模态传感器数据，识别环境中的物体及其拓扑关系，构建映射外部物理世界的内部虚拟表征空间，建立包含几何信息与语义信息的空间地图，为认知决策提供数据基础。

认知空间是 EAIOS 的“大脑”，其侧重于对物理世界的理解、预测与决策。针对给定的用户任务，系统依据感知信息（如语义地图、本体状态）生成高层行动规划，并调用推演服务对规划方案进行可行性验证。在此过程中，认知空间维护包含物理规律与领域知识的“世界模型”，用于预测动作执行的未来状态；同时引入“价值伦理”与“安全规则”作为决策约束，确保行为符合人类价值观与安全边界。此外，该空间还管理 VLM 模型及多机协作所需的通信协议。

动作空间是 EAIOS 的“小脑”，其负责将高层决策转化为本体的精确控制指令。任务的执行依赖于机器人对“技能”的调用与组合。系统对已成功执行的任务进行评估与抽象，可将其转化为固化的技能存入技能库。在后续处理类似任务时，系统可直接检索并调用相应技能，规避重复的复杂规划与推演过程，从而形成类似于生物体“小脑本能反应”的高效执行机制。

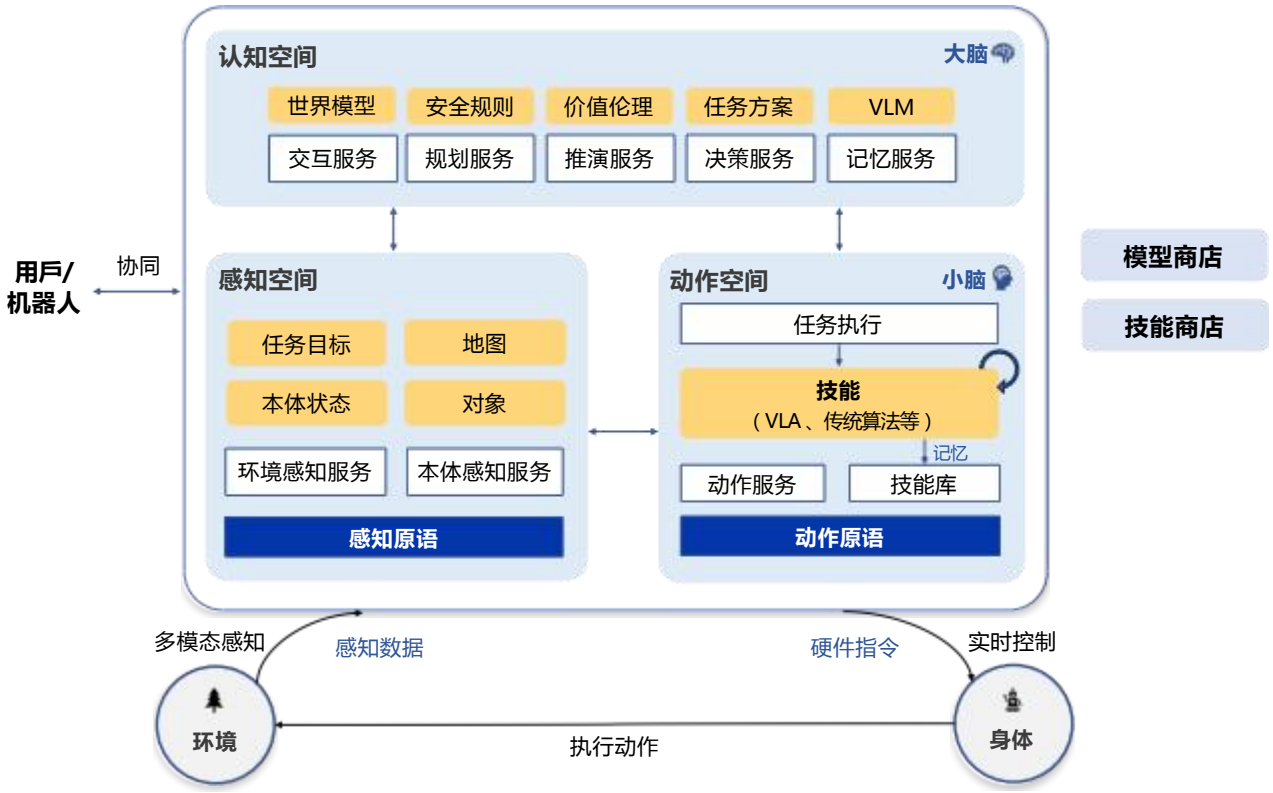


图 13: 具身智能操作系统的三个逻辑空间

动作空间（小脑） 并不参与任务级或策略级的智能决策，其职责限定于在既定技能调用框架下完成动作执行， 并在执行过程中保障本体运行的稳定性、自保护能力以及安全约束的落实。针对不同机器人本体在结构形态、传感器配置与安全指标上的差异， EAIOS 在系统层面能够定义动作执行与安全相关的接口规范， 由技能和动作原语提供者（如硬件厂商）提供具体实现。

## 3.2 核心概念与抽象体系

具身智能操作系统的抽象体系由机器人的行为抽象与对世界表征的抽象两个关键维度共同构成。

在行为抽象维度， 系统采用“任务—技能—服务—原语”的四层架构，实现了从高层语义目标到底层硬件控制的系统化分层建模，确立了具身智能系统从“用户意图解析”到“物理动作执行”的全链路运行框架。具体而言：

- 任务（**Task**）：用于结构化表达系统的顶层目标与用户意图；
- 技能（**Skill**）：用于封装具有特定语义且可复用的行为范式；
- 服务（**Service**）：用于承载系统内可调度的各类计算资源与认知功能；
- 原语（**Primitive**）：作为物理执行能力的最小抽象单元，实现了对底层异构硬件基础动

作能力的标准化映射。

在世界表征维度，系统引入“面向对象（Object- Oriented）”的建模范式，将物理环境中的各类实体统一抽象为“对象（Object）”，构成物理世界的虚拟映像， 以此构建支持计算与推演的数字世界表征。

### 3.2.1 任务层：意图与目标抽象

任务（**Task**）定义为由用户或系统自主提出的、具备明确目标状态与终止条件的高层工作请求。它是具身智能系统运行的顶层逻辑单元，通过对服务、技能及原语的编排与调用来实现。

典型的任务实例包括：“巡查所有房间并关闭窗户”、“前往厨房获取瓶装水”或“将桌面物体整理至收纳盒”。在执行流程上， 系统接收任务指令后， 首先调用任务规划服务将其分解为包含若干流程阶段的抽象执行方案，在 EAIOS 中使用任务描述语言（**Robot Task Description Language，RTDL**）对方案进行描述， RTDL 是 EAIOS 定义的统一任务表达规范，用于形式化描述单机或多机场景下技能、服务与原语的组合调用逻辑、时序执行顺序及资源依赖关系。在具体实现上， RTDL 并不局限于单一的表达形式， 可采用行为树（Behavior Tree）[63]、程序式代码结构（如 PLEXIL[64]）、有向图结构或其他形式化逻辑表示，是能够表达任务执行时对技能、服务、原语调用的形式化语言， 任务方案对应的 RTDL 随后交由任务管理器进行解释与调度执行。为确保执行的安全性与可行性，生成的规划方案需经由基于世界模型的方案推演服务进行模拟验证，并经决策服务评估通过后，方可进入物理执行阶段。

RTDL 是 EAIOS 在系统内部用于描述任务执行流程与调度逻辑的统一表示，用于刻画任务方案的编排结构， 在实际执行过程中， EAIOS 依据 RTDL 中描述的执行结构，调度相应的技能、服务或原语实例，其向下传递的具体输入形式由被调用技能的接口定义。因此，发往动作空间或技能层的控制信息可以采用多种表达方式，包括但不限于结构化参数、轨迹描述、图片数据，以及由高层模型生成的自然语言或中间向量等， EAIOS 通过技能调用接口来约束其语义边界。

### 3.2.2 技能层：可复用行为单元

技能（**Skill**）是指封装了特定语义功能的、可复用的行为操作序列。作为连接高层任务与底层原语的中间层，技能屏蔽了底层控制的复杂细节，使系统能够以模块化的方式组织复杂行为。具身智能系统通过构建技能库来积累与管理这些能力，随着技能库的扩充，系统的泛化适应能力将显著增强。

学术界已普遍采用“技能”作为机器人行为的语义基础。例如， Stanford 在2025 BEHAVIOR Challenge 赛事的数据集中，将复杂任务拆解为 31 类通用技能[65]，涵盖 chop（切）、close door

（关门）、hold（握持）、insert（插入）等原子化操作。DaDu- E[66] 进一步提出了“技能指令”（Skill Instruction）概念，构建了包含 navigate、pick、place 等指令的集合， 为任务规划中的动作表达提供了规范化的语义结构。

在 EAIOS 中，技能复用是指将能够稳定完成特定语义目标的行为执行能力， 以可调用的技能形式进行长期保留并在后续任务中重复使用； 具体而言，这种复用主要体现在两种形式：一是将预先训练完成的 VLA 或其他算法封装为技能长期使用，二是将任务执行过程中由系统动态生成且在特定场景下稳定成功的 RTDL 控制流程固化为技能，在后续相似任务中直接复用。

在 EAIOS 中，技能分为基本技能与 **RTDL** 技能。基本技能以独立执行单元的形式注册到系统中，例如 VLA 模型、ROS2 执行算法等， 是预先开发或训练得到、具有固定运行实现的静态技能；RTDL 技能则是在任务执行过程中由任务规划生成的控制流程描述，在运行时由 RTDL 解释器驱动，对系统中已注册的技能（包括基本技能与 RTDL 技能）进行调度与组合执行， 是动态生成的技能。当任务规划生成的 RTDL 控制流程在推演、决策与真实执行中被验证能够稳定完成目标时，系统可对其中具备独立语义边界与参数化条件的子流程进行抽取，并将其固化为新的 RTDL 技能。该过程由系统自动完成， 包括子流程边界识别、参数抽象与技能接口生成。例如， 在“取快递”任务中，整体 RTDL 流程包含“前往门口”、“与快递员对话”、“拿取快递”、“放回屋内”几个阶段， 其中“与快递员对话”的前后阶段可直接调用已有基本技能，而“与快递员对话”阶段由任务规划新生成的 RTDL 代码片段实现。该子流程在多次执行中表现稳定后，可被抽取为一个独立的 RTDL 技能，并由系统自动生成可配置技能参数（如目标人员、对话内容或场景约束），供后续任务复用。

典型的可复用技能包括： 导航至指定坐标、目标物体抓取、物体放置、门窗开闭、衣物叠放等。从实现角度看， 一个技能通常由若干服务与原语组合而成。例如，“关闭窗户”技能在内部需协同调用视觉感知服务（目标识别）、位置与姿态（简称位姿，位置通常采用相对某一坐标系的空间坐标 $(x, y, z)$表示， 姿态可使用欧拉角、四元数等方式表示） 估计服务以及机械臂运动控制原语；“导航”技能则依赖于空间地图服务与路径规划服务， 并最终由底盘运动原语执行。技能不仅可在不同任务中被重复调用，技能之间亦可进行嵌套组合，形成更复杂的行为逻辑。

技能层的复用主要服务于工程实现层面的能力积累与重复利用，在结构化程度较高、任务规则相对明确的场景中具有较好的实用性，例如标准化导航等任务。对于高度开放或强依赖即时感知与隐式经验的复杂任务，技能复用本身并不承担行为泛化的目标，其效果亦受限于具体的规划、动作模型或算法的能力。在 EAIOS 中，此类复杂任务通常通过认知空间中高层模型（如 VLM）提供的规划与决策泛化能力， 与动作空间中执行模型（如 VLA）在动作层面的泛化能力相结合完成，形成认知规划与动作执行协同的混合策略。系统本身不预设具体的泛化能力来源，而是通过统一的任务、技能与接口抽象，为不同模型能力的接入与协同提供

运行框架，具体的泛化效果取决于所采用模型的能力。

### 3.2.3 服务层：运行时功能组件

服务（**Service**）是指由操作系统统一注册、调度与管理的标准化功能模块，旨在为任务的决策与执行提供“感知—规划—推演—控制—反馈”的全链路能力支撑。开发者遵循统一的接口规范开发并接入服务，系统则在运行时根据任务需求动态完成服务的实例化与编排组合。

典型的服务包括：语义地图服务、空间地图服务、任务规划服务、任务推演服务、任务决策服务、数据采集服务、物体识别服务、校准服务、记忆服务、结果反馈服务等。

### 3.2.4 原语层：硬件抽象接口

原语（**Primitive**）构成了具身智能操作系统的硬件抽象层（HAL），旨在定义“抽象机器人”的标准化接口。通过原语机制，上层应用与模型仅需与抽象机器人进行交互，从而屏蔽了底层物理硬件的异构性与复杂性。

**1.** 感知原语（**Perception Primitive**）

感知原语是系统连接物理世界与机器人本体的数据入口，负责采集原始观测数据。其输出通常为未经高层语义加工的多模态数据流，为后续的感知建模与规划服务提供基础输入。

典型的感知原语包括：相机 RGB 彩色图像或 RGB-D（彩色与深度）图像流采集、惯性测量单元（Inertial Measurement Unit，IMU）数据读取、关节编码器状态反馈，以及力觉或触觉传感器数据获取等。

**2.** 动作原语（**Action Primitive**）

动作原语是对底层硬件执行能力的最小化抽象，定义为一组确定性的、标准化的执行指令，用于直接驱动底盘、机械臂、夹爪、相机云台等物理设备完成原子动作。动作原语通常对接硬件厂商提供的驱动程序，并由操作系统进行统一注册与权限管理。

典型的动作原语涵盖：底盘运动控制（如线速度与角速度设定）、机械臂关节空间控制（如关节角度设定）、末端执行器笛卡尔空间控制（如目标位姿设定）以及夹爪开合控制等。任何高层复杂行为最终均需被分解为动作原语的有序序列，才能在物理世界中落地执行。

针对 VLA 模型的接入，动作原语发挥了关键的代码解耦作用。面向机械臂的 VLA 模型通常输出末端执行器的位姿增量序列作为控制指令，单个动作步包含三维平移与旋转增量 $(\Delta x, \Delta y, \Delta z, \Delta r_x, \Delta r_y, \Delta r\approx)$ 及可选的夹爪状态[15,27,67-68]。然而，这种输出形式仍需要根据本体相关的驱动和 SDK 的具体实现，转化为具体的关节控制等指令，面向不同本体的工程适配具有一定工作量。

在本系统中，将“移动到目标位姿”定义为一种标准的动作原语，作为连接VLA 模型与底层控制 SDK 的中间层。模型仅需输出目标位姿，由具体本体的动作原语负责完成逆运动学求

解、轨迹插值与电机控制（不涉及到感知原语）。这种设计有效降低了模型与特定硬件 SDK 的耦合度，使得同一模型能够以统一接口对接不同厂家的机器人硬件，模型开发者无需为同一个模型反复适配不同本体，仅需要一次对接系统的原语抽象层即可，从而解决模型与本体的工程适配难题。

需要说明的是，动作原语主要缓解的是工程层面的接口与适配问题， 并未消除当前VLA 模型对具体机器人本体的隐含依赖，即便通过原语统一了控制接口，模型输出的目标位姿仍与训练时的相机参数、执行误差等因素相关，导致模型跨本体使用时仍需针对本体与任务进行微调才能正常工作[27]。如何提高 VLA 的跨本体、跨任务泛化性仍是一个正在探索的研究性问题，目前已有一些工作通过多本体数据集训练的方式来提升跨本体泛化能力（例如 RT- X[69]等）。

### 3.2.5 对象：内部世界表征的统一抽象

对象（**Object**）作为具身智能操作系统对“被感知、操作与推理的世界要素”的统一抽象，构成了系统内部数据组织与计算交互的基本单元。该模型遵循面向对象的设计范式，将现实环境中的物理实体、机器人本体、虚拟资源以及分布式的计算节点统一抽象为“数据 + 接口”的“对象”形态。在数据层面， 对象以结构化属性集的形式存在， 其典型属性定义如表2所示。

表 2: EAIOS 中对象数据的典型定义

| 数据字段 | 含义 | 示例 |
|---|---|---|
| 对象标识 | 系统全局唯一标识符（ID），用于运行时索引对象实例 | `obj_00042` |
| 名称标识 | 面向人类交互与高层语义规划的可读标签 | `"kitchen_window_1"` |
| 对象类型 | 描述对象所属的语义类别， 用于类型筛选与逻辑匹配 | `class=window` |
| 空间属性 | 对象在三维空间中的几何位姿描述 | `frame=map, pose=(x,y,z,q)` |
| 物理属性 | 描述对象的物理材质、质量、摩擦系数等特性 | `texture=wood` |
| 语义属性 | 描述对象的功能可供性（Affordance）等 | `affordance=grasp` |
| 能力描述 | 关联该对象所支持的操作集合（服务、技能或原语） | `supports=[skl :close, skl :open]` |
| 规则约束 | 定义与对象相关的系统级安全边界或用户级业务规则 | `forbidden_room=room1` |
| 对象状态 | 描述对象随时间演化的动态状态信息 | `motion_state=moving` |

