自主机器人软件工程的研究综述[1]

毛新军 1), 2)

[1)] 国防科技大学计算机学院计算科学系，湖南长沙 410073

[2)]国防科技大学计算机学院复杂系统软件工程重点实验室湖南长沙 410073

摘要： 自主机器人是一类运行在开放环境下具有自主行为的信息物理系统，软件是其核心和关键，提供计算、控制、决策等多样化功能，负责驱动机器人安全、灵活和高效地运行。自主机器人软件的开发面临着来自系统自身、外部环境和现实约束等方面复杂性带来的诸多挑战。自主机器人软件工程是一个多学科交叉的新兴研究领域，旨在为自主机器人软件的开发、运行和维护提供工程化的方法、技术和平台支持，其研究与实践近年来引起学术界和工业界的高度关注并取得了长足进步。本文围绕三个方面的研究问题，综述了自主机器人软件工程的研究与实践：（1）自主机器人软件的特点及开发复杂性；（2）自主机器人软件工程的现有研究方向及已有成果；（3）自主机器人软件工程当前研究的局限性及未来研究的机遇。

关键词：自主机器人；软件工程；信息物理系统；环境；软件开发

A Systematic Review on Software Engineering for Autonomous Robot

Xinjun Mao[1), 2)]

[1)] Department of Computing, College of Computer, National University of Defense Technology, Hunan Changsha, 410073

[2)] Key Lab. Of Software Engineering for Complex Systems, College of Computer, National University of Defense Technology, Hunan Changsha, 410073

{xjmao@nudt.edu.cn}

Abstract: Autonomous robotsare a cyber-physical system that operates in open environments and has autonomous behaviors. Software that plays an important and core role in such a system provides various functions of computations, controls and decisions, in order to drive the robotic systems to operate in asafe, flexible and efficient way. To develop software for autonomous robotsis challenged by a number of development complexities resulting from the system itself, operating environments and circumstance constraints. Software engineering for autonomous robot, an emerging interdisciplinary field, aims to offer method, technique and platform supports for the development, running and maintenance of autonomous robot software. It gains increasing attentions from bothacademic and industry,and has made great progresses in recent years. The aim of this paper is to presenta systematic survey on the researches of software engineering for autonomous robot. The contributions are three-fold:(1) in-depthanalysesof the characteristics and the development complexities of autonomous robot software; (2) systematic reviews on the researchbranches andthe progressesof software engineering for autonomous robot, and (3) detailed discussionsaboutthe limitations of existingresearches and the opportunities of future researches.

Keywords: autonomous robot;software engineering; cyber-physical system;environment; software development

本课题得到国家自然科学基金(No. 61532004、61379051)的资助.毛新军，男，1970 年生，博士，教授， CCF 杰出会员（06609S）,主要研究领域为群智软件工程、智能软件技术、多主体系统.

1. 引言

近年来，随着机器人研究的不断深入及应用需求的不断增长，尤其是机器人硬件成本日趋下降及能行计算技术（如大数据、人工智能、云计算等）的快速发展，机器人应用正从工业制造等传统领域逐步延伸到诸如家庭、服务、救助、国防、太空等众多新领域[1][2]。越来越多机器人运行在开放、多样的环境中（包括计算环境、物理环境和社会环境），需与环境持续交互，在没有人接入和干预的情况下，自主决策并实施行为，完成所赋予的任务和目标，这类机器人称之为自主机器人（Autonomous Robot，AR）[2][3]。通常类人形机器人[13]、认知机器人[17]、移动机器人[10]等都属于自主机器人范畴。当前自主机器人已经成为机器人学领域的重要研究分支及研究热点，受到学术界、工业界、政府和组织的高度关注和重视[4]。

自主机器人本质上是一类安全和任务攸关、软件密集型的信息物理系统。它不仅配备有各种物理设备（如运动马达、机械臂、传感器等）以与外部环境相互作用，而且还需要功能强大的软件系统，以完成各种计算（如处理传感数据）、决策机器人的行为（如根据任务规划行为），并控制和驱动机器人在开放环境下的自主、安全、高效和健壮运行，因而软件是自主机器人的核心和关键。自主机器人软件（Autonomous Robot Software，ARS）是一类特定领域的软件系统，它运行在开放、难控、多样的环境中，提供从底层控制到高层决策等一系列功能，需满足实时性、安全性、自主性等非功能需求[2]。这类软件系统的开发需综合运用多学科的知识[5]，解决其特有的软件需求、复杂的运行环境、多样化的现实约束等方面复杂性带来的诸多挑战。例如，由于需与人类发生相互作用，自主机器人软件必须确保机器人行为的安全性，防止行为的实施对人类或者自身产生损害[8]；由于环境的开放性和动态性，自主机器人软件需要有灵活的软件体系结构、具备自主决策的能力，以有效应对环境的变化，确保系统的健壮性和灵活性[6]。

为了迎接上述挑战，近年来自主机器人软件工程（Software Engineering for Autonomous Robot, SEAR）引起了学术界和工业界的高度重视。它借助软件工程领域的诸多思想、方法和成功实践[5]，交叉利用机器人学、人工智能、认知科学、控制学、复杂性科学等学科的知识及成果[6]，为自主机器人软件关键功能的实现、系统的开发和运维等提供有效的方法、技术和平台支持。在过去几年，自主机器人软件工程的研究与实践取得了长足进步，包括软件体系结构[26]、模型驱动开发[9]、程序设计技术[35]、软构件技术[34]、软件开发框架和中间件[10]、软件测试和质量保证[6]等等。目前已有一些文献对自主机器人软件工程的某些方面进行综述分析，如自主机器人系统的形式化确认与验证[7]、移动自主机器人的开发环境[10]、机器人领域的模型驱动开发[9]、机器人系统的软件体系结构[23]，但是缺乏相关的文献系统地分析自主机器人软件工程的研究概况及特点、现有的研究方向及成果、存在的不足及面临的挑战。

本文旨在综述自主机器人软件工程的研究，论文剩余章节组织如下。第二节介绍本文的研究方法，第三节分析自主机器人软件的特点及开发复杂性，第四节评述自主机器人软件工程的主要研究方向及已有成果，第五节讨论了当前研究的局限性、展望未来的研究方向，最后第六节总结全文。

2. 研究方法

本文设计了以下三个研究问题，依此来综述自主机器人软件工程的研究。

– **研究问题 1（RQ1）**：自主机器人软件具有什么样的特点及开发复杂性？

该研究问题旨在关注自主机器人软件的特殊性以及其开发需要解决的问题。对这一问题的研究有助于理解自主机器人软件工程研究与实践的主要驱动力以及需要关注和解决的关键问题。

– **研究问题 2（RQ2）**：自主机器人软件工程研究现状如何？

该研究问题旨在关注自主机器人软件工程的现有研究及成果、它们如何解决自主机器人软件的开发复杂性。对这一问题的研究有助于从全局视角系统地掌握自主机器人软件工程的研究概况及取得的研究成果。

– **研究问题 3（RQ3）**：自主机器人软件工程当前研究有何局限性、未来研究如何开展？

该研究问题旨在关注该领域现有研究工作的不足及展望未来的工作。对这一问题的研究有助于深入分析现有研究工作的局限性以及有待解决的研究问题。

本文基于以下的在线学术文献资源库及检索工具来获取研究所需的文献资料，包括：Google Scholar、ACM Digital library、IEEE Explore、EI Compendex 和 ISI Web of Science，并通过以下关键字 “(mobile robot OR autonomous robot OR humanoid robot OR cognitive robot OR service robot) AND (software OR software engineering)”来检索资料。在阅读所检索出的文献过程中，基于这些文献的参考文献，本文还进一步补充了密切相关的学术文献，包括书籍、著作和学位论文等。

上述检索共获取 310 多篇研究论文，我们阅读了每一篇论文的题目和摘要，并基于以下一组原则对其进行筛选，形成支撑本文研究的参考文献集：（1）分析论文针对的是否为自主机器人，剔除那些针对工业机器人的研究文献；（2）分析论文关注的是否是软件工程问题，剔除那些与此主题不相关的研究文献，如针对机器人的 SLAM、智能控制和算法等方面的研究工作。在机器人产业界，虽然一些企业已成功开发出许多自主机器人系统（如 BostonDynamic 的 Spot 和 Atlas 产品），然而有关这些自主机器人的软件开发技术较少公开发表，导致本文在该方面的参考文献较为缺失。

最终筛选出的 80 多篇文献的出处大致可以分为以下四类：机器人学、软件工程、人工智能和其它。每个类别的具体来源形式包括学术会议论文集、学术期刊、学术著作或学位论文（见表 1）。

表 1. 参考文献出处分析

来源类别	参考论文数目比例	论文主要来源
机器人学	44%	Science Robotics Autonomous Robots IEEE Robotics and Automation Letters IEEE journal on robotics and automation Frontiers in Robotics and AI Robotics and Autonomous Systems Int. Conference on Robotics and Automation Int. Conference on Intelligent Robots and Systems Int. Journal of Advanced Robotic Systems 等
软件工程	21%	Journal of Systems and Software Journal of Software Engineering for Robotics IEEE Software Int. Workshop on Robotics Software Engineering Int. Conference on Software Architecture Int. Conf. on Model-Driven Engineering and Software Development 等
人工智能	6%	Engineering Applications of Artificial Intelligence, Expert Systems with Applications AI Communications International Conference on AAMAS 等
其它	29%	Scientific American ACM Computing Surveys IEEE International Conference on SMC it-Information Technology ACM Symposium on Applied Computing Int. Conference on Cyber-Physical Systems Int. Conf. on System Analysis and Modeling 等

3. 自主机器人软件特点及开发复杂性

自主机器人本质上是由各种软硬件设备所组成的一类复杂混成系统。软件在该系统中扮演着极为重要的角色，提供计算、控制和决策等功能和服务。它通过与机器人硬件设备交互来控制机器人的运行（如通过设备接口来控制机器人移动、调整传感器的观察角度），接收、分析和处理来自于硬件设备的数据（如视觉、红外、雷达等传感器的感知数据）来掌握环境和自身的状况，基于高层任务和目标及所掌握的状况来自主规划并执行机器人的行为，根据任务执行、行为实施及环境变化等情况来适应性地调整和优化机器人的任务和动作。自主机器人系统的自主性及智能性等特征主要通过软件来加以实现。因此，自主机器人软件不仅要完成各种计算输出数据，还有通过决策产生行为来驱动机器人运行；它本质上是一种“粘合剂”，连接了机器人的软硬件设备、外部环境、内部要素、高层任务、底层动作及控制等，并确保整个机器人系统的高效、安全、灵活和健壮运行。

