ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 60.0.7.1— 2016

РОБОТЫ И РОБОТОТЕХНИЧЕСКИЕ УСТРОЙСТВА

Методы программирования и взаимодействия с оператором

Издание официальное

Москва
Стандартинформ
2016

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Федеральным государственным автономным научным учреждением «Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики» (ЦНИИ РТК) совместно с Институтом прикладной математики им. М.В. Келдыша РАН

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 459 «Информационная поддержка жизненного цикла изделий»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2016 г. № 1845-ст

4 ВВЕДЕН ВПЕРВЫЕ

5 ПЕРЕИЗДАНИЕ. Декабрь 2018 г.

Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)

© Стандартинформ, оформление, 2016, 2018

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины и определения 1
4 Классификация систем программирования 5
4.1 Общие положения 5
4.2 Этапы исполнения программы управления функционированием робота 5
4.3 Классификационные признаки применяемых программных технологий 6
4.4 Система императивного программирования 6
4.5 Система декларативного программирования 6
4.6 Методы построения программ 7
5 Методы программирования роботов, уровни абстракции 7
5.1 Общие положения 7
5.2 Уровень программируемой логики 8
5.3 Низкий уровень процессора и робота 8
5.4 Высокий уровень процессора и робота 8
6 Методы взаимодействия оператора с роботом 8
6.1 Общие положения 8
6.2 Взаимодействие с использованием задающих устройств 9
6.3 Звуковое взаимодействие 9
6.4 Визуальное взаимодействие 9
6.5 Взаимодействие через церебральные интерфейсы 9
6.6 Взаимодействие, использующее иные биометрические показатели 9
6.7 Комплексное взаимодействие 9
7 Организация взаимодействия программного обеспечения и исполнительных механизмов роботов, включая организацию параллельных вычислений 9
7.1 Способы программирования распределенных систем 9
7.2 Протоколы взаимодействия 10
Библиография 11

Введение

Стандарты комплекса ГОСТ Р 60 распространяются на роботы и робототехнические устройства. Их целью является повышение интероперабельности роботов и их компонентов, а также снижение затрат на их разработку, производство и обслуживание за счет стандартизации и унификации процессов, интерфейсов и параметров.

Стандарты комплекса ГОСТ Р 60 представляют собой совокупность отдельно издаваемых стандартов. Стандарты данного комплекса относятся к одной из следующих тематических групп: «Общие положения, основные понятия, термины и определения», «Технические и эксплуатационные характеристики», «Безопасность», «Виды и методы испытаний», «Механические интерфейсы», «Электрические интерфейсы», «Коммуникационные интерфейсы», «Методы программирования», «Методы построения траектории движения (навигация)», «Конструктивные элементы». Стандарты любой тематической группы могут относиться как ко всем роботам и робототехническим устройствам, так и к отдельным группам объектов стандартизации — промышленным роботам в целом, промышленным манипуляционным роботам, промышленным транспортным роботам, сервисным роботам в целом, сервисным манипуляционным роботам и сервисным мобильным роботам.

Настоящий стандарт относится к тематической группе «Методы программирования» и распространяется на все роботы и робототехнические устройства.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

РОБОТЫ И РОБОТОТЕХНИЧЕСКИЕ УСТРОЙСТВА

Методы программирования и взаимодействия с оператором

Robots and robotic devices. Techniques for programming and operator interaction

Дата введения — 2018—01—01

1 Область применения

Настоящий стандарт устанавливает классификацию систем программирования и взаимодействия с оператором роботов и робототехнических устройств.

Требования настоящего стандарта распространяются на системы программирования и взаимодействия с оператором промышленных и сервисных (в том числе медицинских, домашних, инспекционных, учебных и т. п.) роботов и робототехнических устройств.

В настоящем стандарте термин «робот» относится как к роботам (3.1), так и к робототехническим устройствам (3.2), если иное не оговорено особо.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ 19.005 Единая система программной документации. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения

ГОСТ 19781 Обеспечение систем обработки информации программное. Термины и определения

Примечание — При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

Издание официальное

3.1

робот (robot): Исполнительное устройство с двумя или более программируемыми степенями подвижности, обладающее определенным уровнем автономности и способное перемещаться во внешней среде с целью выполнения поставленных задач.

[1], [пункт 2.6]

3.2

робототехническое устройство (robotic device): Исполнительное устройство, обладающее свойствами промышленного или сервисного робота, но у которого отсутствует требуемое число программируемых степеней подвижности или определенный уровень автономности.

[1], [пункт 2.8]

3.3

степень подвижности (axis): Управляемая координата, используемая для определения вращательного или поступательного движения робота.