基于对象间的关联关系， 系统构建了对象图（Object Graph），用于显式描述空间位置、拓扑结构及语义依赖（如“包含”、“邻接”、“支撑”、“机械连接”等）。对象图为高层的任务规划、约束条件推理以及技能的参数化选择提供了结构化的环境语义支撑。

对象接口定义了系统在运行时访问与更新对象状态的标准化规范。在 EAIOS 架构中，对

象以结构化状态的形式承载规划、决策与执行所需的关键信息； 接口则规定了各组件读取、订阅及变更这些状态的协议。借助统一的对象接口，感知、规划、推演与执行等异构服务能够在同一语义平面上共享对象状态，并执行查询与更新操作。典型对象接口的定义如表 3所示。

表 3: EAIOS 中对象接口的典型定义

| 接口类别 | 含义 | 示例 |
|---|---|---|
| 属性读取接口 | 提供对象属性的获取 | `get(id)` |
| 状态订阅接口 | 订阅对象状态变化事件 | `subscribe(id, "pose")` |
| 状态更新接口 | 对象数据的增加、删除、修改 | `update(id), add("box"), delete(id)` |
| 关系查询接口 | 提供在对象关系图上的关系查询 | `query(id, "inside")` |

## 3.3 感知空间：环境与本体的数字化重构

### 3.3.1 环境与地图构建

为了支撑自主定位、导航及交互任务， 系统必须实时维护关于外部环境的高保真描述。环境与地图服务（Environment/Map Service）常驻系统后台运行， 采用“双层地图”架构对物理世界进行数字化重建：

- 空间地图层（**Spatial Map**）：利用占据栅格、点云、网格/体素或距离场（SDF）等数学形式，精确表达环境的几何结构与可达性信息。该层主要服务于定位定姿、避障规划及碰撞检测，支持包含时间戳与置信度的局部/全局状态查询及增量式更新。
- 语义地图层（**Semantic Map**）：构建于空间地图之上，负责组织对象实例、语义类别及其可供性（Affordance，如可抓取、可放置、可开启等）。同时， 该层显式维护对象间的拓扑与语义关系（如“位于...上方”、“包含”、“物理支撑”、“空间相邻”等），为高层任务规划提供知识推理基础。

上述两层地图通过统一的数据服务接口对外提供查询与更新能力。典型的查询请求涵盖“特定区域的可达性分析”、“指定对象的位姿检索”以及“满足特定交互属性的候选集筛选”等。在数据维护机制上， 系统采用严格的时间对齐与质量评估策略， 以实现新观测数据流（来自视觉、深度相机、 IMU、力/触觉传感器）与现有地图的同步融合； 当探测到空间几何结构发生显著变更时，相关的语义属性（如可放置区域边界、最小通行宽度） 将触发自动重评估机制。例如， 在家庭服务场景中， 系统需同时维护“桌面”的几何平面信息与“杯子”的语义实例信息，从而在统一的感知接口下支撑抓取与放置动作的混合规划。

### 3.3.2 本体状态表征

本体模块（Body Module） 旨在构建机器人自身“所处状态”与“可控能力”的数字化孪生，是状态估计、轨迹生成及闭环控制的核心基础。该模块由状态管理服务与标定/坐标系管理组件构成，其核心表征内容包括：

- 运动学状态：实时维护各关节的位置、速度及（可选的）加速度信息， 记录上一控制周期的输入模式与指令数值； 同时计算末端执行器在任务空间中的位姿与速度， 以支撑任务级规划与执行误差校正。
- 动力学约束与能力边界：明确定义关节行程限位、速度/加速度/力矩饱和阈值、末端安全速度、功率/电流上限以及人机协作的安全最小距离。这些边界条件直接定义了系统的瞬时可行性动作空间（Action Space），并作为硬约束被纳入规划器与控制器的优化求解过程。
- 运行工况与健康监测：监控电机温度、母线电流、机械振动、夹爪开度及夹持力等关键运行指标，用于实时异常检测及系统降级保护策略的触发。
- 多坐标系管理：维护底盘、连杆、末端执行器及各类传感器之间的一致性坐标变换关系（例如 TFTree）。本体对象需负责各部件局部坐标系与全局坐标系的相互转换， 并在必要时执行外参标定与零漂校准，确保多模态感知数据与控制指令在统一的参考系下对齐。

此外，系统应具备主动的安全监测与反馈机制： 当检测到系统状态逼近关节限位、奇异位形或碰撞风险阈值时，动作原语层将向上游模块反馈“可行域收紧”的信号。这使得技能与规划模块能够及时调整姿态、重定位底盘或动态放宽局部任务的精度要求，从而在非结构化环境中维持执行的稳定性与安全性。

### 3.3.3 地图服务

空间地图服务（Spatial Map Service）负责构建机器人作业环境的几何结构与拓扑表征。该服务负责维护墙体、地面、家具等物理环境要素的数字化描述， 通常支持占据栅格、点云、距离场或三维网格模型等多种存储格式。作为底层的几何约束提供者，空间地图服务为路径规划、动态避障及空间可达性分析提供核心数据支撑。例如，在执行“导航至指定房间”技能时，路径规划算法的解算空间与可行性验证直接依赖于该服务提供的几何环境信息。

语义地图服务（Semantic Map Service） 负责在系统运行时构建并维护环境的高层语义表征。该服务以底层感知原语提供的多模态数据流（如 RGB- D 图像、点云、位姿及本体状态）为输入，实时识别并实例化场景中的对象（Object），进而维护对象的结构化属性及其相互间的语义拓扑关系。在系统交互层面，语义地图服务通过标准化的对象接口对外提供服务，使得

任务规划、方案推演、决策及执行等上层组件能够通过统一的协议查询对象集合、检索属性状态或遍历对象关系图，从而基于高层语义约束实现复杂的推理与操作决策。

### 3.3.4 数据采集服务

数据采集服务（Data Collection Service）负责在任务仿真推演或物理执行的全生命周期内，持续记录机器人的本体状态、传感器观测数据流、技能调用序列及控制指令流。该服务的功能定位主要体现在两个方面： 一是形成系统的“情景记忆”，为方案推演中的经验回溯与结果反馈机制提供数据基础；二是构建高保真的具身数据集，用于离线的大模型训练与技能策略优化，形成“采集—训练—部署”的数据闭环。

### 3.3.5 物体识别服务

物体识别服务（Object Detection Service） 负责从多模态感知输入中检测目标实体对象并解析其语义类别与空间属性，是连接原始感知数据与语义地图的关键环节。该服务接收相机图像、深度图或点云数据，输出环境对象的语义标签、置信度及其在三维空间中的位置（如3D 边界框或实例掩码）。

在技术实现上，该服务支持集成多种视觉算法范式，包括但不限于：基于卷积神经网络的实时目标检测模型（如 YOLO 系列）、具备开放词汇理解能力的多模态视觉语言模型（如 Qwen_VL），以及面向三维场景理解的空间感知模型（如 SpatialLM）。服务通过标准接口将检测结果上报，供语义地图服务进一步完成对象实例的初始化与属性更新。

### 3.3.6 校准服务

校准服务（Calibration Service） 旨在解决机器人长期运行导致的机械性能退化与模型参数漂移问题。由于机械磨损、形变或装配松动， 机器人的实际运动学/动力学特性会逐渐偏离上层模型所依赖的理论参数，导致控制误差累积甚至任务失败。因此，该服务负责定期执行感知系统（如相机外参）与执行系统（如关节零点、连杆长度） 的在线校准程序。更为关键的是，校准后的参数需实时同步至上层控制模型与规划器，确保模型生成的动作原语始终基于当前真实的物理能力边界，从而维持系统的执行精度与安全性。

## 3.4 认知空间

### 3.4.1 世界模型

EAIOS 中的世界模型（World Model） 主要负责建立对物理环境演化与本体行为结果的预测能力，为系统的方案推演与决策评估提供动力学仿真或状态预测支撑。其技术实现形式具

有多样性，既包括基于物理引擎的确定性仿真器（如 Webots[70] 等），也涵盖基于神经网络的生成式预测模型（如 V- JEPA[71]、Sora[46]、RoboBrain2.0[16] 等）。在系统架构层面， 世界模型作为推演服务内部调用的核心预测组件， 由 EAIOS 的模型管理模块进行统一的加载、配置与调度。

### 3.4.2 价值伦理

价值伦理（Value Ethics）表征了用户对具身智能系统的行为偏好设定与伦理规范要求。在 EAIOS 的当前实现中，该模块主要以结构化规则列表的形式存在（并预留了向其他高维表征形式拓展的接口）。作为系统级的全局约束机制， 价值伦理贯穿于任务语义解析、方案规划生成、推演评估及执行控制的全流程（例如，通过大模型提示工程或约束求解器嵌入）。其核心功能在于对可行行动空间进行剪枝，并对规划生成的候选方案进行价值过滤，从而确保系统的整体行为逻辑符合用户偏好设定并遵循基本的公共伦理准则。例如：

```
{
    "type": "ethics",
    "scope": "all",
    "time_range": "22:00-08:00",
    "target": "audio_output",
    "action": "avoid",
    "description": " 晚上 10 点至早上 8 点期间应保持安静，尽量避免使用语音播报等发声类原语"
}
{
    "type": "ethics",
    "scope": "all",
    "target": "harm_human",
    "action": "forbid",
    "description": " 禁止任何以伤害人类、诱导危险行为为目标的任务"
}
```

用户可通过 EAIOS 提供的标准系统接口对价值伦理规则进行动态配置（包括添加、更新或删除）。此外， EAIOS 内核预置了一组不可变更的基础价值伦理集， 旨在从底层拦截任何以伤害人类为目标或显著违背公共伦理准则的任务请求。

### 3.4.3 人机交互与协同服务

交互服务（Interaction Service） 是用户意图接入 EAIOS 的入口，负责支持具身智能系统与人类用户及其他智能体之间的高层任务级交互。当前的机器人系统多依赖于人工遥控操作，

缺乏对高层语义意图的自主理解与执行能力，已成为制约其在复杂非结构化场景中落地应用的主要瓶颈。

面向未来的具身智能系统应当具备自然交互能力，即无需依赖低级遥控指令，即可通过多模态通道自主解析用户意图并完成任务部署。为此， EAIOS 提供了标准化的交互服务， 负责接收源自语音指令、自然语言文本、图形用户界面（GUI）或机器间通信（M2M）的输入信息。该服务的核心职能在于意图理解与任务生成：通过对多模态输入进行语义解析，将非结构化的自然语言或指令映射为系统可理解的明确任务目标、约束条件及参数集合，从而为后续的任务规划服务提供标准化的任务描述接口。

此外，在多机协作或异构系统集成场景下，交互服务还承担着跨节点通信职能，用于接收来自上层调度系统或其他协同机器人的任务请求，实现分布式环境下的任务分发与协同执行。

### 3.4.4 任务规划服务

任务规划服务（Task Planning Service） 负责执行从高层用户意图到可执行动作序列的转化， 其输出为一种结构化的任务中间表示 RTDL。例如， 当系统接收到“前往厨房获取水瓶”的高层任务时，任务规划服务将结合空间地图信息、语义对象状态及本体能力模型，生成包含“导航至厨房区域 → 视觉定位水瓶 → 执行抓取技能 → 返回指定位姿”等步骤的 RTDL 程序，并提交至系统运行时进行解释与调度。

### 3.4.5 方案推演服务

方案推演服务（Plan Simulation Service）负责对任务规划生成的 RTDL 指令序列进行可行性与安全性评估。该服务以当前的环境表征、系统安全规则及价值伦理规范为输入，利用世界模型（World Model）作为预测引擎，对规划方案进行动态推演。其输出通常包含预测的视频流、状态变化序列或文本化描述， 旨在提前识别潜在的碰撞风险、运动越界、不可达操作或对系统限制与用户约束的违背情况，为后续的决策制定提供依据。

### 3.4.6 决策服务

决策服务（Decision Service）是系统进入物理执行阶段前的最终裁决模块， 负责从任务规划生成的候选 RTDL 方案集中优选出最佳执行路径。该服务基于多目标优化逻辑， 综合考量方案推演的可行性报告（如碰撞检测结果）、价值对齐服务的伦理评估、安全规则的硬性约束，以及任务执行代价（涵盖时间成本、能耗估算及潜在风险） 等关键指标，最终输出唯一确定的可执行方案并下发至执行层。

其中，价值对齐（Value Alignment）机制负责定义并维护系统在任务执行过程中必须遵循

的价值约束体系。这包括公共伦理准则与用户自定义的偏好规则，例如“严禁进入特定隐私区域”、“禁止对特定类别物体执行破坏性操作”、“严格遵守人机安全距离”以及“行动需符合社会伦理规范”等。

### 3.4.7 记忆服务

记忆服务（Memory Service）致力于构建具身智能系统的长期认知基础， 负责对系统运行过程中的多模态经验数据进行统一的存储、组织与访问管理。该服务为任务规划、方案推演、决策评估及结果反馈提供可检索的历史证据与先验知识支持。在技术实现上，记忆服务通过对执行过程中的感知数据、决策逻辑与执行结果进行摘要提取与结构化固化，支持基于时间维度、空间位置、对象及任务情境的高效索引与检索， 从而形成具身智能系统“可回忆、可复用、可演化”的系统级记忆能力。

## 3.5 动作空间

### 3.5.1 技能库

技能库（Skill Library）是 EAIOS 中汇集系统已掌握、并可被直接调度执行的技能集合。该模块采用结构化机制对每个技能的执行入口、参数接口规范及环境依赖关系进行统一维护， 以确保系统在运行时能够实现对技能的高效检索与即时调用。在系统实现层面，单个技能单元被封装为一条技能记录（Skill Entry），其典型的结构定义如表4所示。

表 4: 技能记录（Skill Entry）结构定义

| 字段 | 含义说明 |
|---|---|
| skill_id | 系统分配的全局唯一技能标识 |
| name | 技能名称 |
| type | 技能类型：基本技能或 RTDL 技能 |
| entry | 技能执行入口，指向主 RTDL 程序或基本技能执行入口 |
| input_schema | 技能启动参数的结构化定义 |
| output_schema | 技能执行结果与状态反馈的数据结构定义 |
| metadata | 技能的附加筛选信息，如适用场景与本体类型 |
| status | 技能当前状态标记，如可用或禁用 |

当系统面对熟悉任务或稳定场景时，可直接从技能库中选取匹配技能执行，从而避免重复的规划与推演过程。

### 3.5.2 任务集合

任务集合（Task Collection）是当前系统统一维护的任务实例集合。任务从进入系统任务队列开始，到执行完成或被终止为止， 其整个生命周期均处于系统管理之下。任务上下文（Task Context）则用于保存任务在运行过程中的动态状态与中间信息。

在实现层面，任务数据结构用于描述该任务在调度与管理层面的信息，任务结构定义如表 5所示；同时，系统为每个任务维护一份独立的任务上下文，用于记录执行过程中的运行态信息，其结构定义如表 6所示。

表 5: 任务（Task）结构定义

| 字段 | 含义说明 |
|---|---|
| task_id | 系统分配的全局唯一任务标识 |
| goal | 任务的目标描述，可为结构化目标或自然语言指令 |
| priority | 任务调度优先级，用于任务队列排序与调度决策 |
| queue_state | 任务在任务队列中的状态，例如排队、就绪或挂起 |
| owner | 任务发起者与权限域信息，用于访问控制与审计 |
| result | 任务结束后的执行结果摘要与完成状态 |
| timestamp | 任务创建、开始执行与结束的时间信息 |

表 6: 任务上下文（Task Context）结构定义

| 字段 | 含义说明 |
|---|---|
| rtdl_program | 当前的 RTDL 程序，作为执行控制流程 |
| execution_status | 任务当前所处的执行状态标记 |
| progress_cursor | RTDL 代码执行进度位置，用于异常恢复与继续执行 |
| context_state | 任务执行依赖的上下文状态集合， 例如语义地图快照、本体状态与关键对象状态 |
| model_session | 与任务关联的模型运行会话信息，例如所用模型实例与会话状态 |
| runtime_data | 任务执行过程中的中间数据， 包括技能调用结果、原语反馈与诊断信息等 |

任务的执行过程由任务状态机（Task State Machine）进行刻画。状态机描述任务在“任务规划、推演、决策、执行、完成或终止”相关状态之间的转移关系。

### 3.5.3 技能管理

技能构成了动作空间的核心抽象单元，机器人的能力边界主要由其掌握技能的丰富程度所定义。技能的构建主要源于两条路径：一是经由显式的工程化编程生成；二是在任务规划阶段，利用 VLA/VLM 模型自动生成执行逻辑并经由系统封装固化。