自主机器人是人机物融合系统，自主机器人软件与物理硬件、外部环境甚至人类等要素交织在一起，这类软件系统的开发还受自主机器人系统的现实约束等多因素的影响。本节从外部环境、内部需求、现实约束三个不同视角来分析自主机器人软件的复杂性特点（见图 1）。它们都与自主机器人软件的分析、设计、实现、测试、运行、

维护和演化等密切相关，并对软件工程方法、技术和工具等提出要求和挑战。自主机器人软件工程的研究与实践需要关注和解决的关键问题都可从这三个视角来进行观察、获取和分析。

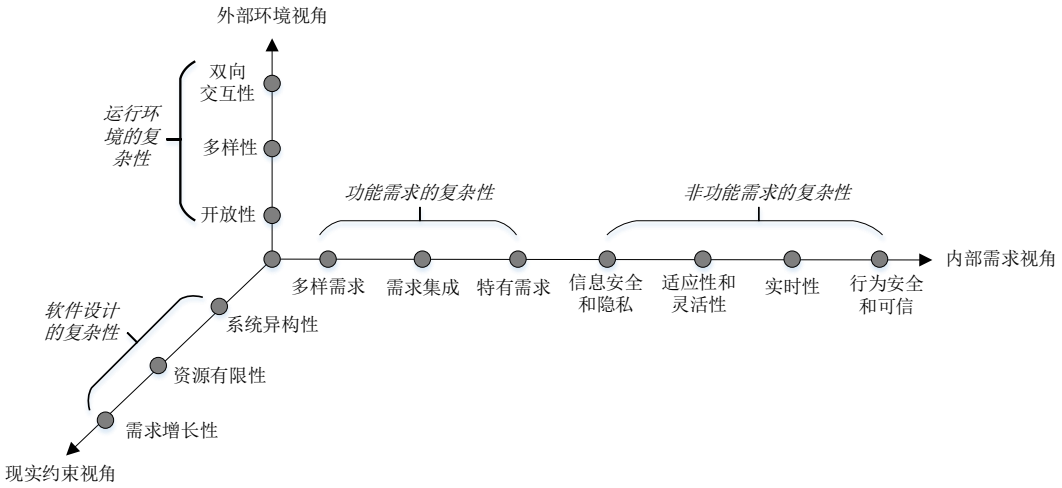

图 1. 从外部环境、内部需求和现实约束三个视角观察自主机器人软件的特殊性及其开发的复杂性

(1) 外部环境视角及运行环境的复杂性

自主机器人都有其特定的运行环境，环境及其变化将对自主机器人的运行产生实质性的影响。相较于工业机器人，自主机器人的运行环境有其特殊性和复杂性，从而增加自主机器人软件开发的难度。

– **多样性**。自主机器人具有多样化的运行环境，包括现实世界的物理环境，如机器人的现实工作场所、场所中的人和设备等；社会环境，如与机器人发生交互作用的其他机器人、人类或者云机器人等；计算世界中的逻辑环境，如自主机器人软件需要访问的云服务、大数据中心等。环境的多样性及其对自主机器人软件的影响增加了自主机器人软件开发的复杂性，如由于环境中存在人类，因而需要解决行为的安全性问题[8]；为了适应物理环境的变化，需要解决任务决策的自主性[6]和灵活性[76]问题。

– **双向交互性**。自主机器人软件需与外部运行环境进行双向交互。一方面，环境的状况及变化会对自主机器人的决策、规划、执行、控制等产生影响。例如，一旦运动前方出现障碍物，机器人需尽快停止移动以免发生碰撞；如果抓取对象的位置发生了变化，机器人需要调整规划以搜寻待抓取对象。因此自主机器人需要通过传感器等部件持续感知环境以掌握其状况和变化信息。另一方面，由软件所控制的自主机器人行为会对环境产生影响，机器人的行为作用在环境上将导致环境状态发生变化。例如，机器人的机械臂抓取了某个物体将会导致环境中该物体的位置信息发生变化。

– **开放性**。自主机器人所在的环境具有开放、动态等特点，环境的边界不确定，环境要素动态加入或者退出，由此导致环境及其变化具有不可预知、难控、不确定、部分可观察等复杂性特点，机器人行为对环境的影响具有不确定性。例如，运行在太空或月球中的机器人事先很难预知所在的外太空信息。在此情况下自主机器人无法获得有关环境的完整、一致、及时和准确的信息，为此自主机器人软件需要根据机器人的任务和目标，控制机器人传感器针对性、按需地观察和感知环境、及时获取所关注的环境及其变化信息，从而有效地支持其自主决策、适应调整和灵活控制。

(2) 内部需求视角及软件需求的复杂性

自主机器人软件自身需要满足一组软件需求以完成各种计算、控制和决策，驱动机器人在开放复杂环境下的运行、实现其任务和目标。不同于其他的软件系统，自主机器人软件的需求有其特殊性和复杂性，完整地实现这些软件需求更为困难。

– **复杂的功能性需求。**自主机器人软件不仅需

要提供多样化的复杂功能和服务，包括底层的控制、管理和分析（如运动控制及合成[11]、物理设备管理、目标对象跟踪、传感数据分析、运行环境建模等），高层的任务规划、目标推理和行为实施（如避障和导航、行为决策、调度和实施、反馈和调整等[30]），还需要集成不同层次的功能，实现它们之间的信息交换和交互协作，确保整个系统的协调运行。尤其是，由于运行环境的多样性、环境及行为执行结果的不确定性、传感数据的失真性和噪音[68]，自主机器人软件需要提供一系列特有的功能需求，如任务规划、行为决策、目标推理[5]、环境监视、动作调度、定位与地图构建 SLAM 等等。

– **复杂的非功能性需求。**环境及功能需求的复杂性给这类软件系统的非功能性需求提出了更高的要求，包括（1）响应环境变化的实时性，自主机器人能快速获取环境状态及其变化，及时采取适当的响应；（2）行为实施的安全性（Safety）和可信性[74]，自主机器人所决策和实施的行为不会对人类或机器人自身产生损害，系统能够正确可靠地运行；（3）任务决策的自主性及理性，自主机器人能够根据所赋予的任务及感知到的环境状态，在没有人类介入和干预的情况下决策、规划和实施有效的行为，最终达成任务的实现；（4）感知数据分析及处理结果的正确性和准确性，经处理和分析后的感知信息能正确和准确地反映环境的实际状态和变化；（5）机器人自身的信息安全性（Security）和隐私保护，自主机器人的数据及信息不会被非法获取，保护自主机器人及其服务对象的个人隐私，等等。这些非功能需求对于自主机器人软件而言极为重要。

(3) 现实约束视角及软件设计的复杂性

当前自主机器人领域的研究与实践存在一系列的现实状况，如机器人平台的多样性、实现技术的异构性、机器人中计算资源的有限性、自主机器人应用需求的快速增长性等。它们对自主机器人软件及其开发提出了各种约束和限制，增加了此类软件系统的设计复杂性。

– **系统异构性。**现有的许多自主机器人系统无论是硬件还是软件都具有异构性特点。不同的软硬件厂商采用不同的实现技术、提供不同的开发接口、产生不同的部件和平台。因此，如何实现异构自主机器人系统的互操作，支持不同软构件之间的交互协同和数据共享，成为自主机器人软件设计及实现必须要考虑并解决的问题。

– **资源有限性。**自主机器人（如空间机器人等）通常具有有限的计算资源（如计算能力、存储空间、网络带宽等）。在开发自主机器人软件时，尤其实现大容量的传感数据处理、任务决策和目标推理等，软件开发人员需要考虑并解决如何基于有限的计算资源来完成复杂的软件功能并确保整个系统的运行性能[45]，。

– **需求增长性。**当前自主机器人软件开发面临二方面快速增长带来的挑战。一个是不断增加的应用需求，自主机器人正越来越多的应用于众多领域，应用数量不断增加；二是不断增加的质量要求，一些新出现的安全攸关应用（如家庭服务、老人看护等）对自主机器人软件的质量，尤其是安全性、隐私性、友好性等方面提出了更高的要求。软件开发人员在确保软件外部质量以满足用户需求的同时，还需充分考虑软件系统内部质量，如可扩展性、可重用性、可验证性、可维护性、持续演化等，并寻求有效软件开发技术，如构件封装、软件重用、自动化开发等，以提高自主机器人软件开发的效率和质量。

自主机器人软件的运行环境、需求及设计的复杂性使得自主机器人软件的开发和运维有其特殊性。它需要综合运用机器人学和软件开发等多学科的知识，考虑自主机器人软件系统与物理系统和社会系统间的交互协同和相互作用及影响，因而需要系统方法学的指导，并解决由于环境和需求等复杂性给软件工程方法和技术带来的诸多挑战。在需求分析阶段，需要分析自主机器人软件与运行环境之间的依赖性、不同软件需求之间的关联性，准确描述自主机器人软件的特有功能和非功能性需求；在软件设计和实现阶段，需要提供有效的软件设计技术（如软件体系结构）和构造技术（如程序模型、编程语言）来支持软件需求（尤其是非功能需求）的实现，解决异构跨平台的互操作问题，并确保自主机器人软件系统

的质量（尤其是正确性和安全性）；在运行和演化阶段，需要提供运行机制、架构、中间件等方面的支持，使得自主机器人软件可有效地感知外部环境，灵活、适应性地应对环境变化，确保系统安全和高效地运行。