技能管理机制在架构层面实现了“慢系统与快系统”的双重认知模式[72]。“慢系统”主要针对系统尚未掌握的复杂任务，通过调用大模型进行深度推理以生成任务方案，并执行全流程的推演评估。该过程基于对任务可行性、价值伦理及安全规则的全局建模，能够提供具备可追溯性与可检查性的决策依据， 但通常伴随较高的计算时延。“快系统”则面向已熟知的任务场景，侧重于低时延的反应式快速执行。 EAIOS 将这两种能力纳入统一的动态演进框架：对于陌生任务，系统通过慢系统反复进行规划、推演与实地验证； 当特定任务的执行逻辑在多次迭代中表现出稳定性与可靠性时，系统将其识别并固化为可复用的技能存入技能库，从而转化为快系统的一部分。这一机制使得智能系统能够在后续任务中直接调用技能实现高效响应，在无需离线人工编程干预的前提下，实现技能的在线积累与智能水平的持续增长。

### 3.5.4 安全规则

安全规则（Safety Rules）严格定义了系统物理运行的安全边界。在 EAIOS 的当前实现中，安全规则以结构化规则列表的形式存在（并预留了未来向其他表征形式拓展的空间），其作用在于直接对技能与动作原语的物理执行过程施加确定性约束。例如， 以下安全规则通过设定阈值机制，确保当机械臂末端执行速度超过安全上限时，能够立即触发中断机制以终止对应动作原语的执行：

```
{
    "type": "safety",
    "primitive": "arm_move_ee",
    "max_velocity": 0.4
}
```

相较于价值伦理， 安全规则通过运行时检查（Runtime Check）机制实施强制性约束。该检查机制由 EAIOS 在动作原语的执行周期内持续运行， 旨在对受控的物理量、空间几何区域及系统执行状态进行实时监控。 一旦检测到安全规则被触发（即违背设定约束），系统将立即中断当前动作原语的执行， 并激活预定义的安全回退流程。此外， EAIOS 提供标准化的系统接口，支持用户根据具体应用场景对安全规则进行动态配置（包括添加、更新或删除），典型的配置操作包括为动作原语设定速度与力矩的饱和阈值，或划定严格的移动禁入区域等。

### 3.5.5 任务执行

任务执行是 EAIOS 中连接高层任务语义与底层物理执行的关键运行时组件，负责将经由决策层确认的任务实例转化为具体的执行流程。该功能实现了对任务的统一调度与全生命周期管理，能够依据任务的语义属性、当前系统状态及资源负载情况，动态分配所需的计算与执行资源。在运行过程中，系统支持基于任务优先级与执行阶段的动态调度，具备对多任务

并发、抢占、中断及恢复的管控能力，并能有效协调多任务间的资源竞争与依赖关系。

### 3.5.6 结果反馈

结果反馈服务（Result Feedback Service） 负责在物理执行阶段结束后，对任务执行效果进行自动化的评估与判定。该服务通常基于多模态智能模型构建，通过对执行全过程的多维数据（涵盖现场视频流、运动轨迹及系统操作日志等） 进行综合分析， 实现对任务完成状态、关键步骤正确性及潜在执行异常的精准检验，并将最终判定结果与诊断信息反馈至系统闭环中。

## 3.6 具身模型与 EAIOS 抽象概念的关系

在 EAIOS 中，具身模型以服务化能力组件的形式，通过统一的服务接口接入系统运行流程。（1）大语言模型（LLM）通常以任务规划或认知类服务的形式接入系统， 用于将用户输入转化为以对象及其目标状态为核心的任务目标描述与任务执行流程， 并生成 RTDL 形式的任务控制逻辑。LLM 仅参与任务层面的语义理解与流程生成， 不直接参与技能执行。（2）VLM 同样以服务形式接入系统，在具备 LLM 任务生成能力的基础上，进一步提供空间理解、时序图像理解与轨迹推理等能力，并可用于对任务推演结果进行基于视觉的评估与验证。VLM 不直接执行技能，其作用体现在对任务生成与评估过程的支撑。（3）VLA 及基于强化学习训练得到的策略模型通常被封装为技能的具体实现形式，通过技能接口被系统调度执行，用于在给定目标对象、约束条件与当前对象状态的情况下生成可执行的硬件动作序列，承担动作空间中的执行职责。（4）感知模型（如目标识别模型 YOLO 等）以感知服务的形式接入系统， 负责对系统对象级世界表征进行持续更新，为任务规划与技能执行提供环境状态与对象信息支撑。（5）世界模型（如 V- JEPA） 以推演服务的形式接入系统，用于对对象及其关系随时间演化的过程进行模拟，支撑任务层对执行可行性、安全性与风险的判断。

## 3.7 支持现场实时智能计算的安全内核

具身智能系统通常部署于算力与功耗受限的端侧硬件平台。为支撑抓取、导航等复杂具身任务，系统需并发运行多类异构进程。这些进程的资源需求特征差异显著：控制类进程对响应时延及抖动具有严格的确定性约束，而感知与规划类进程则对计算吞吐量有较高要求。

在具身智能交互过程中， 机器人执行的任务类型（如物体抓取操作、长程路径规划） 及所处的环境状态（如动态障碍物干扰、非结构化场景） 具有高度的时变性， 导致系统整体的资源负载呈现剧烈的动态波动。通用操作系统基于静态优先级或固定时间片的调度机制，难以在资源受限的端侧条件下，持续兼顾不同具身任务对时延确定性与计算吞吐量的差异化需求。

因此，为确保具身智能机器人在复杂动态环境中的作业稳定性与安全性，操作系统内核必须具备面向运行时状态的动态资源编排能力。系统需结合具身任务的当前执行阶段及其即时算力需求，对底层计算与 I/O 资源进行自适应优化与实时调配。

在构建内核架构时， 需充分考量现有的软硬件生态兼容性。 一方面，Linux 内核具备广泛的异构硬件支持能力； 另一方面，ROS 等主流机器人中间件生态高度成熟。借鉴Android 操作系统的设计理念，本系统基于 Linux 内核 进行面向具身智能场景的深度定制与优化， 引入“任务语义驱动”的动态调度机制：

1. 环境与任务感知： 内核层引入感知机制， 主动解析当前具身任务的语义上下文及环境状态，实时评估各进程的资源紧迫度与业务重要性；
2. 优先级动态调整：基于重要性分析结果， 自适应调整各任务的调度优先级与资源配额，优先保障关键路径的资源供给；
3. 与中间件的调度协同： 建立内核调度器与机器人中间件（如 ROS）资源管理机制的深层映射，实现软硬一体的统一优化调度。

经由上述机制优化的内核不仅能显著提升多任务并发场景下的系统稳定性，同时具备以下关键特性：

1. 生态兼容性：保持对主流硬件驱动与上层应用生态的高度兼容，降低迁移与开发成本。
2. 系统级安全可靠：提供确定性的底层内核服务， 为具身智能系统在物理世界中的安全运行构建坚实的混合关键性（Mixed Criticality）保障。

## 3.8 运行视图

EAIOS 的运行视图如图14所示。当用户提交任务后，该任务首先被置入任务管理器的任务队列（Task Queue），并由任务管理器进行统一的生命周期管理。任务管理器为每个任务构建并维护一份任务上下文（Task Context），用于持久化存储该任务在整个生命周期中的关键运行信息。任务上下文不仅包含任务目标、输入参数及当前执行状态，还涵盖生成的任务方案（RTDL）、环境与对象的状态快照（如语义地图与本体状态），以及模型运行时的中间状态与会话信息。

基于任务上下文，系统的执行流程如下：在任务规划阶段，调用相关服务生成 RTDL 方案；在方案推演阶段，对方案进行可行性与约束验证；在决策执行阶段，调度器（Scheduler）根据任务队列、优先级及系统资源状况调度可执行任务。执行期间，调度器负责协调技能库与原语的协同运行， 而异常处理程序（Exception Handler）则持续监控技能与原语反馈的运行状

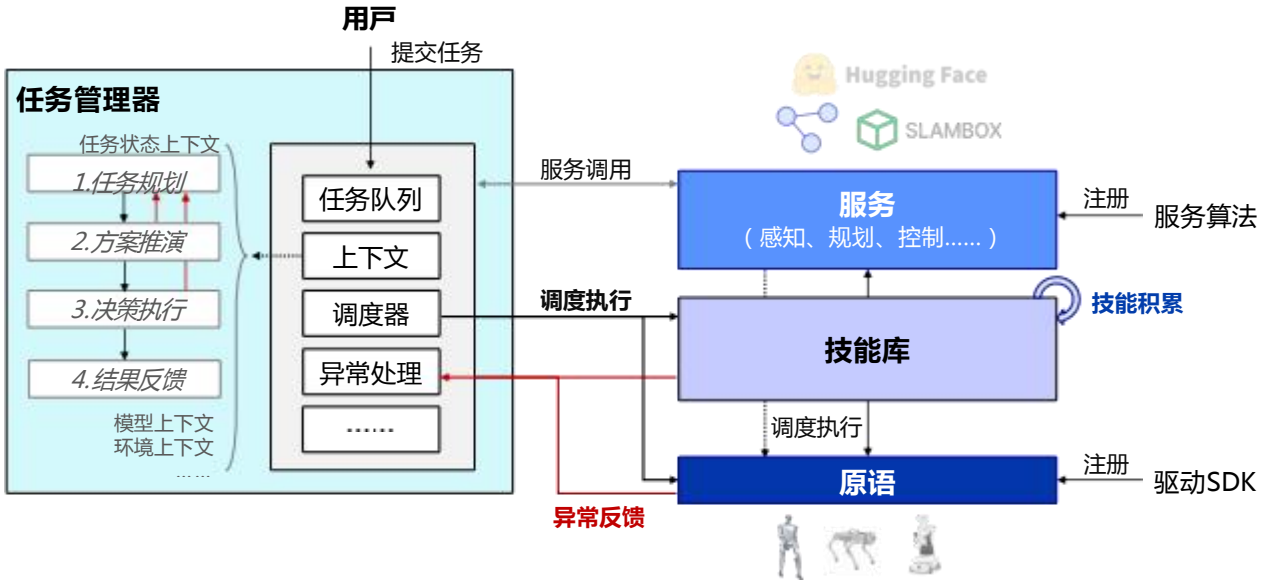


图 14：EAIOS 运行视图

态；一旦捕获异常，系统将结合任务上下文触发暂停、局部回退或重规划等处理策略。任务结束后，系统进入结果反馈阶段，对执行过程与结果进行验证。值得注意的是， 当某类任务的执行流程在多次运行中表现稳定时， 系统将其内部逻辑固化为可复用的技能并存入技能库，从而在后续任务中直接调用，实现智能体能力的持续积累。

此外，EAIOS 通过数据采集服务对任务执行全过程中的模型输入输出、中间决策结果、最终执行效果及感知控制原语数据进行结构化记录，形成任务运行日志。用户可利用这些日志作为 VLM 与 VLA 模型的训练数据，支持模型的持续微调与优化，并通过系统的模型管理功能完成新模型的部署。

## 3.9 案例：基于厂商VLA 模型的“室内巡检”

本案例旨在阐述当厂商或模型开发者直接提供端到端视觉一语言一动作（VLA）模型时，EAIOS 中的任务执行机制。在此模式下，复杂的任务能力被封装为单一的技能实体，系统规划层可直接调度该技能以执行任务， 这构成了 RTDL 程序的一种基础形态。本案例假定厂商针对特定机器人平台及室内环境，训练并发布了名为 `vla_indoor_inspection` 的技能。

当用户提交“巡查所有房间并关闭所有窗户”的任务后，任务目标将以自然语言指令的形式传递至该技能。模型在运行过程中自主完成环境感知、目标锁定及动作生成。在执行阶段，机器人持续采集环境中的多模态数据并反馈至 VLA 模型，模型基于实时观测生成控制信号以驱动机器人本体。在此过程中， EAIOS 负责运行时的生命周期管理， 涵盖技能进程的启停、异常状态监测及系统级安全约束的触发。技能执行完毕后，系统利用结果反馈服务对任务完成度进行量化判定（如确认房间遍历情况及关窗操作的成功率），并将最终结果反馈至任务管理

器及用户端。

**异常处理**

在端到端 VLA 技能执行模式下，任务上下文囊括了近期的多模态观测历史（如连续的 RGB_D 图像流、机器人位姿及关节状态轨迹） 以及模型内部的推理状态（如 KV Cache）。当系统检测到运行时异常（例如机械臂受力超限、运动速度异常、目标操作失败或传感器数据丢失）时，将触发技能的中断或暂停机制。系统随即将异常信息与当前任务上下文作为增量输入回传至VLA 模型。模型在随后的推理周期中， 依据更新后的观测数据与上下文信息， 自主制定后续的行动策略， 如执行重定位、调整抓取姿态或跳过当前子任务以继续巡检其余区域。

该案例展示了一种以端到端 VLA 模型为核心的具身智能执行范式。在此范式下，复杂的任务能力由开发者在离线阶段完成端到端训练并封装为技能，运行时系统则以低侵入式的管理方式实现模型的加载、调度与结果评估。该模式适用于任务定义明确、场景相对封闭且对整体推理性能与部署效率有较高要求的应用场景，与基于标准原语及可组合技能的执行路径构成了互补的技术路线。

## 3.10 案例：基于标准原语的“室内巡检”

本节以“巡查所有房间并关闭所有窗户”为例，说明任务在 EAIOS 中的常见执行方式。任务在运行时可以直接调用技能、服务与原语，系统负责在调度阶段根据需要选择合适的实现实例。原语由机器人本体厂商提供，如底盘控制、机械臂末端控制与深度图像采集；服务由开发者实现，包括空间地图、语义地图、控制、方案推演与结果反馈等功能； 技能在服务与原语之上封装可复用的动作逻辑。任务执行前假设系统的能力支持情况如表 7所示。

表 7: “室内巡检”任务前系统能力总览

| 类型 | 系统支持情况 |
| --- | --- |
| 原语 | `base_setvel`（底盘速度控制），`arm_move_ee`（机械臂末端控制），`gripper`（夹爪控制），`capture_rgbd`（RGB_D 采集），`audio_play`（语音播报） |
| 服务 | `spatial_map`（空间地图），`semantic_map`（语义地图），`control`（控制服务），`plan_simulation`（方案推演），`decision`（决策服务），`result_feedback`（结果反馈） |
| 技能 | `speak`，`navigate_to`，`close_window` |

当用户提交“巡查所有房间并关闭所有窗户”任务后，任务规划服务将生成一段 RTDL 代码，例如：

```
task patrol() -> bool:
    rooms = srv:semantic_map.query(types=["room"])
    for r in rooms:
        skl:navigate_to(target=r)
        win = srv:semantic_map.query_one(types=["window"], room=r)
        skl:navigate_to(target=win)
        prm:arm_move_ee(pose=win["handle_pose"])
        skl:close_window(target=win)
    return true
endtask
```

该程序形成后会先交由方案推演服务进行可行性检查。方案推演服务的实现可基于仿真器（如 Webots[70]、NVIDIA Isaac Sim[73]等） 或神经网络模型（如 V- JEPA），根据空间地图服务提供的几何结构与语义地图提供的对象位姿构建可预测的环境表示， 逐条解释 RTDL 指令仿真运行，并在内部模拟机器人轨迹、碰撞、关节限制与约束可达性。

当方案推演服务返回预测结果（如视频序列）后，还要经过任务决策，对预测结果进行判断，通过价值对齐、符合安全规则约束等要求的方案才会被允许执行， 之后 RTDL 程序进入真实执行阶段， 此时每条技能、服务、原语调用会由任务管理器调度至相应的实例进行执行。例如，`skl:navigate_to` 内部首先向语义地图服务请求目标房间的全局位姿， 然后通过 Nav2 进行路径规划，最终通过底盘速度控制动作原语完成移动；`skl:close_window` 的内部逻辑则通常包括使用语义地图定位窗户位置，再由基本行动模型 VLA 根据视觉原语规划生成对夹爪原语的调用，最终完成关闭动作。

任务在执行过程中，数据采集服务持续记录状态与视频， 以供结果反馈服务在执行结束后进行判断。结果反馈基于多模态模型逐帧分析巡检视频，判断机器人是否依次进入所有房间、是否成功完成关窗动作、是否存在漏关或失败情况， 从而给出最终任务成功与否的结论。

当系统尚未掌握某项技能时， 会进入“慢系统”，任务规划服务生成 RTDL 程序，方案推演服务在仿真环境中进行多轮预演与可行性评估，决策通过后再进行真实执行， 由结果反馈服务判断执行质量，并在必要时修正规划与策略。在某类任务经过多轮迭代后表现稳定时，系统会将对应的 RTDL 程序固化为技能， 记录其名称、参数结构及依赖的服务与原语， 并注册到技能库中（如表 8所示）。此后同类任务可直接调用该技能， 由“快系统”快速完成决策与控制。

此后，当用户提出新的指令， 如“先播报开始巡检， 再巡查所有房间并关窗”，系统直接调用技能库中已有的 `skl:patrol` 完成任务，不需要重新尝试对“巡检并关窗户”进行规划验证等流程：

表 8: 技能库的更新

| 类型 | 系统支持情况 |
| --- | --- |
| 技能 | speak，navigate_to， close_window，**patrol** |

```
task patrol2() -> bool:
    skl :speak(" 开始巡检")
    skl :patrol()
    return true
endtask
```

在基于标准原语与可组合技能的执行模式下，任务的执行过程由显式的 RTDL 程序描述，系统能够对任务的执行状态进行跟踪，并在异常发生时进行处理。

**异常处理**

当任务执行过程中出现异常（例如技能、服务、原语执行异常），对应的技能、服务或原语会返回错误状态或异常信息， 任务管理器在接收到异常后进行异常处理， 处理策略包括任务暂停、局部回滚或重新规划等。例如在执行“导航到房间—关闭窗户”的流程中，若在某一房间内发现窗户位姿不可达， 系统可中断当前子流程， 并基于当前任务上下文（已完成步骤、当前对象状态与语义地图等），调用任务规划服务重新生成针对该房间的替代执行方案，或跳过该房间继续后续巡检。由于 RTDL 程序在系统中以结构化的代码形式存在，任务管理器能够明确定位异常发生的步骤和代码位置， 并在重规划后从指定位置继续执行，无需重新创建任务。

通过上述流程可见，任务执行成功后，系统可将满足条件的任务放入技能库， 成为可复用的执行逻辑， 从而使机器人在连续任务中不断积累新的可复用能力。该案例完整展示了 EAIOS 中“原语由硬件厂商提供、服务由开发者实现并按接口接入、技能由技能开发者通过大模型或手动编程方式进行封装复用、任务由系统统一调度”的工程分工模式，同时也清晰体现了“仿真评估 + 执行验证”所构成的双重安全闭环机制，为后续具身智能生态体系中多角色协同开发提供了可落地的参考范式。

通过引入 EAIOS，系统在工程层面形成了一套成体系的任务组织与执行机制。任务不再直接依赖具体硬件控制逻辑，而是通过技能、服务与原语的组合完成执行，从而显著降低了任务开发与系统维护的复杂度。在该框架下，系统能力可以通过技能的不断积累持续扩展，新任务可以直接复用已有技能完成；任务在真实执行之前通过世界模型进行安全验证，在执行之后通过视频与大模型进行结果验证，从而形成完整的安全闭环。这使得具身智能系统在安全性、复用性与可扩展性方面均得到提升。

更重要的是，EAIOS 使得硬件和软件（包括模型）真正解耦，让专业化分工和规模化生产成为可能。

# 4 新型具身智能计算硬件

具身智能操作系统（EAIOS）是连接硬件生态与软件生态的共性基础设施， 其“原语抽象—统一调度—跨平台兼容”的核心诉求，要求底层计算硬件不仅提供算力，更要提供可被原语语义驱动、可被运行时约束管理、可跨平台复用的系统级能力。当前主流计算硬件（GPU、NPU 等） 的设计仍以传统设备驱动与算子加速为中心，缺乏对具身系统“感知—规划决策—执行”闭环数据流与安全边界的系统化支撑：上层原语难以直接落到硬件、实时确定性缺乏保障、生态接口碎片化导致适配成本高、模型部署在端侧与分布式场景中缺乏柔性。本节从 EAIOS 运行机制出发，先剖析现有计算硬件的关键瓶颈，再给出面向 EAIOS 的硬件设计原则，并基于“三域（计算域—I/O 中枢—控制域）”给出下一代具身智能计算硬件的参考架构与软硬件接口规范化要点。

## 4.1 现有计算硬件的局限

现有计算硬件与 EAIOS 的核心诉求存在结构性不匹配，其局限集中体现在：原语语义与硬件执行单元脱节、实时确定性缺失、生态接口碎片化、模型部署缺乏端边云一致性与柔性，直接制约 EAIOS 的硬件抽象、资源调度与跨平台复用能力。

### 4.1.1 架构专用性与原语调度的适配鸿沟

现有计算硬件的架构设计通常未围绕 EAIOS 的“感知原语—动作原语”执行路径进行规划，导致硬件能力难以被原语语义直接调用，原语链路不得不依赖 CPU/驱动中转，端到端时延与抖动显著放大。

NPU 通过大规模 MAC 阵列优化了深度学习模型的张量计算，但高度专用化的指令集与算子路径往往以特定模型结构为中心， 难以覆盖具身系统中感知采集/预处理、跨模态对齐、控制指令解析、约束计算等多类型原语。例如，某些 NPU 可显著加速视觉特征提取，但在动作原语链路（如关节控制指令解析、轨迹约束计算） 上缺乏就近执行能力，通常需要 CPU 侧进行协议/语义转换与调度补偿，从而引入额外时延与不确定性。

GPU（GPGPU）具备更强的通用性， 但其以吞吐为中心的调度机制与 EAIOS 的细粒度原语并行与闭环控制预算并不天然一致。具身系统往往需要同时处理多源感知原语（相机、 IMU、力触觉等）与多维动作原语（底盘、机械臂、夹爪等） ，并要求关键动作链路具备可预测的调度与抢占能力。若硬件线程/队列切换开销达到 20ms 量级，则“感知采集—动作执行”闭环周期可能被拉长至 20ms 以上，显著超过许多动态控制场景常用的 10ms 级安全预算，导致实时控制失效风险上升。

### 4.1.2 实时性与确定性调度的硬件短板

EAIOS 的闭环控制不仅要求低平均时延， 更要求在最坏情况下可验证的确定性边界。现有 GPU/NPU 多采用“尽力而为”的排队与资源分配策略，缺乏面向安全关键原语的硬件级优先级识别、抢占、隔离与时延上界保证，导致同一原语在不同负载下的执行时延波动明显。

在具身系统中，紧急避障、急停、限幅保护等安全相关动作原语属于安全关键路径，其响应应可在最坏情况下仍满足严格预算； 但在通用加速器上， 高优先级原语往往与感知/推理任务共享同一执行队列或内存带宽，容易受到突发负载影响而延迟。另一方面，NPU 即便在特定推理任务上具备低时延优势，也常缺乏对原语优先级与抢占语义的硬件表达能力，导致安全关键动作原语无法被优先保证。更关键的是，硬件缺乏确定性时延保障会破坏闭环控制模型的稳定性： 当系统同时运行多模态感知、世界模型推演与规划计算时，动作原语链路的时延抖动会放大为轨迹跟踪误差与执行偏差，进而影响高精度场景（如工业装配、医疗辅助等）的可靠性与可验证性。

### 4.1.3 生态碎片化与系统适配的成本黑洞

EAIOS 需要屏蔽硬件差异以实现跨平台复用， 但现有硬件在接口、驱动框架与运行时能力上缺乏统一标准：NPU 领域厂商接口差异显著，架构迭代带来大比例接口变化，导致 EAIOS 驱动与算子适配需要反复重写； GPU 生态虽然在部分平台上较为成熟， 但在端侧散热/功耗约束、国产生态算子完备度、以及端—边—云协同一致性方面仍存在短板。

此外，私有协议与非标准数据格式会迫使系统在端侧进行多次格式转换与跨域搬运，增加额外时延与能耗， 并使“端—边—云”统一调度难以实现： 硬件间数据与任务迁移缺乏一致的接口语义，系统不得不在适配层承担复杂度，违背“硬件抽象、软件复用”的目标。

### 4.1.4 模型部署兼容性与端边云一致性的障碍

EAIOS 需要支持从端侧轻量化模型到具身大模型的多尺度部署，并在不同硬件形态（端/边/云、单芯片/多芯片）之间保持一致的运行与管理语义。现有硬件常见问题包括： 对新型模型结构（如 VLA/VLM 融合、注意力主导架构、动作指令解析相关算子） 支持不足；对小模型的加载与缓存机制不完善导致技能调用冷启动慢；对多精度（如 FP32、FP16、INT8） 与动态精度切换支持不一致，使系统难以在精度、功耗、时延之间做运行时权衡；对多芯片扩展缺乏与运行时一致的资源表达与调度接口，限制大模型在边缘集群与云侧的弹性扩展。

## 4.2 下一代具身智能计算硬件：原则、参考架构与接口

为支撑 EAIOS 的原语抽象与资源管理机制，硬件不应仅被视为具身模型推理的算力载体，而应被视为可被操作系统统一抽象、可被运行时安全管理、可被任务语义驱动调度的系统资源集合。面向这一目标，下一代具身智能计算硬件的设计应遵循四项一致的原则：（**1**）硬件原语化——以原语为最小能力单元对齐系统抽象；（**2**）实时确定性增强——为安全关键链路提供可验证的时延与抖动边界；（**3**）生态标准化——以统一接口吸收差异，实现跨平台复用；（**4**）模型适配柔性化——以可配置与多精度机制支撑多尺度模型部署。四项原则在工程上共同指向一套可落地的三域组织方式：计算域—**I/O** 中枢—控制域，并由 EAIOS 运行时在“任务—技能—服务—原语”层级进行统一编排。

### 4.2.1 具身硬件需求：面向闭环数据流与安全边界

具身智能系统的本质是“感知—规划决策—执行”的闭环。在这一闭环中，硬件需要同时承载三类性质显著不同的负载：连续流式的多模态数据输入、强实时的执行控制闭环，以及高算力的模型推理与规划计算。它们相互依赖，同时在资源层面强烈竞争；缺乏系统化硬件支撑将导致“感知不同步”“控制不稳定”“规划与执行脱节”等系统性问题。

因此，下一代具身硬件首先必须围绕闭环数据流提供高带宽、低延迟、低抖动的 I/O 能力。具身系统的 I/O 不是传统意义的“外设接入”，而是包含多源采集、跨模态对齐、流式汇聚、低延迟下发与高频反馈的连续通路： 上游多路视觉/深度传感器产生大带宽视频/点云流，本体传感器与力触觉提供百 Hz 乃至更高频率的状态反馈， 下游控制链路需要稳定地将控制指令送达执行机构并接收执行回传。若这些 I/O 流在系统中被迫绕行或频繁在通用处理器上进行协议与格式转换，将引入不可忽略的延迟与抖动，并把问题放大为全局性的闭环不稳定。

其次，硬件必须为强实时控制闭环提供确定的执行域与安全边界。具身系统稳定性不仅取决于关节级伺服周期， 还取决于动作原语调用与反馈链路的端到端确定性： 导航、抓取、力控等原语在执行中需要持续接收状态回传并在约束下更新控制输入。硬件平台必须确保在系统繁忙时关键控制路径仍维持稳定响应，并使限幅、急停、避障等安全机制在最坏情况下可生效、可验证。

最后，硬件需要支撑高算力的模型推理与规划计算，但必须与运行时管理职责明确分离并受运行时约束： 模型推理是可变延迟的高负载计算， EAIOS 的任务执行、资源调度、服务编排与安全监护是必须稳定的运行时负载。下一代具身硬件的关键不在于让推理“占满所有资源”，而在于让推理在被约束的资源窗口内运行， 并与任务执行的实时要求相容， 使系统同时满足吞吐、时延与安全边界。

### 4.2.2 三域层次化硬件设计思路：从设备堆叠到可调度资源

现有机器人平台多以通用处理器作为 I/O 与协议栈中枢，以加速器承担模型推理。在规模较小、模块较少时该方案可工作， 但当具身系统进入“多模态并发 + 多任务并发 + 多级控制链路”的形态后，会出现结构性瓶颈： I/O 通路被拆散为大量软件路径， 数据跨进程与跨模块多次搬运导致端到端延迟放大；模型推理的突发负载与控制、地图更新、任务管理等运行时负载争夺资源，造成难以预测的抖动与性能退化。

下一代具身硬件需要完成的关键转变， 是将硬件能力从“设备堆叠”重构为“可被 EAIOS 抽象与调度的资源集合”，并形成三域协同的系统组织方式：

控制域（**Deterministic Control Domain**）：面向执行器提供高频闭环控制与最小、确定的动作原语执行接口，承担底层安全边界落实（速度/力矩限幅、姿态约束、急停与安全态切换等）。控制域的目标是隔离不确定计算对确定控制的干扰，使上层推理出现异常输出或系统负载骤增时，控制域仍能在安全规则约束下维持可控行为。

**I/O** 中枢（**I/O Hub**）：承担传感器汇聚、总线适配、数据预处理与时间组织， 向上提供面向感知原语的统一输入，向下提供稳定的动作原语下发与回读通路。 I/O 中枢强调“就近收发、集中汇聚、统一对齐”，使多模态数据进入系统时已具备一致的时间组织与格式组织， 从而为语义地图、状态估计与数据采集等服务提供稳定输入基础，并在多源并发情况下维持可控时延与稳定服务质量。

计算域（**Compute Domain**）：承载高层感知、任务规划、方案推演与模型推理等计算密集型工作。计算域必须纳入统一资源管理框架： 当任务进入关键控制阶段或触发安全规则时，计算域的资源配额、执行优先级与并发度应能够被动态调整， 避免非关键推理阻塞关键控制链路。

### 4.2.3 下一代具身硬件参考架构

面向复杂具身系统，可采用以“计算域—I/O 中枢—控制域”为骨架的参考架构（图15）。其核心思想是将闭环链路拆解为三类可管理通路， 并由 EAIOS 运行时在“任务—技能—服务—原语”层级进行统一编排，从而把“硬件差异”主要收敛在原语实现与 I/O 中枢层，而不扩散到任务与技能层。

在该架构中，**I/O** 中枢以硬件近端方式完成多源采集、协议适配、数据预处理与时间对齐，对上输出满足感知原语语义要求的统一输入流，对下提供动作原语稳定下发与反馈回读通道。通过减少通用处理器对协议栈与格式转换的路径依赖， I/O 中枢显著降低多模态并发输入场景下的端到端延迟与抖动，并提升输入一致性与可观测性。

计算域承担模型推理与规划推演等可变延迟的高负载计算，但其运行必须受运行时管理约束：EAIOS 可基于任务阶段、优先级与安全状态， 对计算域进行配额控制、并发限制与抢占策略调整，确保关键控制阶段的资源窗口不被非关键推理吞噬，形成系统级的性能边界。

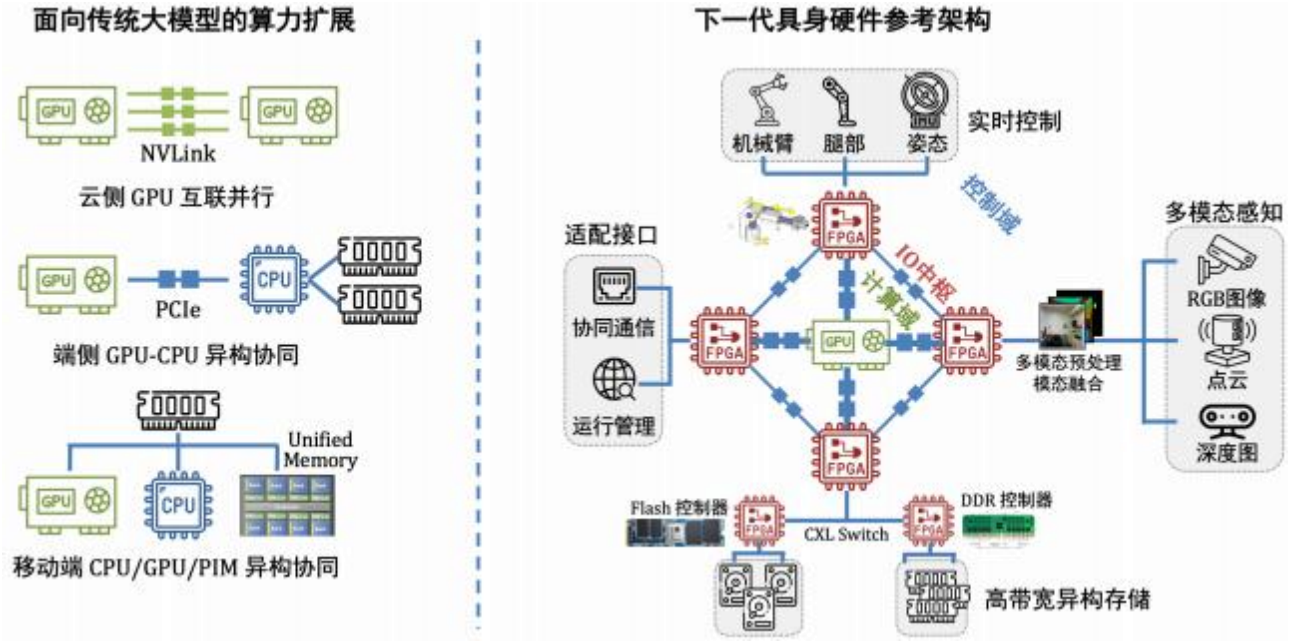


图 15: 具身智能硬件参考架构

控制域面向执行器提供确定性闭环控制，并以动作原语形式向上暴露最小执行接口，负责在最坏情况下落实安全规则与故障保护。控制域的工程目标是：即便上层推理出现异常输出或系统负载骤增，控制域仍能在安全约束下维持可控行为，避免概率模型的不确定性穿透到物理世界。