4. 自主机器人软件工程的研究方向及成果

本节针对研究问题 RQ2，评述自主机器人软件工程的研究状况，分析现有的研究方向及已有的研究成果，讨论它们如何解决自主机器人软件的开发复杂性问题。

4.1 研究方向概述

当前自主机器人软件工程的研究借鉴了软件工程的诸多工程化开发原则（如模块化、软件封装、关注点分离、信息隐藏等），利用了许多成功的软件开发方法和实践以解决自主机器人软件开发面临的复杂性及带来的挑战，如软件重用、模型驱动开发、软件质量保证技术、程序设计技术及语言、形式化方法等等[7]-[11]。此外，它还吸纳了软件工程领域的最近相关研究及成果，如自适应系统的软件工程（Software Engineering for Self-Adaptive System）[76]、自治系统的软件工程（Autonomy-Oriented Software Engineering）[2]、面向主体的软件工程（Agent-Oriented Software Engineering）[75]等，交叉了其他相关学科的知识和成果，推动自主机器人软件开发特有问题的解决，如利用人工智能领域的多主体系统技术、知识推理和决策技术、机器学习方法等来实现自主机器人的自主任务规划和理性行为决策[6]。

近年来，自主机器人软件工程发展迅速、取得长足进步。根据现有研究工作的特点及要解决的开发问题，可将自主机器人软件工程的现有研究分为以下几个研究方向，图 2 描述了这些研究方向对自主机器人软件开发复杂性及开发运维活动的支持。表 2 描述了围绕这些方向的当前研究概况。

- **自主机器人软件体系结构。**该方向旨在研究高层、有效的软件体系结构及风格以支持自主机器人软件的设计及实现。它通过分解、封装、组织不同特征的软构件，加强它们之间的交互和通讯，以解决自主机器软件的复杂软件需求及现实约束带来的诸多挑战。
- **模型驱动软件开发。**该方向旨在研究如何描述自主机器人软件不同抽象层次的模型，并通过模型转换技术来开发此类复杂软件系统。它不仅有助于促进软件需求的实现，而且可以有效解决由于系统异构性、快速增长需求等现实约束给软件开发技术带来的挑战。
- **自主机器人软件的质量保证方法。**该方向针对自主机器人软件的环境复杂性和需求复杂性给软件质量带来的挑战，寻找有效的质量保证方法，以确保软件开发过程及软件制品的质量，如软件测试、静态分析、模型检验、形式验证等。
- **自主机器人软件的构造技术。**该方向旨在构造层面为解决环境复杂性、需求复杂性、设计复杂性等提供可行的实现技术，如程序模型、程序设计语言、软构件技术、开源软件等，并确保软件质量。
- **自主机器人软件的支撑平台。**该研究方向旨在为自主机器人软件系统的开发、部署、运行和维护提供各类平台的支持，以促进软件需求的实现、简化设计复杂性、应对环境变化，如软件开发框架、软件中间件、开发工具集、集成开发环境等。

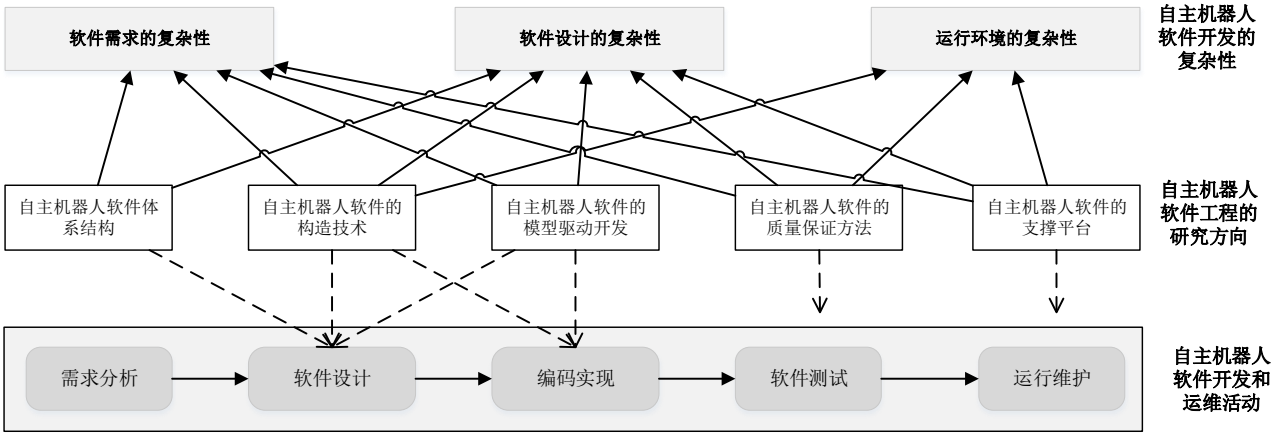

图 2. 自主机器人软件工程的研究方向及其对软件开发复杂性、软件开发和运维活动的支持

表 2. 自主机器人软件工程的研究概况

研究方向	研究内容	研究特点	代表性研究工作
自主机器人软件的体系结构	数据为中心的软件体系结构	集中的数据中心作为架构元素，优点是易于实现，缺点是数据中心易成为通信瓶颈，系统较为脆弱	黑板体系结构及 KAIST 机器人软件体系结构[13]等
	发散式软件体系结构	不同软构件地位对等，采用扁平结构、可重构、发散式的组织和管理模式，优点是可提高系统的健壮性、灵活性和可维护性，不足是难以掌握全局	OROCOS[61], iCub[14], ROS[59], COROS [19]等
	层次式软件体系结构	将软件系统组织为多个不同抽象层次，优点是可简化设计的复杂性，不足是增加了层次间的软构件交互和通讯	包容式[12], CLARAty[60], Choi J[16], LAAIR[21], Jeong[15]等
	混合式软件体系结构	将多种软件体系结构风格集成于一身，优点是可利用不同软件体系结构的优势，不足是增加了系统复杂性及实现难度	NUClear[13], 分布混合层次的控制系统[20], SERA[22], AutoRobot[72]等
自主机器人软件的构造技术	程序模型	采用多种不同软件技术（如 OO、软构件、反应式、基于主体等）来组织、封装和实现自主机器人软件的基本程序单元及其相互间的交互，不同程序模型有各自的优缺点	OO 模型如 CLARAty[60]、player/stage[63]等, 软构件模型如 ROS[68]、BRICS [46]等, 反应式模型[11][12]等，基于主体模型如[2][75]等
	程序设计技术及语言	命令式程序设计技术及语言，用命令式控制语句来实现自主机器人软件的执行步骤、流程和动作，优点是可采用主流的程序设计技术及语言，缺点是软件不具有规划和审慎能力	C 和 C++、Java、Python 等
		描述性程序设计技术及语言，用描述性方式来定义自主机器人软件的知识、任务和动作，优点是具有自主规划能力，不足是易用性不好，软件难以集成	类 Lisp 语言[12]、基于 BDI 的 Agent 编程[17]、SitLog 语言[32]、描述性规则[27]等
		混合式程序设计技术及语言，将命令式与描述性程序设计相结合，以发挥各自的优势，缺点是较为复杂，难以互操作	混合 Python 和 AgentSpeak(L) [25], Julia[30]等
	软构件技术	基于构件技术来封装、实现、选择和组合自主机器人软件，优点是可实现粗粒度的封装和重用、灵活替换，但依赖特定的软构件技术及中间件平台	OROCOS [61], MIRO [77], AutoRobot[72], ESROCOS[62]等
自主机器人软件的模型驱动开发	面向自主机器人的领域知识、软件模型及描述语言	特定自主机器人的元抽象模型及模型描述语言，自主机器人软件的不同抽象层次软件模型，自主机器人领域分析、模式重用、模型及代码的自动生成	高层抽象模型[42]; 领域及体系结构描述语言 AutomationML[37] 、 BRICS[46] 、 SmartTCL[39] 、SSML[41]等; 模型转换和代码生成[36][39][43]等
自主机器人软件的质量保证技术	软件测试	面向自主机器人软件的测试技术、方法和策略，发现软件中的缺陷，提高软件质量	测试驱动的软件开发[14], 自主机器人软件测试方法[56], 多层次测试策略[55]及测试工具[14][58]等
	静态分析	静态分析自主机器人软件模型及代码，在运行前发现问题，减低自主机器人软件测试和运行引发的风险和带来的损害	评估机器人软件质量[57], ROS 节点代码不一致性和潜在错误警告分析[47], 代码分析[71]等
	动态分析、诊断和修复	监视和分析自主机器人运行状况，实现在线实时修复，可减低运行错误带来的风险，提高系统健壮性，但会损害性能	基于模型的缺陷实时检测、自动定位和修复方法[48], ROS 节点运行时检测和分析技术[47]等
	形式化方法	借助形式工具来规约自主机器人系统，应用形式化的确认和验证技术来分析自主机器人软件的性质和特征	安全性验证[50], 基于模型检验的并发特征分析[28], 代码确认和验证[49], 代码自动合成方法[54]等
自主机器人软件的支撑平台	软件开发框架和中间件,	为自主机器人软件的开发提供可重用软件开发包，为软件运行提供中间件，可提高开发效率和质量	ROS[68], CLARAty[60], XBotCore[65], Fawkes [67], Orca[34], YARP[14]等
	软件开发工具及集成环境	辅助开发人员开展自主机器人软件的开发活动，如分析、设计、建模、编码、测试和调试等，可提高开发效率和质量	Choregraphe[64], RobotFlow/FlowDesigner[29], D-Finder[50], Drona [31]等
	混合支撑平台	既提供自主机器人软件开发框架和中间件，还集成软件开发工具集，可为自主机器人软件开发和运维提供系统支持	OROCOS [61], AutoRobot[72], ESROCOS[62], HAROS [71], OPRoS[66], Spica[73]等

4.2 自主机器人软件的体系结构

软件体系结构定义了软件系统的软构件组织和布局以及相互间的交互和通信，它为软件系统的开发提供高层指导。在自主机器人软件工程领域，人们针对自主机器人软件的特殊软件需求，提出了多种软件体系结构及风格，它们大致可分为以下几类：以数据为中心的软件体系结构、发散式软件体系结构、层次式软件体系结构和混合式软件体系结构。自主机器人的每种软件体系结构的设计不仅要考虑欲实现的软件需求（尤其是非功能性需求，如实时性、灵活性、适应性等），还要考虑多态网络互联对自主机器人软件带来的影响，并确保软件设计的质量，促进软件重用，提高软件开发效率[22]。