### 4.2.4 具身操作系统软硬件接口：以原语为中心的规范化要求

要使多样化硬件进入产业生态并支撑跨平台复用，需要形成与 EAIOS 一致的软硬件接口，使硬件差异被系统性吸收， 并使实时性与安全边界可表达、可管理、可验收。为此， 下一代具身硬件应在接口层满足以下规范化要求。

原语作为最小交付单位：硬件以原语为最小能力单元暴露能力。动作原语不仅包含控制指令格式，还应包含执行约束（限幅、速率、力矩/力控边界）、反馈字段（状态回读、误差、完成条件）、以及错误语义（可恢复/不可恢复、降级策略、故障码）；感知原语不仅输出数据流， 还应包含采样频率与抖动指标、同步信息（时间戳语义、对齐基准）、以及必要的标定上下文与坐标系引用，使上层语义地图、状态估计与规划服务能够在一致语义下融合多源观测。

实时确定性与资源可管理：硬件应提供原语级优先级、资源预留与隔离机制，使 EAIOS 在多任务并发、紧急任务插入与异常恢复场景下维持稳定资源供给与控制通路。硬件需支持可观测的调度状态反馈（队列深度、抢占事件、资源占用、时延统计），并允许运行时对关键通路实施配额、抢占与并发控制，从而为安全关键原语建立可验证的最坏时延边界。

校准与状态同步下沉：具身系统运行中本体性能会变化，感知与动作能力会漂移。硬件应支持将标定/校准结果下沉到原语实现， 并以一致的能力边界向上暴露， 使规划与控制服务能够

在稳定约束下运行， 避免“模型输出正确但执行偏差累积”的系统性失效。同时， 硬件应提供跨域时间基准与状态同步机制，保证感知输入、控制输出与反馈回读在系统层可对齐、可追踪。

模型适配柔性化与多精度支持：硬件应提供统一的模型加载与运行接口，并支持多精度（如 FP32、FP16、INT8 等） 与动态精度切换，使 EAIOS 能够依据场景在精度、时延与能耗之间进行运行时权衡。对于端侧小模型与技能模型，应具备快速装载与缓存机制；对于具身大模型与多芯片扩展，应通过标准化的分布式调度接口实现弹性扩展，并保证与控制域确定性链路的资源隔离。

综上，下一代具身智能计算硬件的核心价值不在于单点算力指标的提升， 而在于为 EAIOS 提供一套围绕闭环运行构建的系统级底座： I/O 可汇聚、控制可隔离、计算可分工、运行时可调度、安全规则可落地。只有当硬件以原语语义被统一抽象，并在 I/O 与控制层面提供可工程化的稳定通路，上层“任务—技能—服务—原语”框架才能实现跨平台复用与规模化落地。

## 4.3 基于虚拟机监控器的具身智能软硬件协同架构

具身智能软硬件系统的整体架构如图16所示，系统以虚拟机监控器（Hypervisor）作为底座，对计算资源、 I/O 通路与控制资源进行隔离， 在同一物理平台上并行运行实时安全内核与 RTOS，其中实时安全内核侧运行 EAIOS 的任务、技能、服务、原语的管理与执行，承载模型推理等高算力负载， RTOS 侧直接管理执行器与外设， 负责电源管理、电机控制等强实时任务，当执行过程中出现异常时，RTOS 可进行快速安全响应（例如急停），并通过高优先级事件上报触发 EAIOS 的异常处理机制，在保证实时性与安全性的同时支撑复杂具身任务的持续运行。

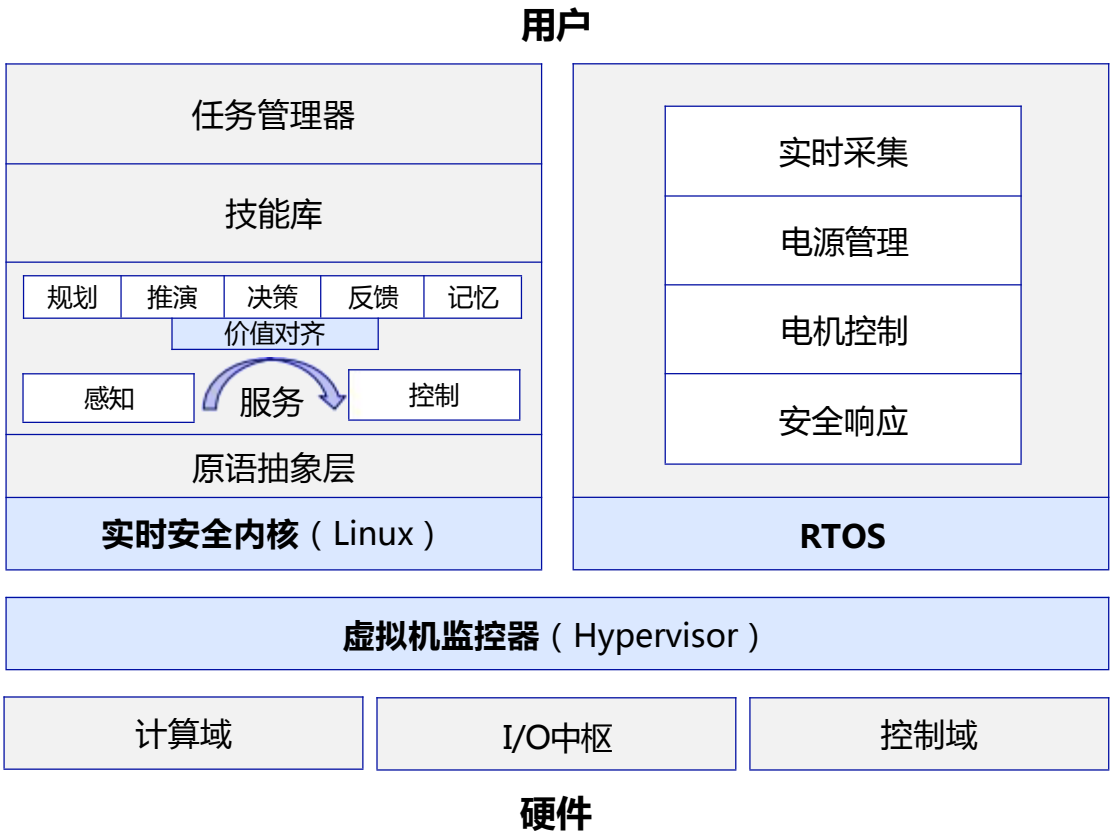


图 16: 基于虚拟机监控器的具身智能软硬件协同架构

# 5 测试与验证

本节阐述 EAIOS 的测试与验证方法论，旨在确保系统在接入异构本体、采用差异化原语实现及动态组合服务的复杂环境下，仍能严格遵循统一接口规范实现任务调度与资源管理，并在异常工况下维持系统行为的鲁棒性与可预测性。

## 5.1 系统基本能力测试

从系统工程视角，EAIOS 的测试与验证应当覆盖以下三个核心维度：首先是接口规范的符合性验证， 即确保任务、技能、服务与原语之间的数据交互与调用逻辑严格符合系统约定；其次是运行时调度的稳定性验证，涵盖任务生命周期管理、服务注册与注销机制及并发请求处理能力； 最后是跨硬件平台的抽象一致性验证， 即验证同一任务描述在不同机器人本体上，能否通过替换底层原语与服务实现而保持高层行为逻辑的一致性。在 EAIOS 架构中，任务管理器、技能库、服务管理模块与原语接口共同构成了系统的核心运行时环境。

鉴于 EAIOS 的核心设计目标是通过统一的原语与服务抽象来屏蔽机器人本体与硬件配置的异构性，系统测试必须重点覆盖跨本体与多实现的兼容性验证。验证过程可选取两种或多种典型的硬件平台（例如四足机器人与人形机器人），分别接入各自符合 EAIOS 接口规范的原语与服务实现。在保持上层任务描述与技能逻辑不变的前提下，对比分析同一任务在不同本体上的执行流程与最终结果：例如，验证“室内巡检”任务在不同平台上是否均能准确完成房间遍历与指定操作， 以及任务生命周期事件（如启动、重试、终止等） 的状态流转是否保持一致。

针对同一服务的多种技术实现（例如不同的地图构建算法或方案推演后端），测试重点在于验证“服务可插拔”设计的工程可行性。通过在不修改任务管理逻辑与技能实现的前提下替换具体的服务组件，确认 EAIOS 能够在服务切换过程中保持上层任务描述与执行流程表示的稳定性与无关性。

## 5.2 运行时行为与故障场景测试

为评估系统的鲁棒性， EAIOS 需在故障场景下进行运行时行为验证。通过故障注入（Fault Injection）技术，在任务执行过程中模拟服务调用失败、原语响应超时及传感器数据异常等边界条件，以验证任务管理器及相关服务能否准确检测错误并执行相应的异常处理策略，如中止当前技能、触发重规划或终止任务，并确保系统状态在故障恢复后保持一致性。在多任务并发场景下， 通过构造资源竞争场景（例如多个任务同时请求同一硬件原语），验证 EAIOS 调度策略在避免死锁（Deadlock）[74] 与饥饿（Starvation）[75] 方面的有效性， 并确保任务隔离机制

与资源分配逻辑符合预期设计。

## 5.3 回归测试与基准用例集

为保障 EAIOS 在持续演进过程中的行为稳定性，需建立系统级的回归测试（Regression Test）[76] 体系与基准用例集。回归测试集应包含一组覆盖典型应用场景的标准任务， 如单机巡检、多房间自主导航及基础操作任务等，每个测试用例均需定义明确的任务描述、预期的状态转移序列及最终结果。在系统版本迭代、接口规范调整或参考实现更新后，通过自动化执行基准用例集，检测任务生命周期管理、服务调度逻辑及日志记录机制是否存在功能回退。

# 6 开放生态与社区

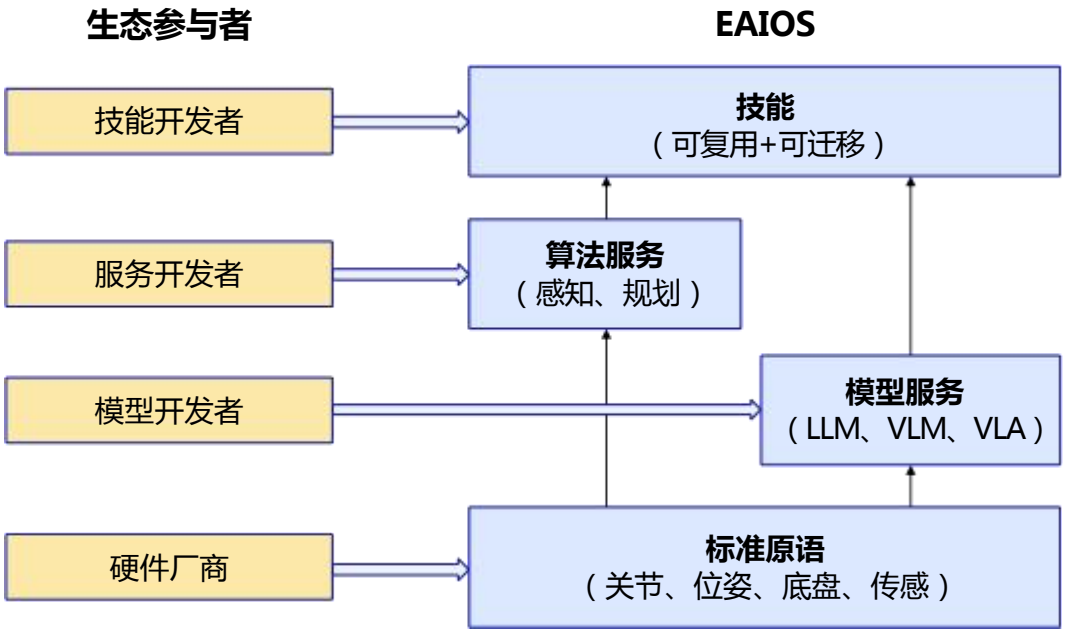


图 17: EAIOS 开放生态与社区

EAIOS 秉持开放、共建与标准化的设计理念。系统建立在统一的接口规范之上，旨在通过标准化的架构吸引科研机构、企业及个人开发者共同参与生态建设。EAIOS 采用 MulanPSL- 2.0开源协议，支持学术研究与工业生产中的自由使用与功能拓展，并致力于推动具身智能操作系统相关标准体系的建立。EAIOS 生态体系由四类核心角色构成，即硬件厂商、服务开发者、技能开发者与模型开发者：

硬件厂商（**Hardware Manufacturer**）涵盖机器人本体、机械臂、移动底盘及各类传感器设备的供应商。在 EAIOS 架构中，硬件厂商负责将其硬件能力封装为标准化的原语接口，从而提供包括关节运动控制、末端位姿控制、夹爪开合、底盘速度控制及多模态传感数据采集在内的基础本体能力。

服务开发者（**Service Developer**）专注于感知、规划、控制、决策与记忆等功能的算法封装。基于底层原语，服务开发者构建上层服务，并通过统一接口对外暴露语义地图查询、目标检测、任务规划及控制计算等运行时能力。在此模式下，服务开发者与具体的任务调度逻辑解耦，仅需依据接口规范提供可被系统调度的服务实例。

技能开发者（**Skill Developer**）致力于将经由规划生成的任务执行序列，在通过仿真评估与物理环境验证后，固化为标准化、可复用且具备迁移特性的技能。技能的来源包括大模型自动规划、人工编程、示教学习或流程设计。封装完成的技能将被发布至技能库，支持在不同机器人本体间进行复用，从而实现跨本体的行为迁移。

模型开发者（**Model Developer**）面向 EAIOS 提供各类智能模型能力，涵盖大语言模型（如

GPT[14]）、多模态模型（如 Qwen_VL[77]）、视觉模型（如 YOLO[78]）以及动作模型（如 OpenVLA[27]）等。模型开发者依据 EAIOS 服务接口规范，将其模型封装为可调用的服务实现（如任务规划服务、目标识别服务、结果反馈服务及控制服务等），使得模型能力能够作为标准系统服务参与任务的执行流程。

EAIOS 的开放生态以统一接口为基础，以服务的可插拔机制为核心，确立了硬件适配、算法封装、模型提供与技能构建之间的明确分工：硬件厂商专注于物理执行能力的实现，服务开发者负责封装可组合的算法功能，模型开发者提供可替换的智能模型支持，技能开发者则构建可迁移的行为逻辑。各类参与者通过 EAIOS 实现高效协同，共同构建具身智能技术的开放生态与社区（图 17），从而支持系统在异构本体、多样化场景及不同能力组合下的快速构建、复用与部署。

# 7 典型应用场景

本节设计 EAIOS 在不同领域的潜在应用，分析其如何解决面临的工程挑战，讨论其作用与优势。

## 7.1 酒店服务机器人

尽管酒店服务机器人已逐步迈向规模化应用阶段，但其在跨楼层自主通行、异构物联网系统（如电梯控制、门禁管理、PMS/客控） 协同，以及动态非结构化环境中的安全决策等方面，仍面临显著的工程挑战。传统解决方案多采用针对特定场景的脚本化集成模式， 由于不同酒店设施的接口协议存在差异，导致系统部署周期冗长、功能模块复用率低，且缺乏统一的任务回溯与审计机制。

针对上述问题，EAIOS 设计在酒店场景中构建统一的设备抽象层与数据总线（涵盖时钟同步、全局坐标系及语义地图），实现跨系统的异构接口编排（集成电梯调度、门禁验证及客控指令），并提供面向高层任务的“原语—技能”抽象体系。在此架构愿景下， 物品配送、宾客领位、安防巡逻等业务逻辑将主要由语义地图所维护的场景语义信息（如楼层拓扑、房间状态、禁行区域、人群密度分布） 驱动。在任务执行前， 系统拟利用内置的世界模型对候选路径及预期效果进行推演评估； 当遭遇环境拥堵、电梯拒载或门禁验证失败等异常工况时，系统将能够自动触发局部回退机制与即时路径重规划。安全内核层设计基于系统级约束，对人群安全距离、运动速度边界及动态礼让策略进行强制管控，并提供全链路的黑匣子日志记录与灰度发布能力，以确保系统行为的可追溯性与升级的稳健性。

基于上述设计机制，酒店机器人的一些关键指标有望获得显著优化：从任务分派至到达房门的端到端时延预计将大幅降低，配送失败率与人工干预频率显著下降；同一套标准化的技能库将具备在不同物理环境的酒店中实现快速迁移与复用的能力， 从而大幅缩减部署周期。以“送 1208 房毛巾”任务为例， 自然语言指令经解析后可转化为包含呼梯、进梯、楼层定位、门口等待及确认签收等标准化技能序列； 若执行过程中出现电梯拥堵或门禁异常， EAIOS 预期将自动调度备用电梯或触发前台协同流程，确保服务交付体验的一致性与可审计性。

## 7.2 巡检机器人（楼宇/园区/工厂）

巡检机器人需在长周期的半结构化环境中执行多模态检测任务，涵盖表计读数识别、热异常探测、液体渗漏监测、声学异常诊断及设施安全状态确认等。现有系统多基于独立的算法管线进行堆叠式开发，导致任务优先级管理混乱、异常中断策略分散，线上故障难以复现与定位，且缺乏可控的模型在线更新与灰度验证路径。

EAIOS 在巡检场景中计划将各类“检测能力”标准化为系统级服务（如表计 OCR 服务、热成像分析服务、液位/反光检测服务、声纹异常检测服务等），并拟利用世界模型动态维护巡检路径拓扑、站点语义属性及限制区域约束。同时，在任务调度层设计引入支持可中断与可回退的编排机制：一旦感知到高优先级异常信号，系统将基于运行时成本评估模型决定中断当前巡检路径，优先执行就地复核采样，随后生成包含结构化信息的异常工单并上报。通过云边协同的数据闭环机制，海量的巡检记录可被回流至离线评估系统进行仿真回放，新迭代的模型与检测阈值经由灰度策略逐步发布至端侧，以确保系统运行的稳定性。

该运行模式预期将带来更低的漏检率与误报率，延长无人工干预的自主运行时长，并显著缩短告警闭环时延。以“配电柜过热检测”为例：当机器人在常规巡检路径中捕获异常热斑后，EAIOS 设计依据预设策略中断当前导航任务，调度机器人执行近距离复核与多角度成像，并在同一任务上下文中生成附带温度曲线与位置标签的标准化工单；异常处理完毕后，系统自动恢复原巡检路线。全链路日志记录与确定性状态机机制将为故障定位与责任追溯提供客观的数据支撑。

## 7.3 物流机器人（仓内搬运/转运）

仓内物流场景对作业吞吐量、准时率及安全性有着极高要求，其核心难点在于多智能体协同、动态拥塞控制以及高峰时段的实时调度优化。传统系统通常采用“集中式调度中心为主、机器人端侧为辅”的控制架构，导致端侧智能体对环境语义变化感知迟钝，在遭遇异常（如通信中断、路径受阻、货位临时变更）时回退与恢复效率低下，制约了整体物流效率。

EAIOS 可为物流场景提供统一的资源管理与任务编排框架：拟将库位/工位状态、通道拓扑结构、空间几何约束（限宽/限高）及临时禁行区域纳入语义地图与场景记忆模块。借助端侧部署的世界模型与推演服务，机器人将具备局部预测与冲突预判的自主能力；多车协同机制设计通过时间窗分配与动态优先级策略实现高效的让行与插队控制。面对异常状态（如障碍物阻挡、网络断连、货物缺失），系统将能够触发本地重规划逻辑并尝试与调度中心进行状态同步。此外， 系统级的 GPU/NPU 资源管理器旨在实现视觉质检、路径优化算法与实时控制回路在同一计算平台上的并行执行，以显著降低跨模块通信延迟。

在此模式下，机器人的等待时间与空驶率预期将大幅下降， 单位时间任务吞吐量提升， 碰撞与急停事件发生率显著减少。以“波峰拣选”时段为例，EAIOS 能够根据任务截止时间与通道实时拥堵度动态分配最优路径；若目标拣选站点临时不可用，系统将自动切换至备用站点并更新后续任务链，确保整体作业节拍不受影响。统一的日志记录与支持回滚的发布机制， 将保障大规模车队在系统升级与运维过程中的一致性与可靠性。

## 7.4 智能工业机器人（装配/协同/复杂工艺）

工业智能制造场景通常具有强实时性、高安全约束、设备高度异构以及任务流程复杂且持续演化等特点，这使得以 PLC 和RTOS 为核心构建的传统工业控制软件体系， 虽然在确定性控制和稳定运行方面具有成熟优势，但在支撑感知融合、智能决策与复杂任务协同方面逐渐显现出结构性局限， 难以支撑具身智能系统软件的长期稳定运行。在这一背景下，EAIOS 并未将自身定位为传统意义上的机器人控制系统或智能中间件，而是尝试以操作系统范式，重构具身智能在工业场景中的工程基础。EAIOS 面向工业机器人、智能装备及其所处的物理环境，构建一个以“行动”为核心、以“对象”为基本单元的系统运行平台， 使感知、决策与执行能够在统一的系统框架内协同演化。

在工业智能制造场景中，EAIOS 通过“原语－服务－技能－任务”的分层行动抽象，对机器人的能力进行系统化组织。底层原语直接映射具体硬件与控制能力，中间层服务与技能封装可复用的工业操作逻辑，而任务层则描述面向生产目标的高层行为。这种抽象方式使复杂工业任务能够在保持执行确定性的同时，引入更高层次的智能规划能力，从而兼顾工业场景对稳定性与灵活性的双重需求。同时， EAIOS 以“对象”为基本系统单元，构建对外部世界的统一虚拟表征空间。工件、设备、工具以及环境要素均以对象形式纳入系统管理，并通过世界模型进行动态映射和更新。在工业生产过程中，设备状态变化、工件姿态偏移或环境条件波动，均能够被系统持续感知并反馈至运行时决策层，为具身智能在动态环境中的稳定运行提供基础支撑。

针对工业场景中对安全性和可控性的高度要求，EAIOS 将世界模型与安全内核作为核心运行时组件。在安全规则层面，系统通过世界模型对大模型生成的规划与行为进行推演与验证，在物理执行前评估其是否满足安全约束、工艺规范以及预设的价值和伦理要求。这一机制使智能决策不再直接作用于物理系统，而是经过操作系统层面的系统性校验，从而显著提升工业具身智能系统软件的可靠性与可预测性。在资源管理框架上， EAIOS 自底向上统一对异构硬件的抽象，将不同类型的控制器、传感器和计算单元纳入一致的系统管理框架之中，并向上提供标准化的行动接口。这种设计使工业现场常见的多品牌、多型号设备能够在同一操作系统框架下协同运行， 降低系统集成和长期维护的复杂度。 EAIOS 在系统层面实现了具身智能中“身体”与“大脑”的有效解耦。硬件能力通过原语和服务形式暴露给系统， 而大模型与上层智能软件则通过标准化接口进行调用，使大模型提供者、技能开发者与硬件厂商能够在相对独立的边界内协同发展。 EAIOS 可使智能能力能够跨设备、跨产线迁移， 显著降低智能系统的部署和演进成本。

工业智能制造场景具有明确的生产目标、严格的安全规范以及长期连续运行的要求， EAIOS 在继承工业控制系统对执行确定性高度重视的基础上，进一步将智能决策过程纳入系统级约束与管理之中，能在可控、可验证的系统框架内发挥作用，形成了面向工业智能制造的可落

地运行范式。

在柔性装配场景中， 生产任务一般呈现出小批量、多品种和频繁切换的特点。工件尺寸、公差范围以及装配顺序的变化，使得传统依赖固定程序和人工调试的自动化方案难以保持效率与稳定性。EAIOS 通过将装配对象、工装夹具和操作环境统一映射为系统对象， 并在世界模型中持续更新其状态，使机器人能够基于当前环境进行装配规划。在执行过程中， 由大模型生成的高层装配策略会先在世界模型中进行推演与校验，确保其满足装配约束和安全要求后再下发执行，从而在保证工业确定性的前提下实现柔性装配能力。

在多机器人协同的产线运行场景中，系统面临的核心挑战不再是单一设备的智能性，而是多个具身智能体之间的协同一致性。EAIOS 通过统一的任务与对象模型，将产线中的设备、物料和作业流程纳入同一系统视图之中， 系统将利用语义地图与世界模型动态维护工位属性、治具状态及工件信息（涵盖可达性分析、定位基准、工艺参数配置、禁入区域及人机协作区定义）。各机器人并非孤立执行预设动作， 而是在操作系统层面共享对生产状态的理解， 并通过标准化的行动接口进行协同。这种系统级协同机制使产线能够在任务负载变化、设备状态波动甚至局部异常情况下，保持整体运行的稳定性和连续性。

在焊接、打磨、喷涂等复杂工艺场景中，具身智能操作系统需要面对持续变化的工艺条件和环境干扰。传统自动化系统通常依赖经验参数和固定工艺曲线，一旦条件偏离预期，则需人工干预。EAIOS 将工艺参数、设备状态与环境感知统一纳入世界模型之中， 使系统能够在运行过程中持续评估执行效果。当感知结果与预期模型出现偏差时，系统可以在不破坏安全边界的前提下调整执行策略，从而实现面向工艺质量的自适应控制。

在工业现场的异常处理与恢复场景中， EAIOS 的系统级优势尤为突出。设备磨损、工件异常或环境突变，通常会导致传统系统进入停机或等待人工处理状态。 EAIOS 通过安全内核对异常行为进行约束，并结合世界模型对可行行动进行推演，使系统能够在一定范围内自主调整任务执行路径，完成安全降级或恢复操作，显著降低了非计划停线的风险，提高了工业系统的整体韧性。

EAIOS 在工业制造中的落地并非以单一应用为目标，而是形成了一种可复用的系统运行范式。通过“原语 - 服务 - 技能 - 任务”的行动抽象，硬件能力、工艺逻辑与智能规划被清晰解耦，使不同厂商、不同模型和不同设备能够在统一操作系统框架下协同演进。通过统一的接口规范与运行时治理架构，工业机器人系统有望从离散的“点式集成”向“平台化运营”转型，在确保功能安全与全流程可追溯性的同时， 加速智能算法的迭代优化与规模化部署进程。这种范式不仅降低了具身智能在工业场景中的部署门槛，也为构建开放、可持续的工业具身智能生态提供了基础条件。

## 7.5 移动操作机器人

面向未来的移动操作应用场景， 机器人需要在非结构化环境中完成移动与对按钮、把手、抽屉等对象的精细交互操作，并与电梯、门禁、告警系统等基础设施进行复杂协同。此类场景的技术实现路径多样，但落地难点集中体现在技能的跨场景复用性、操作过程的鲁棒性以及行为安全的可验证性。

EAIOS 可将“按钮按压、把手下拉、抽屉开合”等通用交互动作抽象为可复用的技能模块，并由本体状态机实时维护其可达性与物理约束（如关节力矩限幅、末端接近速度、人机安全距离）。在动作执行前，系统可利用内置推演引擎对候选动作序列进行仿真评估，必要时主动调用远程协助服务或切换至替代操作流程； 安全内核层设计提供基于地理围栏的空间约束、动力学层面的力/速度限幅及标准化的功能安全接口， 并对每次操作行为记录详尽的黑匣子日志，为合规审计与售后故障追踪提供数据基础。

在医院“跨楼层取送药”这一典型的综合性任务中， EAIOS 可将“取药—乘梯—移动至科室—确认交付—返回”的长程业务流程拆解为标准化的技能链， 并依据时间窗口约束、通行优先级策略及实时人群密度进行动态调整；若遭遇电梯故障或门禁权限受限等异常情况，系统预期将自动切换至备用路径或触发人工协同机制，确保整体服务能力的连续性。通过灰度发布与可回退机制，新型控制策略与感知模型将得以在保障安全的前提下逐步上线，从而推动移动操作技术从实验性试点走向规模化服务应用。

## 7.6 护理机器人（养老/助残/康复）

护理机器人对整机功能安全性、个性化及执行精准度要求较高，且面临严苛的系统伦理挑战，主要聚焦三大应用方向： 一是康复训练，为偏瘫、失能患者提供关节活动矫正、肌力恢复引导，适配个体状况并规避代偿动作风险； 二是日常护理，实现起床、翻身、如厕等精细交互动作，适配居家、病房等场景的复杂环境；三是应急监护，通过非接触式方式全周期监测心率、血氧等生理指标及跌倒风险，异常时实时多端预警至家属与医院。

EAIOS 作为康复机器人操作系统可构建精准安全的感知和控制操作能力， 自适应给出个性化交互和辅助方能，支持高等级的设备安全控制以及严格的伦理管控机制。在精准控制方面，采用感知与动作空间联合感控模式， 通过多传感器统一抽象、专用代偿识别算法， 实时数字化模拟患者康复动作标准度（关节角度误差 $\leq 3^{\circ}$），精准检测耸肩代偿抬臂等异常动作。经数字孪生模拟后， 以无害接触式阻力反馈矫正代偿，保障康复安全性。在个性化与伦理保障方面，依托认知空间服务构建个性化康复方案生成子系统，基于联邦学习模型，结合患者评估报告与过往数据， 自动生成周期性训练计划，支持医师远程调参。同时， 通过伦理与世界模型保障合规与隐私， 遵循本地算力优先策略， 核心数据默认本地存储， 仅加密上传异常数据；

数据共享需家属手动授权，授权记录区块链存证可追溯。在人机协同安全方面，智能安全内核以任务语义驱动构建功能安全机制， 严控任务边界（如搀扶动作力控精度 ±0.1*N*），通过压力反馈闭环控制感知肢体支撑力度，从底层阻隔机器辅助潜在伤害。

基于以上设计机制，EAIOS 操作系统为护理机器人整机性能升级与最受用户关注的关键指标落地提供了全面严格的基础软件支撑。感知控制端，通过多模态融合、数字孪生与闭环控制提升动作精准一致性；伦理隐私端，构建全流程数据管控与伦理审计体系，增强用户信任；人机协同端，以底层安全内核规避二次伤害，全面保障康复进程与人机交互安全。

# 8 路线图

具身智能操作系统的整体发展可分为四个阶段， 逐步从基础框架构建走向生态体系完善。

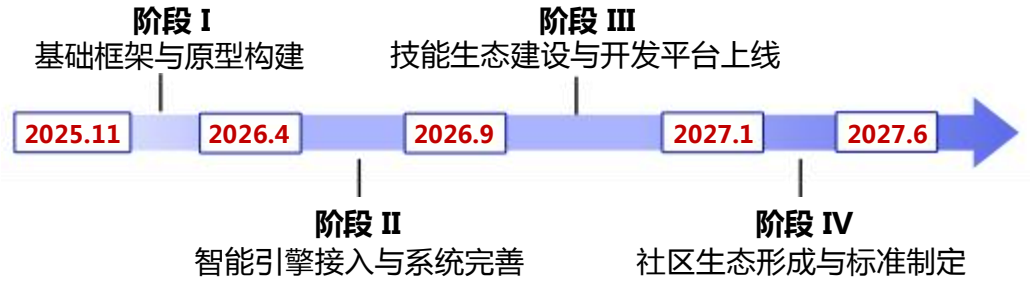


图 18: 路线图

目前矽望开源社区[3] 开始研发具身智能操作系统原型 Robonix：

- 阶段 **I**（**2025** 年 **11** 月至 **2026** 年 **4** 月）：基础框架与原型构建。 完成 Robonix 的总体架构设计，包括运行时调度机制、资源管理框架、通信与中间件抽象层的初始化开发。构建首个软硬件协同的原型机 Prototype 0.1，用于验证本体抽象层、能力接口层的可移植性与稳定性。

- 阶段 **II**（**2026** 年 **4** 月至 **2026** 年 **9** 月）：智能引擎接入与系统完善。 拓展系统对主流AI 模型（如 VLA、V- JEPA、GPT 系列等）的统一接入能力；完善运行时资源调度策略， 实现低延迟、高可靠的资源分配； 增强具身设备支持， 接入多类型传感器、执行器与移动平台，形成标准化驱动抽象。

- 阶段 **III**（**2026** 年 **9** 月至 **2027** 年 **1** 月）：技能生态建设与开发平台上线。 上线 Robonix 技能商店，提供标准化能力接口和技能包格式； 发布技能开发平台， 支持技能组合、参数化、迁移与版本管理；推动研发者与大模型能够基于统一接口快速构建与复用技能。

- 阶段 **IV**（**2027** 年 **1** 月至 **2027** 年 **6** 月）：社区构建与标准制定。 开源社区形成规模， 推动产业与学术双向协作； 制定数据规范、能力标准与接口协议， 提升跨平台兼容性； 完成多场景示范应用（服务机器人、工业协作、科研教学等），验证系统的通用性与扩展性。