(1). 数据为中心的软件体系结构

这类软件体系结构源自于 AI 领域的研究成果，其特点是拥有一个集中的数据中心，其他软构件通过对数据中心的操作（如写入和读取数据）

来实现相互间的交互和通讯。总体而言，该体系结构风格易于实现，计算和通讯开销相对较低，但是数据中心可能成为整个软件系统的通信瓶颈，导致系统较为脆弱。典型的成果如基于黑板的软件体系结构，KAIST 机器人开发团队在参加 DARPA 机器人挑战赛中采用了这一软件体系结构来开发机器人软件[13]。

(2). 发散式软件体系结构

在该类软件体系结构中，不同软构件虽然实现不同需求、提供不同功能和服务（如感知、决策、执行等），但是它们处于同一个抽象层次，在系统中的地位是对等的。整个体系结构对应于一个软构件网络，没有集中控制的软构件，采用发散而非集中方式对这些软构件进行组织和管理，一些工作借鉴社会组织的结构模式[18]以及基于 Agent 的分布式结构[20]。总体而言，该体系结构提供了扁平结构、灵活的通讯方式和可重构的组织模式，可促进与多态互联网络中其他软件系统（如云服务和云机器人等)的集成、共享和协作，有助于提高自主机器人软件的健壮性、灵活性和可维护性。当前许多软件开发框架/中间件采用发散式软件体系结构，如 OROCOS[61]、iCub[14]、ROS[59]、COROS [19]等。例如，ROS 中的各个节点（Node）构件高度独立且相互对等，可分布式部署在不同的计算平台上运行，不同节点间不存在集中控制的关系，但可以借助 ROS 提供的事件、Topic 等机制开展交互和协同。

(3). 层次式软件体系结构

该类软件体系结构将整个软件系统组织为多个不同的抽象层次，每个层次负责实现与该层次相对应的软件需求[3]，如底层的传感-效应-处理，高层的决策、规划和适应调整。总体而言，该类体系结构采用层次化组织、软构件分离等手段来设计自主机器人软件，有助于简化高层设计的复杂性，但同时增加了不同层次间的软构件交互和通讯问题。目前有许多的研究及平台采用层次化的软件体系结构，如 Brooks 的包容式软件体系结构[12]、CLARAty[60]由功能层和决策层组成的二层体系结构、Wei 提出的不同符号控制层次结构[17]、Choi J[16]和 LAAIR[21]提出的三层体系结构、Jeong[15]设计的五层结构等等

(4). 混合式软件体系结构

该类软件体系结构将上述多种软件体系结构风格集成于一身，以发挥不同软件体系结构的优势，但必然增加了软件体系结构自身的复杂性及实现难度。如何支持不同层次、不同类别体系结构间的集成、交互和互操作也是一个开放的问题。代表性工作包括：NUClear 软件开发框架[13]、移动机器人的分布混合层次的控制系统[20]、SERA[22]和 AutoRobot[72]等。

4.3 自主机器人软件的构造技术

自主机器人软件需要通过设计和实现，生成可运行的软件系统，并在目标机器人系统和平台上运行。自主机器人软件的构造技术面临着源自外部环境、系统自身和现实约束等诸多复杂性因素及其带来的一系列挑战。目前自主机器人软件工程开展了多方面的研究以支持自主机器人软件的构造，包括程序模型、程序设计技术及语言、软构件技术和开源软件实践。

(1). 程序模型

自主机器人软件的程序模型描述了自主机器人软件的程序组成、封装形式和实现手段。现阶段人们提出了多种程序模型以支持自主机器人软件的构造。

- **结构化程序模型**。自主机器人软件被封装为一组过程或函数，它们间通过过程/函数调用进行交互，并采用结构程序设计技术及语言加以实现。早期人们通常采用这一程序模型来构造自主机器人软件。
- **面向对象程序模型**。自主机器人软件被封装为一组对象类，对象间采用消息传递进行通讯，并借助面向对象程序设计技术及语言加以实现。这一程序模型很早就应用于自主机器人软件开发[24]，当前许多自主机器人软件开发框架采用面向对象程序模型，如 CLARAty[60]、player/stage[63]、Choregraphe[64]、AutoRobot[72]等。
- **软构件模型**。自主机器人软件被组织和封装为一组粗粒度、具有良定义接口和运行时可被替换的软构件，构件间基于接口协议进行交互。这一程序模型有助于促进异构机器人软件系统的互操作、软构件的重用、软件系统的灵活组装，在许多自主机器人支撑平台中得到广泛应用，如 ROS[68]、MIRO[77]、BRICS 模型[46]等。
- **反应式程序模型**。采用刺激-响应机制，用一

组反应式规则、感知输入、反应式控制器等要素来封装自主机器人软件的程序单元，它有助于快速输出行为决策以响应变化，确保自主机器人的行为安全性（如避障）[11][12]。

- **基于主体程序模型**。将自主机器人软件组织和封装为一组自主软件主体，主体间基于事件、消息或主体通讯语言（如 KQML、ACL）等进行交互。每个软件主体内部拥有目标推理、行为决策等部件并封装实现了自主机器人某些方面的功能（如感知、运动、规划等）。该程序模型适合于实现自主机器人的高层功能、支持自主和理性的行为决策[2][75]。
- **专有程序模型**。如 PGCD 程序模型[28]封装了自主机器人的物理状态、构件的几何约束、并发和同步、动态信息等要素。

(2). 程序设计技术及语言

在自主机器人软件工程领域，目前人们基于多种程序设计技术及语言来实现自主机器人软件的程序模型、编写程序代码，并最终生成可运行的软件系统。

- **命令式程序设计技术及语言**

命令式程序设计技术提供了一组机制（如函数调用、消息传递等）和结构（如过程、函数、类等）来实现自主机器人软件的基本模块单元以及它们间的通讯。软件开发人员可利用命令式程序设计语言（如 C 和 C++、Java、Python 等）及其提供的命令式控制语句（如顺序、循环、条件等）等来定义自主机器人软件的执行步骤、流程、动作和行为，适合于实现自主机器人的结构化、面向对象、软构件、反应式程序模型，可有效支持相关计算（如传感数据处理等）和底层控制等功能的开发。当前绝大多数的自主机器人软件开发框架借助命令式程序设计语言作为其基础语言来封装软构件、实现软件功能，如 CLARAty[60]、player/stage[63]、Microsoft Robotics Developer Studio [70]、ROS[68]、Choregraphe[64]等。

- **描述性程序设计技术及语言**

命令式程序设计技术有其固有的局限性，如程序的执行步骤要由程序员在设计阶段预先确定，所开发的软件系统缺乏行为自主性，不具有规划和思考的能力。描述性程序设计技术借鉴 AI 领域的编程技术及语言（如 Prolog、Lisp 和面向主体的程序设计），用描述性方式和语言（如类 Lisp 形式、经典或时序逻辑）来表征自主机器人软件的知识、目标、任务和动作等。自主机器人软件系统在运行时对这些描述性信息进行推理，可自发地产生任务，实现任务规划和行为决策，输出动作，实施自主行为，适合于实现自主机器人软件的反应式和基于主体的程序模型，可有效支持自主机器人软件的高层功能需求的实现。当前有诸多的研究与实践采用描述性程序设计技术及语言来开发自主机器人软件，如用类 Lisp 语言描述反应式规则[12]、采用基于 BDI 的面向主体描述性语言来实现自主机器人软件的目标推理和认知决策[17]、用面向格局（situation-oriented）的描述性逻辑语言 SitLog 来编写服务机器人软件[32]、借助描述性语言来编写自主机器人软件主体的反应式规则[27]等。

- **混合式程序设计技术及语言**

自主机器人研究领域一些学者尝试将命令式程序设计技术与描述性程序设计技术相结合，以发挥各自的优势，产生混合式的程序设计语言，从而为自主机器人软件的编程及混合式程序模型的实现提供有效的技术支持。Fichera 等[25]将基于 Python 的行为编程与基于 AgentSpeak(L)描述性语言的 BDI 主体编程相结合，以支持自主机器人软件开发。Julia[30]将基于 Lisp 函数式程序设计、类型系统和元程序设计等技术相结合，实现自主机器人的底层控制、各类计算和高层反应式行为。

(3). 软构件技术

近年来软构件技术广泛应用于自主机器人软件的研究与实践，以实现自主机器人的软构件程序模型，封装自主机器人的功能（如规划、感知、SLAM、数据分析等），促进软件重用，实现灵活组装。软构件技术使得自主机器人软件开发从传统的编程方式转变为基于构件的组合方式，它有助于促进自主机器人软件体系结构的实现、提高整个软件系统的质量（如可重用性、易维护性、可互操作性等）。现有自主机器人软件开发框架和中间件大多采用软构件技术，支持基于构件的软件开发、部署和运行，如 OROCOS[61]、MIRO[77]、AutoRobot[72]、ESROCOS[62]。软件开发框架支持软构件的不同形式（如 COBRA 对象、软件主体等）封装；中间件可为软构件的运行、通讯、组装和管理提供容器和基础设施，并提供相关软

件工具以支持软构件的选择[33]、生成 (如基于MDD 方法[62])、组装和质量分析。

(4). 开源软件实践

当前越来越多的自主机器人软件支撑平台（如开发框架、运行中间件等）、程序设计设计及支撑环境（如 Julia[30]）采用开源软件的方式，托管在开源社区中，借助群体化的软件开发模式，以提高软件质量，支持长期演化，重进软件重用，建立开源生态。例如 ROS[68]、OROCOS[61]、ESROCOS[62]、XBotCore[65]等都是开源软件。ROS 开源社区吸引了成千上万外部贡献者来支持 ROS 代码的开发和维护；Player/Stage [63]有多达 42,009 次的下载量。

4.4 自主机器人软件的模型驱动开发

自主机器人软件开发需综合运用软件开发知识和机器人领域知识，考虑多种因素（如自主机器人软件的功能性和非功能性需求、机器人系统的约束和限制、支撑机器人软件运行的中间件和基础设施等），产生不同抽象层次的软件制品（包括领域相关模型、平台无关模型、平台相关模型、特定平台的实现代码等），并确保这些软件制品的质量，因而是一项极为复杂的工作。模型驱动开发（Model-Driven Development，MDD）方法可为自主机器人这类信息物理系统的上述软件开发提供有效支持[44]，包括分离开发关注点，建立不同抽象层次的软件模型，通过模型转换技术生成软件模型及程序代码[9]，从而降低开发复杂性、减少开发投入、提高开发效率和质量。

为了支持自主机器人软件的模型驱动开发，人们提出了许多特定于机器人领域的语言以及独立于机器人软件运行平台的语言来描述自主机器人应用的领域信息、表示自主机器人软件的体系结构、建立自主机器人软件的高层抽象模型[42]。这些描述和建模语言通常借助于元建模技术来表征自主机器人应用领域的概念、对象、关系和规则[43][44]，并提供了基于文本或图形的表示方式来帮助开发者或用户直观地描述模型。至今人们已经提出了诸多针对机器人应用的领域特定语言和体系结构描述语言，以描述自主机器人应用或高层设计的相关信息，如应用问题及需求、上下文、领域本体、软件体系结构、静态和可变信息[40] 等，如 MontiArc-Automaton[36] 、AutomationML[37]、BRICS[46]、SmartTCL[39]、SSML[41]、MAUVE[74]、MechatronicUML[44]等。针对自主机器人运行环境及行为执行的不确定性特点，一些建模语言还提供了模型要素以描述自主机器人软件的可变性和适应性等信息。例如，文献[39]提出了可变性建模语言 VML（Variability Modeling Language）来描述自主机器人应对不测情况的可变需求，以确保系统的健壮性和灵活性，提高机器人的运行性能。模型驱动的软件开发环境 HyperFlex[45]支持显式地描述自主机器人软件体系结构中的可变点和可变体，管理和组合自主机器人软件的可变模型及信息。

面向自主机器人的 MDD 方法通常提供了模型转换技术，将高层的模型转换为底层的模型，并最终生成可在目标机器人系统及软件支撑平台（如软件中间件）上运行的程序代码[38]。例如，自主机器人软件开发框架[43]可自动生成机器人的 Java 程序代码，并运行在 LEGO Mindstorms EV3 机器人平台上。通过模型转换生成的大部分程序代码依赖于特定的自主机器人软件开发框架和中间件（如 ROS [36]等），而非实际的机器人系统。这些程序代码通常重用框架中的可重用软构件、遵循框架的软构件规范和实现体系结构、采用框架的开发接口和技术标准。例如，文献[39]中的 MDD 方法输出一组基于面向服务的软件模型 SMARTSOFT、遵循 COBRA 和 ACE 技术规范的软构件代码。

4.5 自主机器人软件的质量保证方法

自主机器人是一类安全和任务攸关的信息物理系统，确保这类系统中软件系统的质量至关重要，包括外部质量（如正确性、安全性、可信性、实时性等）和内部质量（如一致性、可重用性、可维护性、可移植性等）。尤其是，自主机器人系统极为关注诸如行为安全性、响应实时性、信息安全性等方面的质量要求，在这些方面的任何失效或产生的非安全行为都有可能会对环境中的人或者设备以及机器人自身产生损害，此外运行环境的开放性以及行为执行结果的不确定性极大增加了自主机器人软件质量保证的难度[49]，因而如何确保自主机器人软件质量是自主机器人软件工程面临的一项重要挑战[50]。目前在该方向主要有以下几个方面的研究工作。

(1). 软件测试

软件测试可发现软件系统中的缺陷，因而是

软件质量保证的一种手段。自主机器人软件测试面临着诸多方面的挑战，包括环境的动态性及其变化的难控性、复杂的功能需求（如任务规划、行为决策等）、多样化的非功能需求（如行为和信息的安全性、实时性、自主性、灵活性等）以及发生在系统与环境、不同系统层次、不同软构件、软硬件之间的复杂交互。当前围绕自主机器人软件测试有一些研究工作，它们大多侧重于软件测试的方法和策略，以指导测试过程或者支持测试驱动的开发[14]。Laval 等人[56]提出了一种三阶段的自主机器人软件测试方法，包括测试机器人的动作（如先测试感知动作，然后再测试感知和效应动作）、针对环境知识的测试（如先针对拥有准确环境知识的场景进行测试，然后针对缺乏甚至没有环境知识的场景进行测试）、针对环境的动态情况进行测试。该测试方法有助于减少测试过程中机器人对人类和自身带来的损害。Petters 等人[55]提出了多层次测试策略，同样是为了确保机器人软件测试的安全性、减少损害，包括软构件测试、离线测试、在线测试、软件在环路中的测试等等。人们还提供了支持自主机器人软件测试的模拟器，要求先在模拟器上进行测试，通过后再在实际机器人上进行软件测试。一些自主机器人软件开发框架（如 iCub[14]）还提供了软件测试工具集（如 TestIt[58]）以支持自主机器人的软件测试，如测试底层控制代码、开展软构件的强度测试和基于模型的测试等。

(2). 模型和代码的静态分析

在自主机器人软件运行之前对其软件制品（如模型和代码）进行静态分析有助于获取和评估软件制品的质量信息、检测和发现潜在的软件质量问题，如模型与代码间的不一致性、代码编写的不规范性、不安全的程序代码等[71]。软件开发人员可以根据静态分析结果来纠正所发现的自主机器人软件问题，减低自主机器人软件测试可能引发的风险和带来的损害（如不安全的控制代码可能引发损害机器人设备或测试人员），进而提高软件质量。目前在该方面有诸多研究工作，主要针对自主机器人软件的特殊要求，聚焦于高层的软件设计模型和底层的实现代码，分析潜在的质量问题。Neto 等人[57]在欧洲机器人研究计划中专门开发了一个静态分析工作以评估和改进机器人软件的质量。Witte 等人[47]针对自主机器人的 ROS 节点代码及其加载文件，提出了静态分析方法以检查和分析程序代码中潜在的不一致性和错误警告，寻找软件体系结构中的不一致性问题，如错误配置、连接等。一些软件开发框架（如 HAROS[71]）还提供了静态分析工具，以对 MDD 自动生成或者软件开发人员编写的程序代码进行分析和检测。

(3). 动态监视、分析、检测和修复

软件测试和静态分析方法都存在不可遍历问题，即无法通过穷举软件的各种可能执行状况、路径和场景来发现软件中的所有可能问题，因而即使进行了系统的软件测试和全面的静态分析，自主机器人软件任会存在不为测试人员或分析工具所发现的潜在问题[48]。在这种情况下，自主机器人软件自身或者支撑其运行的基础设施（如运行中间件）需对软件运行信息和状况（如状态、动作、流程、结果等）进行监视和分析，据此来检测和发现潜在的软件问题，并对问题或缺陷进行自动解决或修复。目前人们提出了多种方法和技术来支持自主机器人软件的运行时质量保证。一种常见的方法是在运行时对自主机器人需要满足的约束（如安全性条件）等进行动态观察和诊断，据此发现问题及报警。Desai 等人[31]在研究机器人编程框架时，设计了一个运行保障系统，负责检查和分析自主机器人软件在运行时是否满足设计阶段所设定的各种安全性假设，从而确保机器人系统运行的安全性。Steinbauer 等人[48]采用基于模型的方法，针对自主机器人控制软件，提出了一种可实现实时检测、自动定位和修复软件缺陷的方法。Witte 等人[47]通过分析自主机器人软件的 ROS 节点信息、截取运行时节点间的交互信息，构建自主机器人软件运行的节点图，动态检测节点间的不一致性问题。

(4). 形式化方法

在自主机器人研究领域，人们借助于各种数学工具（如时序逻辑、自动机、进程代数、Petri 网、状态机等）及结合机器人领域本体，形式规约自主机器人系统并建立其形式模型，应用确认和验证技术（如模型检验、逻辑推理、定理证明、运行验证等），研究自主机器人软件的性质和特征，如正确性、无死锁、安全性、可达性等[50]。目前，自主机器人软件的形式规约、确认和验证是机器人领域的研究热点[7]，在该方面有诸多的研

究成果。Abdellatif 等人[50]设计了一个高层约束描述语言来形式描述自主机器人软件的功能约束信息，并通过形式化验证方法来分析自主机器人所提供的功能是否满足某些约束，如无死锁等。文献[28]采用模型检验方法来验证基于 PGCD 程序模型的自主机器人软件是否满足一组并发特征，如无死锁、几何不变性（如运行时不会发生碰撞）等。Lyons 等人[49]基于静态分析和贝叶斯网络，提出了一种基于行为的性能验证方法。Meixner 等人[54]提出了一种基于抽象语法树的形式模型，用于确认和验证自主机器人的程序代码。形式化方法还可以用于支持自主机器人软件模型及程序代码的自动生成。Li X 等[53]提出了一种基于模型的代码自动合成方法，它采用时序逻辑语言 CTL 来规约自主机器人的安全需求，基于可验证的模型和时间自动机自动生成基于 ROS 的 C++ 代码。

4.6 自主机器人软件的支撑平台

在自主机器人研究领域，学术界和工业界开发了诸多的支撑平台，支持自主机器人软件的分析、设计、实现、部署、测试、运行和维护，简化自主机器人软件开发的复杂性和难度，提高软件开发效率和质量。根据平台的设计目标及支撑的开发活动，现有自主机器人软件支撑平台可分为以下三类：软件开发框架和中间件、开发工具及环境、混合平台。

(1). 软件开发框架及中间件

自主机器人软件开发框架提供了自主机器人的实现型软件体系结构、软构件模型和技术规范，并以可重用软件开发包或软构件库的形式预先封装和实现了一组自主机器人软件功能，如机器人控制、传感数据分析、任务规划和调度、行为决策、SLAM 等。软件开发人员通过遵循软构件规范、重用软件开发包中的软构件以及对这些软构件的组合来实现自主机器人软件，构造出可运行的软件系统，并部署在软件中间件上运行。通常，自主机器人的软件中间件提供了一组基础设施和服务（如软构件容器、运行引擎和通讯设施、软构件组装等）以支持自主机器人软构件的加载、运行、组织、管理和通讯。