[3] https://syswonder.org

# 9 附录

## 9.1 名词解释

表 9: 名词解释

| 缩写 | 全称 | 解释 |
| --- | --- | --- |
| **DDS** | Data Distribution Service | 数据分发服务 |
| **EAIOS** | Embodied Artificial lntelligence Operating System | 具身智能操作系统 |
| **FPGA** | Field_Programmable Gate Array | 现场可编程门阵列 |
| **PLC** | Programmable Logic Controller | 可编程逻辑控制器 |
| **GPU** | Graphics Processing Unit | 图形处理器 |
| **GPT** | Generative Pre_trained Transformer | 生成式预训练模型 |
| **GUI** | Graphical User lnterface | 图形用户界面 |
| **IMU** | lnertial Measurement Unit | 惯性测量单元 |
| **MCU** | Microcontroller Unit | 微控制器 |
| **MPC** | Model Predictive Control | 模型预测控制 |
| **NPU** | Neural Processing Unit | 神经网络计算加速器 |
| **OCR** | Optical Character Recognition | 光学字符识别 |
| **PID** | Proportional－lntegral－Derivative Control | 比例积分微分控制 |
| **PMS** | Power Management System | 电源管理系统 |
| **ROS** | Robot Operating System | 机器人操作系统 |
| **RTOS** | Real_Time Operating System | 实时操作系统 |
| **VLA** | Vision_Language_Action (Model) | 视觉语言动作模型（基本行动模型） |

| 缩写 | 全称 | 解释 |
| --- | --- | --- |
| **H_VLA** | Hierarchical Vision_Language_Action (Model) | 分层行动模型 |
| **VLM** | Vision_Language Model | 视觉语言模型 |
| **WM** | World Model | 世界模型 |

## 9.2 系统接口总览

本节给出具身智能操作系统在原语、服务、技能与任务四个层级上的典型接口设计示例。接口形式基于 ROS 2 通信机制（Topic、Service 与Action）[79]，并按照统一命名规范暴露于 `/eaios/…` 命名空间下。这里展示的接口用于说明系统抽象方式与调用模式，代表一种参考性的设计方案。

### 9.2.1 原语 API

标准原语的名称、参数名及其 ROS 2 IDL[80]类型均由 EAIOS 规范预定义。同一标准原语可能有多个实例（例如多个摄像头），EAIOS 应提供基于元数据的的原语查询能力，使服务与技能能够选择需要的原语实例。

原语注册接口

```
srv: /eaios/prm/register
inputs:
    name: string            # 标准原语名称
    input_schema: JSON      # {"argname0":"/topic0", .....}
    output_schema: JSON     # {"argname1":"/topic1", .....}
    metadata: JSON          # 用于选择原语实例
    provider: string
outputs:
    ok: bool
```

原语查询接口

```
srv: /eaios/prm/query
inputs:
    name: string            # 标准原语名称
    filter: JSON            # 按每个原语规定的 metadata 格式进行筛选，如:
```

```
                            # {"resolution":">=720p"}
                            # {"index":0}
outputs:
    instances: PrimitiveInstance[]

PrimitiveInstance:
    provider: string
    input_schema: JSON
    output_schema: JSON
    metadata: JSON
```

### 9.2.2 服务 API

服务封装感知、规划、评估、验证等算法能力， 每个标准服务接口在 EAIOS 中具有统一的名称、请求与响应结构和语义定义，实现者只需提供 ROS 2 service 并向系统注册。一个标准服务可对应多个实现实例， 例如任务规划服务的不同实现使用不同大模型， 同时加载到 EAIOS 中，按需进行使用。

服务注册接口

```
srv: /eaios/srv/register
inputs:
    name: string            # 标准服务名称
    entry: string           # 实际 ROS2 service 名称
    metadata: JSON          # 用于选择服务实例
    srv_type: string        # 服务对应的 ROS2 srv 定义
    provider: string
outputs:
    ok: bool
```

服务查询接口

```
srv: /eaios/srv/query
inputs:
    name: string
    filter: JSON            # 用于筛选，例如对任务规划服务设置为
    <-  {"model":"deepseek"}
outputs:
    instances: ServiceInstance[]

ServiceInstance:
```

```
    provider: string
    entry: string
    metadata: JSON
```

### 9.2.3 技能 API

技能（Skill）封装可复用的高层动作逻辑， 在 EAIOS 中分为两类：基础技能与RTDL 技能。基础技能由开发者以静态程序形式提供， 用于承载稳定、确定的能力； RTDL 技能由系统在任务执行成功后，根据规划结果自动固化生成，用于表达可复用的组合动作流程。两类技能在运行时通过统一接口被调度与调用。

技能注册接口

```
srv: /eaios/skl/register
inputs:
    name: string                        # 技能名称
    type: string                        # "basic" | "rtdl"

    start_topic: string                 # 技能启动 topic
    status_topic: string                # 状态反馈 topic

    # 技能本体描述
    entry: string                       # 基础技能入口（basic skill）
    skill_dir: string                   # RTDL 技能目录（rtdl skill）
    main_rtdl: string                   # 主 RTDL 文件名（rtdl skill）

    # 技能输入/输出规范
    start_args: JSON                    # 输入参数
    status: JSON                        # 状态反馈

    # 技能实例筛选元数据
    metadata: JSON                      #
     <-   {"domain":"indoor","capability":["nav","man ip"]}
    provider: string                    # 技能提供者
    version: string                     # 技能版本

outputs:
    ok: bool
    skill_id: string
```

技能查询接口

```
srv: /eaios/skl/query
inputs:
    name: string
    filter: JSON                    # {"domain":"indoor"}, {"type":"basic"}
outputs:
    instances: SkillInstance[]

SkillInstance:
    skill_id: string
    name: string
    type: string                    # "basic" | "rtdl"
    provider: string
    version: string
    start_topic: string
    status_topic: string
    start_args: JSON
    status: JSON
    metadata: JSON
    entry: string                   # basic skill
    skill_dir: string               # rtdl skill
    main_rtdl: string               # rtdl skill
```

技能调用接口

```
# 向 start_topic 发送 JSON，触发技能执行
data:
{
    "skill_id": "skl_close_window_001",
    "params": { "room": "kitchen" }
}

# 从 status_topic 接收技能执行状态
data:
{
    "skill_id": "skl_close_window_001",
    "state": "running",
    "result": {},
    "diagnostics": {}
}
```

技能 **API** 使用实例　假设存在以下 RTDL 代码：

```
# 文件: close_window.rtdl

def skl:close_window(room: str):
    skl:navigate_to(target_label = room)
    srv:semantic_map.update(object = room)
    pose = srv:semantic_map.query_pose(
        object_type = "window",
        parent_room = room
    )
    prm:arm_move_ee(pose = pose)
    prm:gripper.close()
    return True
```

对技能进行注册：

```
srv: /eaios/skl/register
inputs:
    name: "close_window"
    type: "rtdl"

    skill_dir: "/home/user/skills/close_window/"
    main_rtdl: "close_window.rtdl"

    start_args: "{ 'room': 'string' }"
    status: "{ 'state': 'string', 'result': 'any' }"

    start_topic: "/eaios/skl/close_window/start"
    status_topic: "/eaios/skl/close_window/status"

    metadata: "{
     <→  'domain':'indoor','capability':['navigation','manipulation'] }"
    provider: "system_autogen"
    version: "1.0.0"

outputs:
    ok: true
    skill_id: "skl_close_window_001"
```

通过 /eaios/skl/register 注册该技能后，EAIOS 记录技能的入口文件、参数 Schema、元数据与启动/状态Topic，并分配唯一的 skill_id。运行时，当上层任务或其它技能需要关闭窗户时，可通过 /eaios/skl/query按名称与元数据检索可用的技能实例， 获得其启动与反馈Topic

信息。随后， 系统在执行技能时向该技能的 start_topic 发送输入参数（如 room:"kitchen"），技能执行过程中的状态、结果与诊断数据将持续从 status_topic 返回。

### 9.2.4 任务 API

任务是 EAIOS 暴露给用户的最高层接口。用户提交自然语言与可选参数后，任务规划服务将其转换为机器人任务描述语言（RTDL）。

任务提交接口

```
srv: /eaios/task/submit
inputs:
    description: string
    params: JSON
outputs:
    task_id: string
```

任务状态接口

```
srv: /eaios/task/status
inputs:
    task_id: string
outputs:
    status: string       # pending / planning / running / finished / ------
```

任务结果接口

```
srv: /eaios/task/result
inputs:
    task_id: string
outputs:
    result: JSON
```

**RTDL 示例** 行为树式 RTDL 示例：

```
<BehaviorTree ID="patrol_and_close">
    <Sequence>
        <Skill name="speak">
            <Param key="text" value=" 开始巡检">
        </Skill>
```

```
        <ForEach var="room" list="['bedroom', 'kitchen']">
            <Sequence>
                <Skill name="navigate_to">
                    <Param key="target_label" value="{room}">
                </Skill>
                <Skill name="close_window">
                    <Param key="target_label" value="{room}">
                </Skill>
            </Sequence>
        </ForEach>
    </Sequence>
</BehaviorTree>
```

程序式 RTDL 示例：

```
task patrol_and_close() -> bool:
    skl:speak(text: str = "开始巡检")
    rooms: list[str] = ["bedroom", "kitchen"]
    for room: str in rooms:
        skl:navigate_to(target_label: str = room)
        srv:semantic_map.update(object: str = room)
        pose: Pose = srv:semantic_map.query_pose(
            object_type: str = "window",
            parent_room: str = room
        )
        prm:arm_move_ee(pose: Pose = pose)
        skl:close_window(target_label: str = room)
    return true
endtask
```

### 9.2.5 标准原语示例

EAIOS 需要为不同类型的机器人的本体能力定义标准原语， 每个原语均规定标准名称、输入参数、输出参数以及必须包含的元数据字段。原语提供者在注册时须按 JSON 字符串形式提交 `input_schema`、`output_schema` 与 `metadata`。

底盘速度控制（**`base_setvel`**）

```
# 标准原语：base_setvel

# 输入参数（schema，JSON 格式）：
# {
```

```
#    'linear_x': 'float',    # 前向线速度
#    'linear_y': 'float',    # 侧向线速度
#    'angular_z': 'float'    # 角速度
# }

# 输出参数（schema）:
# {}

# metadata 规范（JSON）:
# {
#    'frame': 'string',      # 速度指令参考坐标系，如 'base_link'
#    'max_linear': float,    # 最大线速度
#    'max_angular': float    # 最大角速度
# }

srv: /eaios/prm/register
inputs:
    name: "base_setvel"
    input_schema: "{
        'linear_x': '/robot/base/linear_x',
        'linear_y': '/robot/base/linear_y',
        'angular_z': '/robot/base/angular_z'
    }"
    output_schema: "{}"
    metadata: "{
        'frame': 'base_link',
        'max_linear': 1.0,
        'max_angular': 1.5
    }"
    provider: "agilex_ranger_base_sdk"
outputs: { ok: true }
```

末端位姿控制（**arm_move_ee**）

```
# 标准原语: arm_move_ee

# 输入参数（schema）:
# {
#    'pose': 'geometry_msgs/msg/Pose'
# }

# 输出参数（schema）:
# {}
```

```
# metadata 规范:
# {
#    'arm_type': 'string',    # 机械臂类型，如 '6dof'
#    'workspace': 'string',   # 可达工作空间描述
#    'max_vel': float,        # 最大末端速度
#    'max_acc': float         # 最大末端加速度
# }

srv: /eaios/prm/register
inputs:
    name: "arm_move_ee"
    input_schema: "{
        'pose': '/robot/arm/target_pose'
    }"
    output_schema: "{}"
    metadata: "{
        'arm_type': '6dof',
        'workspace': 'spherical',
        'max_vel': 1.2,
        'max_acc': 2.5
    }"
    provider: "agilex_arm_sdk"
outputs: { ok: true }
```

**RGB_D 采集（capture_rgbd)**

```
# 标准原语: capture_rgbd

# 输入参数（schema）:
# {}

# 输出参数（schema）:
# {
#    'rgb':    'eaios/msg/RGBImage',
#    'depth':  'eaios/msg/DepthImage',
#    'cloud':  'eaios/msg/PointCloud'
# }

# metadata 规范:
# {
#    'resolution': 'string',  # 如 '640x480'
#    'fps': float,            # 帧率
```

```
#     'fov': float,              # 视场角
#     'position': 'string'       # 安装位置
# }

srv: /eaios/prm/register
inputs:
    name: "capture_rgbd"
    input_schema: "{}"
    output_schema: "{
        'rgb': '/camera/color/image',
        'depth': '/camera/depth/image',
        'cloud': '/camera/points'
    }"
    metadata: "{
        'resolution': '640x480',
        'fps': 30,
        'fov': 80.0,
        'position': 'head_mount'
    }"
    provider: "intel_realsense_sdk"
outputs: { ok: true }
```

### 9.2.6 标准服务示例

本小节给出 EAIOS 对若干关键系统服务的标准接口定义。所有服务均以 ROS 2 service 的形式对外暴露能力，服务的内部算法实现则由开发者自由选择。

空间地图服务（**spatial_map**） 空间地图服务提供机器人环境中的几何结构信息，包括二维占据栅格、三维占据图和点云。该地图由开发者实现的 SLAM 或三维重建节点自动维护， 外部模块可以通过查询接口获取其内容。

```
# ROS2 服务类型定义: eaios/srv/GetSpatialMap
# 请求:
#   uint8   map_type          # 0=2D occupancy, 1=3D occupancy, 2=pointcloud
#
# 响应:
#   nav_msgs/OccupancyGrid   occupancy_grid    # 若 map_type=0
#   eaios_msgs/Occupancy3D   occupancy_3d      # 若 map_type=1
#   sensor_msgs/PointCloud2  cloud             # 若 map_type=2
#   builtin_interfaces/Time  stamp
#
# Metadata: supported_types
```

```
# 在 EAIOS 中的注册示例
srv: /eaios/srv/register
inputs:
    name: "spatial_map"
    srv_type: "eaios/srv/GetSpatialMap"
    entry: "/mapping/get_spatial_map"
    metadata: "{
        'supported_types': ['2d','3d','cloud']
    }"
    provider: "slam_toolbox"
outputs: { ok: true }
```

语义地图服务（**semantic_map**） 语义地图服务在空间地图之上构建对象（Object）级表示。对象的检测、分类、关系建模由开发者实现的语义建图节点完成，外部模块可以通过查询接口访问对象列表及其属性。

```
# ROS2 服务类型定义: eaios/srv/QuerySemanticMap

# 请求:
#   string[] types              # 按对象类型过滤，如
↪  ["room","window","robot"]

# 响应:
#   eaios_msgs/Object[] objects

#   Object:
#       string id
#       string type
#       geometry_msgs/Pose pose
#       float32[6] bbox
#       string[] affordance
#       eaios_msgs/SkillInstance[] skls      # 支持的技能实例
#       eaios_msgs/ServiceInstance[] srvs    # 支持的服务实例
#       eaios_msgs/PrimitiveInstance[] prms # 支持的原语实例
#       builtin_interfaces/Time stamp
#
# Metadata: supported_types

# 在 EAIOS 中的注册示例

srv: /eaios/srv/register
```

```
inputs:
    name: "semantic_map"
    srv_type: "eaios/srv/QuerySemanticMap"
    entry: "/semantic_map/query"
    metadata: "{
        'supported_types':
        <→  ['room','window','door','object','person','robot']
    }"
    provider: "semantic_mapper_v2"
outputs:
    ok: true
```

任务规划服务（**task_plan**） 任务规划服务将用户的自然语言描述转换为结构化的 RTDL，供任务管理器解释执行。

```
# ROS2 服务类型定义: eaios/srv/PlanTask
# 请求:
#   string description            # 任务自然语言描述
#   eaios_msgs/Dict params        # 可选参数
#
# 响应:
#   string rtdl                   # RTDL 代码
#   string rtdl_type              # RTDL 风格
#   builtin_interfaces/Time stamp
#
 # Metadata: model, capabilities, rtdl_type

# 注册示例
srv: /eaios/srv/register
inputs:
    name: "task_plan"
    srv_type: "eaios/srv/PlanTask"
    entry: "/planner/plan"
    metadata: "{
        'model': 'qwen2.5-vl',
        'capabilities': ['navigation','manipulation'],
        'rtdl_type': 'BT', # 行为树
    }"
    provider: "vllm"
outputs: { ok: true }
```