现有的自主机器人软件开发框架及中间件支持多样化和灵活的软件体系结构、允许采用多种不同的程序设计语言来编写自主机器人软件的程序代码。例如，CLARAty[60]采用二层体系结构（包括决策层和控制层）来设计和实现自主机器人软件；ROS[68]允许用 C/C++、 Pythons 等语言来编写节点代码[47]。典型的自主机器人软件开发框架/中间件包括：ROS [68]：一个为人们所普遍接受和应用广泛的自主机器人软件中间件，它独立于具体的机器人型号，支持基于构件的软件开发及多语言编程，提供了灵活多样的交互方式实现跨机器人平台的互操作和协同；XBotCore[65]：一个轻量级、实时的自主机器人软件平台；Fawkes[67]：支持混合体系结构的开发框架；NimbRo-OP[69]：专为 NimbRo 机器人而设计的中间件；CLARAty[60]：一个基于面向对象的通用软件开发框架；YARP[14]：一个支持多平台和多协议的自主机器人软件开发框架。

(2). 软件开发工具及集成开发环境

自主机器人的软件开发工具及集成开发环境提供了一组功能和服务（如可视化建模、代码生成、模型转换和检查、代码质量分析等），共享自主机器人软件开发过程中的相关数据（如质量）及产生的软件制品（如软件模型、程序代码、测试用例等），辅助软件开发人员完成一系列的开发活动，如分析、设计、建模、编码、测试和调试、质量保证等等。通常它们独立于具体的机器人软件体系结构和硬件系统（如机器人类别和型号）以支持高层的分析和设计，但是许多与特定的机器人系统相关联，以支持模型驱动的开发并生成平台相关模型和实现代码。例如，Choregraphe[64]针对 NAO 机器人为其软件的高层设计及代码生成提供支持。当前学术界和工业界提供了诸多的自主机器人软件开发工具及集成开发环境，如 Microsoft Robotics Developer Studio[70]：一个支持自主机器人软件编码、测试和分析的集成开发环境；RobotFlow/FlowDesigner 和 MARIE [29]：移动机器人软件的图形化编程环境；D-Finder[50]：支持对自主机器人软件进行安全性分析的软件工具；Drona[31]：支持自主机器人软件的建模、实现及测试。

(3). 混合支撑平台

还有一些支撑平台既提供了开发框架和中间件，还提供开发工具集或集成开发环境，从而为自主机器人软件的开发、运维和质量保证提供系统和完整的支持。代表性的成果包括：支持自主

机器人实时控制软件开发、基于 CORBA 的框架 OROCOS[61]：基于多主体系统软件架构、支持 MDD 的软件开发框架 AutoRobot[72]；支持代码生成和质量保证的软件开发框架 ESROCOS[62]；提供了静态分析工具以检测和发现 ROS 仓库中代码缺陷的开发框架 HAROS[71]；提供了软件开发工具集以支持自主机器人软件开发的框架 OPRoS[66]；支持模型驱动开发的自主机器人软件开发框架 Spica[73]。

5. 自主机器人软件工程当前研究的局限性及未来研究展望

本节针对研究问题 RQ3，分析自主机器人软件工程当前研究局限性，讨论和展望其未来研究工作。

5.1 现有研究的局限性

尽管近年来自主机器人软件工程受到学术界和工业界的高度关注，但该方向的研究与实践远未成熟。一个具体的证据是：自主机器人软件的应用需求正逐步处于一个快速增长的阶段，现有的软件工程方法和技术尚无法有效解决该领域的“软件危机”，无论对于软件工程师还是机器人专家而言，开发自主机器人软件仍然是一项极具挑战性的工作，当前软件开发效率无法满足日益增长的自主机器人应用开发需求，自主机器人软件的质量要求（尤其是安全性、可信性等）难以得到有效的保证。

在当前主要研究方向上，自主机器人软件体系结构的研究尚无法有效应对环境动态变化、系统灵活适应和长期演化等方面对软件架构提出的要求[18]；自主机器人软件的模型驱动开发缺乏与自主机器人及其应用领域相关的知识图谱或知识库，以有效推动自主机器人软件模型的建立、分析、推理和生成以及与该领域相关的共性问题、解决方法和和设计模式等的识别和重用；自主机器人软件的构造技术需要有效解决跨机器人平台的互操作问题，缺乏面向自主机器人的程序设计技术及语言的支持，需要加强自主机器人软件生态的建设；软件质量保证任然是当前自主机器人软件工程的一个极为薄弱环节，需要结合自主机器人对软件质量的特殊要求，加强在软件测试、安全性保证等方面的研究与实践；自主机器人支撑平台方面需充分借助于人工智能等技术，为自主机器人软件开发及运维提供智能化的支持。

当前自主机器人软件工程的研究与实践尚不足以解决自主机器人软件开发复杂性所带来的诸多挑战，包括如何确保机器人行为和信息的安全性、如何实现有效的自主决策并确保决策结果的理性（Rationality）、如何提升系统的适应性和健壮性以有效应对变化并确保任务的完成等等。作为一个特定领域的复杂软件系统，自主机器人及其软件系统有其特有的运行规律和机理、计算性质和特征，但是关于这些方面，我们缺少相关的基础软件理论支持，对其了解甚少；自主机器人软件的开发还带来了一系列新的工程实践和职业教育问题，如伦理道德、职业教育与认证、系统行为和信息安全认证等，在这些方面现有研究关注不够。考虑到自主机器人系统的构成及涉及内容的多样性和复杂性，自主机器人软件工程的研究与实践需要交叉更多的学科，如 AI、认知科学、控制学理论、复杂性科学、社会组织学、数据科学等，以进一步加强对这一复杂信息物理系统的理解，更为深入地揭示其内在的运行机制和机理，建立起坚实的基础理论，依此来指导在关键软件技术和软件开发方法等方面的研究突破。

更为具体的，自主机器人软件工程的研究并没有充分借鉴和利用软件工程领域的诸多成功实践和和技术来指导自主机器人软件的开发，如软件设计模式、开发方法学等。目前自主机器人软件方面的技术标准化工作进展不大、关注程度不高，势必会影响自主机器人软件工程技术的发展、应用和走向成熟。自主机器人软件需要有更强的自主性、灵活性、适应性、健壮性、可变性和智能性，在这些方面软件工程领域有相应的研究方向及成果，如软件可变性技术、自适应系统的软件工程、面向主体软件工程、智能软件工程等，自主机器人软件工程借鉴这些方向的研究成果甚至交叉融合这些研究方面还不够。自主机器人软件开发是一个复杂的过程，如何将软件工程知识与机器人领域知识将结合，并提供有效的开发策略以促进自主机器人软件的工程化开发仍然是一个开放的研究问题。此外，从运维的角度上看，自主机器人软件的运行中间件、基础设施等缺乏对自主机器人软件运行的自治管理（如自配置、自优化、自保护等）以降低系统运维和管理的复杂性，借助大数据、机器学习等手段来优化系统

的性能等方面的研究工作开展不多。随着越来越多的自主机器人（包括云机器人）部署在互联网或物联网上运行、它们形成集群或者虚拟组织以共同解决问题，自主机器人软件将变得更为复杂，这势必会对自主机器人软件工程的理论及技术研究提出新的挑战。

5.2 未来研究展望

针对现有研究的局限性，自主机器人软件工程需加强以下几个方向研究，以进一步解决自主机器人软件的开发复杂性及其带来的挑战，推动自主机器人软件工程的应用与实践（见表 1）。

（1）自主机器人的基础软件理论。

采用多学科交叉的方式，借助于严格的数学工具（如自动机、时序逻辑、进程代数等）建立自主机器人系统的抽象模型，规约和解释自主机器人在开放环境下的工作原理（如系统与环境的交互，反馈、决策和控制的过程及环路等），研究与揭示自主机器人行为决策和实施的机理及方法（如感知行为和任务行为间的伴随性交互、理性行为的决策机理），分析和验证自主机器人的性质和特征（如安全性、可满足性、可达性等），从而为方法和技术层面的研究奠定坚实的理论基础。

（2）面向自主机器人的领域工程和需求工程

自主机器人的软件需求极为复杂，各个软件需求要素紧密相关且高度依赖于机器人领域知识。为此，需要加强自主机器人领域工程的研究，建立自主机器人的领域本体和知识图谱，识别和分析自主机器人领域中的公共和可重用要素（如设计模式），区分与应用相关的不变部分和可变部分，促进领域知识在软件开发过程中的应用及推理，推动软件开发的自动化及质量保证，加强软件重用；强化自主机器人需求工程的研究与实践，包括软件需求的元抽象、模型及建模语言；需求分析的策略、过程、方法学、工具等，指导自主机器人软件需求的导出、分析、建模和推理，确保需求分析的质量。

（3）自主机器人软件的设计模式

基于自主机器人的基础软件理论、领域工程等方面的研究及成果，分析自主机器人领域重复出现的问题及其解决方法，提炼和识别自主机器人软件的行为模式及有效的组织结构模式，建立面向自主机器人的软件设计模式，以支持自主机器人软件的设计及重用。例如，机器人的运动行为和感知行为之间相互伴随以实现根据任务的需要针对性感知环境变化，促进任务的实现和行为的实施，我们将其称之为伴随行为模式[18]。类似的场景还包括“robot see and robot do”[78], “robot see, feel and act”[79]和“robot see, think and act”[80]等等。

（4）自主机器人软件的开发方法学

自主机器人软件的开发较为复杂，涉及多个环节、要考虑多个要素、综合运用软件开发、机器人和 AI 等领域的多方面知识、开展多种权衡和折中，为此需要研究自主机器人软件的开发方法学，包括软件开发过程、软件建模的方法及语言、软件设计策略、质量保证手段、支撑软件工具等，从而为软件工程师和机器人领域专家提供系统的方法指导、促进重用、确保开发质量。

（5）自主机器人软件的确认和验证

针对自主机器人这一安全和任务攸关的复杂信息物理系统而言，现阶段自主机器人软件质量保证方法的研究与实践有待进一步加强，具体包括：软件测试的模拟和仿真环境，软硬件相结合的自主机器人软件测试技术，针对诸如实时性、健壮性等方面的非功能测试和软件验证技术，针对自主决策和任务规划等特殊功能的软件测试技术，面向行为安全性、信息安全和软件正确性等方面的软件测试和验证技术，以及运行时安全性分析和保证技术等等。