方案推演服务（**plan_simulate**） 方案推演服务用于在仿真环境或世界模型中对技能序列进行执行预测，用于进行可行性与安全性检查。

```
# ROS2 服务类型定义： eaios/srv/SimulatePlan
#
# 请求：
#   string rtdl_code          # 需要评估的 RTDL 任务描述
#   string context_id         # 可选，环境上下文快照标识（语义/空间地图版本等）
#   string options_json       # 可选，JSON 字符串形式的评估选项
#                             # 例如：
#                              #   {"check":["collision","limits"],
#                             #     "max_sim_time": 120.0}
#
# 响应：
#   bool   feasible           # 是否满足物理与安全约束
#   string[] violations       # 未满足的约束项，如 ["collision","joint_limit"]
#   string diagnostics_json # 诊断信息
#   builtin_interfaces/Time stamp
#
# Metadata: backend, checks, env_type

# 注册示例
srv: /eaios/srv/register
inputs:
    name: "task_simulate"
    srv_type: "eaios/srv/SimulatePlan"
    entry: "/simulator/evaluate"
    metadata: "{
        'backend': 'webots',
        'checks': ['collision','reachability','limits'],
        'env_type': 'indoor'
    }"
    provider: "local_machine"
outputs: { ok: true }
```

结果反馈服务（**result_feedback**） 结果反馈服务用于在真实执行后，通过多模态模型对执行过程数据进行语义分析，从而判断任务是否被正确完成。

```
# ROS2 服务类型定义： eaios/srv/ResultFeedback
#
# 请求：
#   string task_id            # 任务标识，由任务管理器分配
```

```
#   string execution_id      # 本次执行标识，对应数据采集记录
#   string options_json      # 可选，验证策略（JSON 字符串）
#                            # 例如：
#                            #   {"mode":"video-only",
#                            #
 <→  "required_checks":["all_rooms","all_windows"]}
#
# 响应：
#   bool   success           # 任务是否通过验证
#   string[] missing_goals   # 未完成的目标（如 ["kitchen_window_closed"]）
#   string report            # 文本报告（自然语言或结构化摘要）
#   builtin_interfaces/Time stamp
#
# Metadata: model, mode, modality

# 注册示例
srv: /eaios/srv/register
inputs:
    name: "result_feedback"
    srv_type: "eaios/srv/ResultFeedback"
    entry: "/verifier/verify"
    metadata: "{
        'model': 'qwen2.5-vl',
        'mode': 'video-semantic-check',
        'modality': ['video','state_log']
    }"
    provider: "verification_node"
outputs: { ok: true }
```

# 10 致谢

本白皮书在 CCF 泛在操作系统开放社区技术委员会的指导下完成。

白皮书的编写工作由矽望开源社区[4]组织，编写人员包括：

北京大学：曹东刚、韩喻泷、张照博、李国玮、陈翔、郑子豪、郭耀

清华大学：陈康

上海交通大学：薛栋梁

中国科学院计算技术研究所：李栋

杭州电子科技大学：游理通

矽望开源社区 Robonix 具身智能操作系统研究小组的吴政、刘昊文、徐金阳、侯云龙、刘凯乐、陈衍廷、熊思民、陈正宁、赵龙淳等同学提供了研究素材和进行了实验验证。

还有很多老师同学参与相关讨论，给出了很好的建议，在此一并感谢。

[4] https://syswonder.org

# 参考文献


[1] 中国发展报告 2025[M]. 中国发展出版社，人民出版社, 2025.

[2] Symbolic artificial intelligence[EB/OL]. https://en.wikipedia.org/wiki/Symbolic_artificial_intelligence.

[3] Connectionism[EB/OL]. https://en.wikipedia.org/wiki/Connectionism.

[4] FRANCISCO V. The Embodied Mind[M]. The MIT Press, 1992. 328 pp.

[5] BROOKS R A. Intelligence without Representation[J]. 1991.

[6] PREM E. Elements of an Epistemology of Embodied AI[J]. 1996.

[7] IIDA F, PFEIFER R, STEELS L, et al. Embodied Artificial Intelligence: International Seminar, Dagstuhl Castle, Germany, July 7–11, 2003, Revised Selected Papers: vol. Embodied Artificial Intelligence[M]. Springer Berlin, Heidelberg, 2004.

[8] ChatGPT[EB/OL]. https://openai.com/index/chatgpt.

[9] ASIMO[EB/OL]. https://global.honda/en/ASIMO.

[10] WEISER M. The computer for the 21st Century[J/OL]. IEEE Pervasive Computing, 2002, 1(1): 19- 25. DOI: 10.1109/MPRV.2002.993141.

[11] 梅宏，曹东刚，谢涛. 泛在操作系统：面向人机物融合泛在计算的新蓝海[J/OL]. 中国科学院院刊, 2022, 37(1): 30- 37. http://www.bulletin.cas.cn/publisher/Bulletin%20of%20Chinese%20Academy%20of%20Sciences/journal/Bulletin%20of%20Chinese%20Academy%20of%20Sciences/37/1/10.16418/j.issn.1000- 3045.20211117009. DOI: https://doi.org/10.1641 8/j.issn.1000- 3045.20211117009.

[12] ROS 2 Humble Documentation[EB/OL]. https://docs.ros.org/en/humble/index.html.

[13] dora- rs[EB/OL]. https://dora- rs.ai.

[14] Generative pre- trained transformer[EB/OL]. https://en.wikipedia.org/wiki/Generative_pre- trained_transformer.

[15] BROHANA, BROWN N, CARBAJAL J, et al. RT- 2: Vision- Language- Action Models Transfer Web Knowledge to Robotic Control[EB/OL]. (2023- 07- 28) [2025- 09- 06]. http://arxiv.org/abs/2307 .15818. arXiv: 2307.15818 [cs].

[16] TEAM B R, CAO M, TAN H, et al. RoboBrain 2.0 Technical Report[EB/OL]. (2025- 09- 14) [2025- 10- 12]. http://arxiv.org/abs/2507.02029. arXiv: 2507.02029 [cs].

[17] CAN bus[EB/OL]. https://en.wikipedia.org/wiki/CAN_bus.

[18] EtherCAT[EB/OL]. https://en.wikipedia.org/wiki/EtherCAT.

[19] Fast DDS Documentation[EB/OL]. https://fast- dds.docs.eprosima.com/en/stable/index.ht ml.

[20] URDF - ROS Wiki[EB/OL]. https://wiki.ros.org/urdf.

[21] KDL: Kinematics and Dynamics Library[EB/OL]. https://www.orocos.org/kdl.

[22] MoveIt 2: Motion Planning Framework for ROS 2[EB/OL]. https://moveit.ai.

[23] Nav2: Navigation Stack for ROS 2[EB/OL]. https://nav2.org.

[24] ros2_control Documentation[EB/OL]. https://control.ros.org.

[25] Proportional - integral - derivative controller[EB/OL]. https://en.wikipedia.org/wiki/Proporti onal- integral- derivative_controller.

[26] Model predictive control[EB/OL]. https://en.wikipedia.org/wiki/Model_predictive_control.

[27] KIM M J, PERTSCH K, KARAMCHETI S, et al. OpenVLA: An Open- Source Vision- Language- Action Model[EB/OL]. (2024- 09- 05) [2025- 03- 24]. http://arxiv.org/abs/2406.09246. arXiv: 2406.09246 [cs].

[28] TEAM O M, GHOSH D, WALKE H, et al. Octo: An Open- Source Generalist Robot Policy[EB/OL]. (2024- 05- 26) [2025- 10- 13]. http://arxiv.org/abs/2405.12213. arXiv: 2405.12213 [cs].

[29] LIU S, WU L, LI B, et al. RDT- 1B: A Diffusion Foundation Model for Bimanual Manipulation [EB/OL]. (2025- 03- 01) [2025- 03- 23]. http://arxiv.org/abs/2410.07864. arXiv: 2410.07864 [cs].

[30] BLACK K, BROWN N, DRIESS D, et al. $\pi_0$: A Vision- Language- Action Flow Model for General Robot Control[EB/OL]. (2024- 11- 13) [2025- 11- 18]. http://arxiv.org/abs/2410.24164. arXiv: 2410.24164 [cs].

[31] RADOSAVOVIC I, SHI B, FU L, et al. Robot Learning with Sensorimotor Pre- training[EB/OL]. 2023. https://arxiv.org/abs/2306.10007. arXiv: 2306.10007 [cs.RO].

[32] TAO T, SRIRAMA M K, LIU J J, et al. DexWild: Dexterous Human Interactions for In- the- Wild Robot Policies[EB/OL]. 2025. https://arxiv.org/abs/2505.07813. arXiv: 2505.07813 [cs.RO].

[33] Hyperstack AI Cloud Pricing[EB/OL]. https://www.hyperstack.cloud/gpu- pricing.

[34] HUNG C Y, MAJUMDER N, DENG H, et al. NORA- 1.5: A Vision- Language- Action Model Trained using World Model- and Action- based Preference Rewards[EB/OL]. 2025. https://arxiv.org/a bs/2511.14659. arXiv: 2511.14659 [cs.RO].

[35] NVIDIA, : BJORCK J, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots[EB/OL]. 2025. https://arxiv.org/abs/2503.14734. arXiv: 2503.14734 [cs.RO].

[36] TAN H, HAO X, CHI C, et al. RoboOS: A Hierarchical Embodied Framework for Cross- Embodiment and Multi- Agent Collaboration[EB/OL]. (2025- 06- 05) [2025- 07- 06]. http://arx iv.org/abs/2505.03673. arXiv: 2505.03673 [cs].

[37] SHI L X, ICHTER B, EQUI M, et al. Hi Robot: Open- Ended Instruction Following with Hierarchi- cal Vision- Language- Action Models[EB/OL]. (2025- 07- 15) [2025- 11- 23]. http://arxiv.org/abs /2502.19417. arXiv: 2502.19417 [cs].

[38] INTELLIGENCE P, BLACK K, BROWN N, et al. $\pi_{0.5}$ : A Vision- Language- Action Model with Open- World Generalization[EB/OL]. (2025- 04- 22) [2025- 11- 23]. http://arxiv.org/abs/2504.16054. arXiv: 2504.16054 [cs].

[39] LECUN Y. A Path Towards Autonomous Machine Intelligence Version 0.9.2, 2022- 06- 27[J]. 2020.

[40] OKADA M, TANIGUCHI T. Dreaming: Model- based reinforcement learning by latent imagina- tion without reconstruction[C]//2021 ieee international conference on robotics and automa- tion (icra). 2021: 4209- 4215.

[41] HAFNER D, LILLICRAP T, BA J, et al. DreamerV2: Mastering Atari with Discrete World Models [Z/OL]. 2020. arXiv: 2010.02193 [cs.LG].

[42] HAFNER D, PASUKONIS J, BA J, et al. Mastering Diverse Domains through World Models [Z/OL]. DreamerV3. 2023. arXiv: 2301.04104 [cs.LG].

[43] LECUNY. A Path Towards Autonomous Machine Intelligence[Z]. Meta AI research perspective. Introduces the JEPA concept. 2022.

[44] ASSRAN M, CARON M, MISRA I, et al. Self- Supervised Learning from Images with a Joint- Embedding Predictive Architecture[Z/OL]. I- JEPA. 2023. arXiv: 2301.08243 [cs.CV].

[45] BARDES A, ALAYRAC J B, et al. Mc- jepa: A joint- embedding predictive architecture for self- supervised learning of motion and content features[Z/OL]. V- JEPA. 2023. arXiv: 2307.12698 [cs.CV].

[46] OpenAI. Sora[Z]. Technical/Research announcement. Text- to- video model. 2024.

[47] ZHU Z, WANG X, ZHAO W, et al. Is Sora a World Simulator? A Comprehensive Survey on Gen- eral World Models and Beyond[Z/OL]. 2024. arXiv: 2405.03520 [cs.CV].

[48] LI F F. From Words to Worlds: Spatial Intelligence is AI's Next Frontier[Z]. https://drfeifei.substack.com/p/from- words- to- worlds- spatial- intelligence. Essay on spatial intelligence and world models. 2025.

[49] World Labs. Marble: A Multimodal World Model[Z]. https://www.worldlabs.ai/blog/marble- world- model. Blog post describing the Marble world model. 2025.

[50] DING J, ZHANG Y, SHANG Y, et al. Understanding World or Predicting Future? A Comprehen- sive Survey of World Models[Z/OL]. v2. 2025. arXiv: 2411.14499 [cs.CL].

[51] LONG X, ZHAO Q, ZHANG K, et al. A Survey: Learning Embodied Intelligence from Physical Simulators and World Models[Z/OL]. 2025. arXiv: 2507.00917 [cs.RO].

[52] NVIDIA Isaac ROS[EB/OL]. https://developer.nvidia.com/isaac/ros.

[53] Intel Embodied Intelligence SDK[EB/OL]. https://eci.intel.com/embodied- sdk- docs/content/Intel_embodied_Intelligence_SDK.html.

[54] 邢伯阳，谢佳胤，李永耀，李涛. 全栈开源技术为人形机器人赋能[J]. 中国计算机学会计算会刊, 2025, 1(4): 57- 63.

[55] 上海交大计算机学院 AirOS 具身智能操作系统[EB/OL]. https://news.sjtu.edu.cn/zhxw/20250808/213521.html.

[56] 开源鸿蒙驱动机器人与 AI 产业生态发展，M- Robots OS 正式开源[EB/OL]. https://www.o penatom.org/journalism/detail/ZcFCgJJfqy3Z.

[57] NVIDIA Jetson[EB/OL]. https://www.nvidia.com/en- us/autonomous- machines/embedded - systems.

[58] NVIDIA CUDA Toolkit[EB/OL]. https://developer.nvidia.com/cuda- toolkit.

[59] Intel oneAPI[EB/OL]. https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html.

[60] Intel RealSense Technology[EB/OL]. https://www.intel.com/content/www/us/en/architecture- and- technology/realsense- overview.html.

[61] OpenHarmony[EB/OL]. https://www.openharmony.cn/HomePage.

[62] 上海具识智能科技有限公司 insightOS[EB/OL]. https://www.insightos.cn.

[63] Behavior Tree - Wikipedia[EB/OL]. https://en.wikipedia.org/wiki/Behavior_tree.

[64] PLEXIL[EB/OL]. https://en.wikipedia.org/wiki/PLEXIL.

[65] 2025 BEHAVIOR Challenge - Dataset[EB/OL]. https://behavior.stanford.edu/challenge/data set.html#dataset- statistics.

[66] SUN W, HOU S, WANG Z, et al. DaDu- E: Rethinking the Role of Large Language Model in Robotic Computing Pipeline[EB/OL]. (2024- 12- 02) [2025- 10- 21]. http://arxiv.org/abs/241 2.01663. arXiv: 2412.01663 [cs].

[67] FANY, DING P, BAI S, et al. Long- VLA: Unleashing Long- Horizon Capability of Vision Language Action Model for Robot Manipulation[J]. CoRL, 2025.

[68] RoboBrain- X0: A Unified Cross- Embodiment Vision- Language- Action Model for Token Rea- soning and Action Generation[EB/OL]. https://github.com/FlagOpen/RoboBrain- X0.

[69] COLLABORATION O X E, O'NEILL A, REHMAN A, et al. Open X- Embodiment: Robotic Learning Datasets and RT- X Models[EB/OL]. (2025- 05- 14) [2025- 10- 12]. http://arxiv.org/abs/2310.088 64. arXiv: 2310.08864 [cs].

[70] Webots Robot Simulator[EB/OL]. https://github.com/cyberbotics/webots.

[71] AI M, Collaborators. V- jepa 2: Self- supervised video models enable understanding, predic- tion and planning[Z]. Video representation and prediction with JEPA- style objectives. 2024.

[72] Dual process theory[EB/OL]. https://en.wikipedia.org/wiki/Dual_process_theory.

[73] NVIDIA Isaac Sim[EB/OL]. https://developer.nvidia.com/isaac/sim.

[74] Deadlock (computer science)[EB/OL]. https://en.wikipedia.org/wiki/Deadlock_(computer _science).

[75] Starvation (computer science)[EB/OL]. https://en.wikipedia.org/wiki/Starvation_(compute r_science).

[76] Regression testing[EB/OL]. https://en.wikipedia.org/wiki/Regression_testing.

[77] BAI J, BAI S, YANG S, et al. Qwen- VL: A Versatile Vision- Language Model for Understanding, Localization, Text Reading, and Beyond[EB/OL]. 2023. https://arxiv.org/abs/2308.12966. arXiv: 2308.12966 [cs.CV].

[78] REDMON J, DIVVALA S, GIRSHICK R, et al. You Only Look Once: Unified, Real- Time Object Detection[EB/OL]. 2016. https://arxiv.org/abs/1506.02640. arXiv: 1506.02640 [cs.CV].

[79] Topics vs Services vs Actions[EB/OL].
https://docs.ros.org/en/humble/How- To- Guides/Top ics- Services- Actions.html.

[80] IDL - Interface Definition and Language Mapping[EB/OL].
https://design.ros2.org/articles/i dl_interface_definition.html.