（6）自主机器人软件的技术标准化

相较于工业机器人领域[51]，自主机器人领域的软件技术标准较为薄弱。为此，需要联合多个相关领域（包括软件工程、机器人学、人工智能等）的学术界和工业界，共同推动自主机器人软件的技术标准化工作，以推动互操作等问题的解决以及技术的广泛应用，如软件体系结构、互操作规范、软构件封装和接口、软件中间件和基础设施等方面的规范和标准[52][81]。

当前我国对自主机器人的研究与应用高度重视并进行多个层面的布局和支持，国内的高等院校和研究机构积极投入，产业和企业持续跟进。自主机器人是一个多学科交叉研究领域，在“软件定义一切”的时代，我们需要对计算机软件在其中发挥的关键性作用给予高度的重视。与自主机器人软件工程的整体发展相比较，目前国内学

术界在该领域的研究还不够活跃，产业界缺乏致力于该方向、有影响力的企业、软件产品和机器人系统。导致这种状况的原因是多方面的，既有研究基础薄弱、缺乏长期投入和积累等因素，也有认识和观念上的问题。为此，我国应该加大产学研的投入与合作，积极推动和参与相关技术标准的制定，采用开放的思想建设自主机器人软件生态，加强自主机器人在诸多领域的应用，防止在自主机器人领域出现当前工业软件的困境。

6. 总结

自主机器人软件工程是一个交叉机器人学和软件工程等多个学科的新颖研究方向，旨在针对自主机器人软件这一复杂软件系统，为其开发和运维提供工程化的方法、技术和平台支持。随着机器人的逐渐普及以及应用的逐步拓展，如何快速、高效、低成本地开发出高质量的自主机器人软件成为学术界和工业界共同关心的问题。近年来，自主机器人软件工程领域的研究学者和工程开发人员围绕这一问题开展了诸多的探索和实践，取得了一系列的研究成果。本文综述了自主机器人软件工程的研究状况，取得了以下三方面的贡献。

- 从外部环境、系统自身和现实约束三个不同的视角，分析了自主机器人软件的特点及其开发复杂性，包括：运行环境的复杂性、软件需求的复杂性、软件设计的复杂性，讨论了这些开发复杂性给自主机器人软件工程带来的挑战。
- 对自主机器人软件工程的现有研究进行了系统的梳理，概括了五个主要的研究方向，包括：自主机器人软件的体系结构、构造技术、模型驱动方法、质量保证技术及支撑平台，分析了它们试图解决的自主机器人软件开发复杂性，概述了它们的研究进展及成果。
- 讨论了自主机器人软件工程领域当前研究存在的局限性，结合自主机器人软件开发的复杂性，分析了自主机器人软件工程的未来研究，包括：基础软件理论、需求工程和领域工程、软件设计模式、开发方法学、确认和验证、技术标准化等。

参考文献

[1]. Gates B. A robot in every home[J]. Scientific American, 2007, 296(1): 58-65.

[2]. XinjunMao, software technologies for autonomous robots, communications of the CCF, 2016, 12(7): 60-65.
(毛新军, 自主机器人软件技术, 计算机学会通讯, 2016, 12(7): 60-65)

[3]. Bensalem S, Ingrand F, Sifakis J. Autonomous robot software design challenge[C]//6thWorkshop on Technical Challenge for Dependable Robots in Human Environments. 2008: 17-2.

[4]. Yang G Z, Full R J, Jacobstein N, et al. Ten robotics technologies of the year[J]. Science Robotics, 2019, 4(26): 1-5.

[5]. Pons C, Giandini R, Arévalo G. A systematic review of applying modern software engineering techniques to developing robotic systems[J]. Ingeniería e Investigación, 2012, 32(1): 58-63.

[6]. Kunze L, Hawes N, Duckett T, et al. Artificial intelligence for long-term robot autonomy: A survey[J]. IEEE Robotics and Automation Letters, 2018, 3(4): 4023-4030.

[7]. Luckcuck M, Farrell M, Dennis L A, et. Formal specification and verification of autonomous robotic systems: A survey[J]. ACM Computing Surveys, 2019, 52(5): 1-41.

[8]. Gribov V, Voos H. Safety oriented software engineering process for autonomous robots[C]//18thConference on Emerging Technologies & Factory Automation. IEEE, 2013: 1-8.

[9]. Heineck T, Gonçalves E, Sousa A, et al. Model-driven development in robotics domain: a systematic literature review[C]// Brazilian Symposium on Software Components, Architectures and Reuse. 2016: 151-160.

[10]. Kramer J, Scheutz M. Development environments for autonomous mobile robots: A survey [J]. Autonomous Robots, 2007, 22 (2): 101-132.

[11]. Maoz S, Ringert J O. On the software engineering challenges of applying reactive synthesis to robotics[C]//1stInternational Workshop on Robotics Software Engineering. IEEE, 2018: 17-22.

[12]. Brooks R. A robust layered control system for a mobile robot[J]. IEEE Journal on Robotics and Automation, 1986, 2(1): 14-23.

[13]. Houliston T, Fountain J, Lin Y, et al. Nuclear: A loosely coupled software architecture for humanoid robot systems[J]. Frontiers in Robotics and AI, 2016, 3: 20.

[14]. Natale L, Paikan A, Randazzo M, et al. The iCub software architecture: evolution and lessons learned[J]. Frontiers in Robotics and AI, 2016, 3: 24.

[15]. Jeong I B, Kim J H. Multi-layered architecture of middleware for ubiquitous robot[C]// International Conference on Systems, Man and Cybernetics. IEEE, 2008: 3479-3484.

[16]. Choi J, Cho Y, Choi J, et al. A Layered Middleware Architecture for Automated Robot Services. International Journal of Distributed

Sensor Networks, 2014, 10(5): 201063.

[17]. Wei C, Hindriks K V. An agent-based cognitive robot architecture[C]//International Workshop on Programming Multi-Agent Systems. Springer, 2012: 54-71.

[18]. Mao XJ, Yang S, Huang YH, Wang S. Towards software architecture and accompanying behavior mechanism of autonomous robotic control software based on multi-agent system. Journal of Software, 2020,31(6): 1619-1637.
（毛新军, 杨硕, 黄裕泓,王硕, 自主机器人多智能体软件架构及伴随行为机制, 软件学报, 2020，31(6):1619-1637）

[19]. Koubâa A, Sriti M F, et al. COROS: a multi-agent software architecture for cooperative and autonomous service robots[M]//Cooperative Robots and Sensor Networks. Springer, 2015: 3-30.

[20]. Posadas J L, Poza J L, Simó J E, et al. Agent-based distributed architecture for mobile robot control[J]. Engineering Applications of Artificial Intelligence, 2008, 21(6): 805-823.

[21]. Jiang Y, Walker N, Kim M, et al. LAAIR: A layered architecture for autonomous interactive robots[J]. arXiv preprint arXiv:1811.03563, 2018.

[22]. Garcá S, Menghi C, Pelliccione P, et al. An architecture for decentralized, collaborative, and autonomous robots[C]//International Conference on Software Architecture. IEEE, 2018: 75-7509.

[23]. Ahmad A, Babar M A. Software architectures for robotic systems: A systematic mapping study[J]. Journal of Systems and Software, 2016, 122: 16-39.

[24]. Boyer M, Daneshmend L, Hayward V, et al. An object-oriented paradigm for the design and implementation of robot planning and programming systems[C]//International Conference on Robotics and Automation. 1991: 204-209.

[25]. Fichera L, Messina F, Pappalardo G, et al. A python framework for programming autonomous robots using a declarative approach[J]. Science of Computer Programming, 2017, 139(6): 36-55.

[26]. Kortenkamp D, Simmons R, Brugali D. Robotic systems architectures and programming[M]//Springer Handbook of Robotics. Springer, 2016: 283-306.

[27]. Wang X, Zhang J. RPL: A Robot Programming Language Based on Reactive Agent[C]// 2ndInt. Conference on Electrical, Automation and Mechanical Engineering. 2017: 250-255.

[28]. Banusić G B, Majumdar R, Pirron M, et al. PGCD: robot programming and verification with geometry, concurrency, and dynamics[C]//ACM/IEEE International Conference on Cyber-Physical Systems. 2019: 57-66.

[29]. Côté C, Létourneau D, Michaud F, et al. Code reusability tools for programming mobile robots[C]// International Conference on Intelligent Robots and Systems. IEEE, 2004, 2: 1820-1825.

[30]. Koolen T, Deits R. Julia for robotics: Simulation and real-time control in a high-level programming language[C]//International Conference on Robotics and Automation. IEEE, 2019: 604-611.

[31]. Desai A, Qadeer S, Seshia S A. Programming safe robotics systems: Challenges and advances[C]//International Symposium on Leveraging Applications of Formal Methods. Springer, 2018: 103-119.

[32]. Pineda L A, Salinas L, Meza I V, et al. Sitlog: a programming language for service robot tasks[J]. International Journal of Advanced Robotic Systems, 2013, 10(10): 358.

[33]. Ryadchikov I, Nikulchev E, Gusev A, et al. Engineering Software for a Mobile Robot Motion Control System[C]//IOP Conference Series: Materials Science and Engineering. 2020, 714(1): 012026.

[34]. Brooks A, Kaupp T, Makarenko A, et al. Towards component-based robotics[C]// International Conference on Intelligent Robots and Systems. IEEE, 2005: 163-168.

[35]. Ziafati P. Programming autonomous robots using agent programming languages[C]//International Conference on Autonomous Agent and Multi-Agent System, 2013: 1463-1464.

[36]. Adam K, Hölldobler K, Rumpe B, et al. Modeling Robotics Software Architectures with Modular Model Transformations[J]. Journal of Software Engineering for Robotics, 2017, 8(1): 3-16.

[37]. Hua, Y., Zander, S., Bordignon, M., & Hein, B. From AutomationML to ROS: A model-driven approach for software engineering of industrial robotics using ontological reasoning[C]// International Conference on Emerging Technologies and Factory Automation, 2016: 1-8.

[38]. Broenink, J. F., Groothuis, M. A., Visser, P. M., &Bezemer, M. M. Model-driven robot-software design using template-based target descriptions. Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications, 2010:73-77.

[39]. Schlegel C, Lotz A, Lutz M, et al. Model-driven software systems engineering in robotics: Covering the complete life-cycle of a robot[J]. it-Information Technology, 2015, 57(2): 85-98.

[40]. Ramaswamy A, Monsuez B, Tapus A. Model-driven software development approaches in robotics research[C]//6thInt. Workshop on Modeling in Software Engineering. 2014: 43-48.

[41]. Ramaswamy A, Monsuez B, Tapus A. SafeRobots: A model-driven Framework for developing Robotic Systems[C]//International Conference on Intelligent Robots and Systems. 2014: 1517-1524.

[42]. Nordmann A, Hochgeschwender N, Wrede S. A survey on domain-specific languages in robotics[C]//International Conference on Simulation, Modeling, and Programming for Autonomous Robots. Springer, 2014: 195-206.

[43]. Courbis A L, Luu K, Grondin B, et al. A Model Driven Architecture Framework for Robot Design and Automatic Code Generation[C]//15thChina-Europe International Symposium on Software Engineering Education, 2019: 1-8.

[44]. Pohlmann U. A Model-driven Software Construction Approach for Cyber-physical Systems[D]. Paderborn, Universität Paderborn, 2018.

[45]. Brugali, D., &Valota, M. Software variability composition and abstraction in robot control systems[C]//International Conference on Computational Science and Its Applications, Springer, 2016:358-373.

[46]. Bruyninckx H, Klotzbücher M, Hochgeschwender N, et al. The BRICS component model: a model-based development paradigm for complex robotics software systems[C]// Annual ACM Symposium on Applied Computing. 2013: 1758-1764.

[47]. Witte T, Tichy M. Checking consistency of robot software architectures in ROS[C]// IEEE/ACM International Workshop on Robotics Software Engineering. 2018: 1-8.

[48]. Steinbauer G, Mörth M, Wotawa F. Real-time diagnosis and repair of faults of robot control software[C]//Robot Soccer World Cup. Springer, 2005: 13-23.

[49]. Lyons D M, Arkin R C, Jiang S, et al. Performance verification for robot missions in uncertain environments[J]. Robotics and Autonomous Systems, 2017, 98: 89-104.

[50]. Abdellatif T, Bensalem S, Combaz J, et al. Rigorous design of robot software: A formal component-based approach[J]. Robotics and Autonomous Systems, 2012, 60(12): 1563-1578.

[51]. Tong X, Lei W. A Systematic Analysis of Functional Safety Certification Practices in Industrial Robot Software Development[C]//MATEC Web of Conferences. EDP Sciences, 2017, 100: 02011.

[52]. Ingibergsson J T M, Schultz U P, Kuhrmann M. On the use of safety certification practices in autonomous field robot software development: A systematic mapping study[C]//International Conference on Product-Focused Software Process Improvement. Springer, 2015: 335-352.

[53]. Li X, Wang R, Jiang Y, et al. Formal modeling and automatic code synthesis for robot system[C] //2017 22nd Int. Conf. on Engineering of Complex Computer Systems. IEEE, 2017: 146-149.

[54]. Meixner K, Winkler D, Novák P, et al. Towards Model-driven Verification of Robot Control Code using Abstract Syntax Trees in Production Systems Engineering[C]// International Conference on Model-Driven Engineering and Software Development. 2019: 402-409.

[55]. Petters S, Thomas D, Friedmann M, et al. Multilevel testing of control software for teams of autonomous mobile robots[C]//International Conference on Simulation, Modeling, and Programming for Autonomous Robots. Springer, 2008: 183-194.

[56]. Laval J, Fabresse L, Bouraqadi N. A methodology for testing mobile autonomous robots[C]// International Conference on Intelligent Robots and Systems. IEEE, 2013: 1842-1847.

[57]. Neto T, Arrais R, Sousa A, et al. Applying Software Static Analysis to ROS: The Case Study of the FASTEN European Project[C]//Iberian Robotics conference. Springer, Cham, 2019: 632-644.

[58]. Kanter G, Vain J. Model-based testing of autonomous robots using TestIt[J]. Journal of Reliable Intelligent Environments, 2020: 1-16.

[59]. Basheer M M, Varol A. An Overview of Robot Operating System Forensics[C]// International Informatics and Software Engineering Conference. IEEE, 2019: 1-4.

[60]. Nesnas I A D, Simmons R, Gaines D, et al. CLARAty: Challenges and steps toward reusable robotic software[J]. International Journal of Advanced Robotic Systems, 2006, 3(1): 5.

[61]. Bruyninckx H. Open robot control software: the OROCOS project[C]//International conference on robotics and automation. IEEE, 2001: 2523-2528.

[62]. Wirkus, Malte, Moritz Schilling, and Benjamin Kisliuk. Development of a Control Software for a Planetary Exploration Robot with ESROCOS[C]// Symposium on Advanced Space Technologies in Robotics and Automation. 2019:27-28.

[63]. Vaughan R T, Gerkey B P. Reusable robot software and the player/stage project[M]//Software Engineering for Experimental Robotics. Springer, 2007: 267-289.

[64]. Pot E, Monceaux J, Gelin R, et al. Choregraphe: a graphical tool for humanoid robot programming[C]//18thInternational Symposium on Robot and Human Interactive Communication. IEEE, 2009: 46-51.

[65]. Muratore L, Laurenzi A, Hoffman E M, et al. Xbotcore: A real-time cross-robot software platform[C]//IEEE International Conference on Robotic Computing. 2017: 77-80.

[66]. Jang, C., Lee, S. I., Jung, S. W., Song, B., Kim, R., Kim, S., & Lee, C. H. OPRoS: A New Component‐Based Robot Software Platform. ETRI Journal, 2010, 32(5): 646-656.

[67]. Niemueller T, Ferrein A, Beck D, et al. Design principles of the component-based robot software framework fawkes[C]//International Conference on Simulation, Modeling, and Programming for Autonomous Robots. Springer, 2010: 300-311.

[68]. Estefo P, Simmonds J, Robbes R, et al. The Robot Operating System: Package reuse and community dynamics[J]. Journal of Systems and Software, 2019, 151: 226-242.

[69]. Allgeuer P, Schwarz M, Pastrana J, et al. A ROS-based software framework for the NimbRo-OP humanoid open platform[J]. arXiv preprint arXiv:1809.11051, 2018.

[70]. Kang S C, Chang W T, Gu K Y, et al. Robot Development Using Microsoft Robotics Developer Studio[M]. Chapman and Hall/CRC, 2016.

[71]. A. Santos, A. Cunha, N. Macedo, C. Lourenço, A framework for quality assessment of ROS repositories[C]//International Conference on Intelligent Robots and Systems, IEEE, 2016:

4491–4496.

[72]. Zhe Liu, Xinjun Mao, Shuo Yang. AutoRobot: A Multi-Agent Software Framework for Autonomous Robots[J]. IEICE Transactions on Information and Systems, 2018, E101D(7): 1880-1893

[73]. Baer P A, Reichle R, Geihs K. The spica development framework–model-driven software development for autonomous mobile robots[C]//10thInternational Conference on Intelligent Autonomous Systems. 2008: 211-220.

[74]. Gobillot N, Lesire C, Doose D. A design and analysis methodology for component-based real-time architectures of autonomous systems[J]. Journal of Intelligent & Robotic Systems, 2019, 96(1): 123-138.

[75]. Gascueña J M, Fernández-Caballero A. Agent-oriented modeling and development of a person-following mobile robot[J]. Expert Systems with Applications, 2011, 38(4): 4280-4290.

[76]. Aldrich J, Garlan D, Kästner C, et al. Model-based adaptation for robotics software[J]. IEEE Software, 2019, 36(2): 83-90.

[77]. Hans Utz, Stefan Sablatng, Stefan Enderle, and Gerhard K. Kraetzschmar. Miro – middleware for mobile robot applications. IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures, 2002, 18(4):493–497.

[78]. Bakker P, Kuniyoshi Y. Robot see, robot do: An overview of robot imitation[C]//AISB96 Workshop on Learning in Robots and Animals. 1996: 3-11.

[79]. Fazeli N, Oller M, Wu J, et al. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion[J]. Science Robotics, 2019, 4(26): eaav3123.

[80]. Keall P, Poulsen P, Booth J T. See, think, and act: real-time adaptive radiotherapy[C]//Seminars in radiation oncology. WB Saunders, 2019, 29(3): 228-235.

[81]. Pons C, Pérez G, Giandini R, et al. Applying MDA and OMG Robotic Specification for Developing Robotic Systems[C]//International Conference on System Analysis and Modeling. Springer, 2016: 51-67.

[82]. Ingrand F, Ghallab M. Robotics and artificial intelligence: A perspective on deliberation functions[J]. AI Communications, 2014, 27(1): 63-80.

Dr. Xinjun Mao, born in 1970, Professor, his research interests include crowd-based software engineering, intelligent software technology, multi-agent system, etc.

Background: We are in the era of “software defining everything”, and software plays an increasingly important role to connect physical systems, information systems and social systems. Such trends speed up recently in many domains like robotics, aerospace, aviation, military, etc., in which software as core element of cyber-physical and social-technical systems becomes an indispensible tool.

Robotics is a software-intensive cyber-physical system. Software for such system providescritical functions like controls, computationsand decisions. Currently the applications of robotics are moving from traditional manufacture factorydomains to some emerging domains like services, rescue, military, etc. Robots in such areas typically operate in open environments and should behave autonomously to accomplish its tasks. Autonomous robot has become an important research branch of robotics and gains great attentions from academic and industry communities.

It becomes a common open problem and challenge in both robotics and software engineering fields to develop software for autonomous robot. Software engineering for autonomous robot emerges as a new and interdisciplinary research. In the past years, rapid progresses have been made. However, there lacks of literatures to investigate the particularities and development challenges of autonomous robot software, review the state-of-the-art researches, and discuss the limitations of current researches and the opportunities of future studies. Against the background, this paper aims to give a systematic review on the software engineering for autonomous robot and to answer the above questions. Our studies are expected to help readers to comprehensively and systematically know well this field, and guide their future researches.