[1], [пункт 4.3]

3.4

автономность (autonomy): Способность выполнять поставленные задачи в зависимости от текущего состояния и восприятия окружающей среды без вмешательства человека.

[1], [пункт 2.2]

3.5

промышленный робот (industrial robot): Автоматически управляемый, перепрограммируемый манипулятор, программируемый по трем или более степеням подвижности, который может быть установлен стационарно или на мобильной платформе для применения в целях промышленной автоматизации.

[1], [пункт 2.9]

3.6

сервисный робот (service robot). Робот, выполняющий нужную для человека или оборудования работу, за исключением применений в целях промышленной автоматизации.

[1], [пункт 2.10]

3.7

манипулятор (manipulator): Устройство, механизм которого обычно состоит из нескольких звеньев, вращающихся или перемещающихся поступательно друг относительно друга с целью взятия и/или перемещения объектов (деталей или инструмента), как правило, по нескольким степеням свободы.

[1], [пункт 2.1]

3.8

степень свободы (degree of freedom): Одна из координат, максимальное число которых — 6, необходимых для определения движения тела в пространстве.

[1], [пункт 4.4]

Примечание — Перемещение любой механической системы однозначно определяется набором независимых переменных — обобщенных координат, минимальное число которых соответствует числу степеней свободы. Обобщенные координаты определяются при составлении механической модели различными способами, но их число однозначно определено для каждой из систем (например, для твердого тела оно равно шести, для двухзвенного манипулятора с двумя вращательными шарнирами — двум). Число степеней свободы равно полному числу независимых уравнений второго порядка (таких, как уравнения Лагранжа) или половине числа уравнений первого порядка (таких, как канонические уравнения Гамильтона), полностью описывающих динамику системы. Программа управления роботом должна учитывать, что на такую систему могут быть наложены дополнительные кинематические или динамические связи, в которые могут входить переменные моменты, создаваемые программно управляемыми приводами или реакциями опоры, элементов конструкции, среды. В зависимости от перемещения системы связи могут исчезать (например, при отрыве твердого тела от опорной поверхности). Связи определяются системой уравнений, размерность которой не должна превышать число степеней свободы.

3.9

мобильный робот (mobile robot): Робот, способный передвигаться под своим собственным управлением.

[1], [пункт 2.13]

3.10 **транспортный робот** (transport robot): Мобильный робот, предназначенный для перемещения на своей платформе физических объектов.

3.11

язык программирования (programming language): Язык, предназначенный для представления программ.

Примечание — К традиционным языкам программирования процедурного типа относят, как правило, языки для представления программ в виде последовательности предписания.

[ГОСТ 28397—89, таблица 1, пункт 1]

Примечание — В настоящем стандарте используется расширение данного термина в форме «язык программирования компьютера» для того, чтобы отличать его от 3.12.

3.12 **язык программирования действий робота** (robot activity programming language): Язык, предназначенный для представления программ функционирования робота.

Примечания

1 Компьютерная реализация операторов языка программирования действий робота осуществляется с помощью языков программирования компьютера (3.11).

2 Языки программирования действий робота позволяют определять последовательность действий робота в терминах координат, траекторий, выполняемых задач, сенсорных данных, реакций на внешние воздействия и т. п.

3.13 **компьютерная система робота** (robot computerized system): Комплекс аппаратно-программных средств, включающих в себя компьютеры, микропроцессоры и другие вычислительные устройства, снабженные соответствующим программным обеспечением и управляющие роботом и/или обрабатывающие сенсорную информацию.

3.14 **реализация языка программирования** (programming language implementation): Программно-аппаратные средства, обеспечивающие работу компьютерной системы робота согласно тому или иному стандарту языка программирования.

Примечание — Языки программирования компьютеров строятся в соответствии со стандартами языка программирования (в соответствии с его описанием), определяемого как формальная знаковая система, предназначенная для записи компьютерных программ и устанавливающая набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит вычислительное устройство. Один стандарт языка программирования может иметь несколько реализаций, позволяющих преобразовывать программы на языке программирования в последовательности инструкций процессора или микропроцессора.

3.15 **парадигма программирования** (programming paradigm): Реализуемая языком программирования модель вычислений.

Примечание — Язык программирования может одновременно поддерживать несколько моделей вычислений.

3.16

трансляция программы (program translation): Преобразование программы, представленной на одном языке программирования, в программу на другом языке и в определенном смысле равносильную первой.

[ГОСТ 19781—90, таблица 1, пункт 50]

3.17

транслятор (translator): Программа или техническое средство, выполняющие трансляцию программы.

[ГОСТ 19781—90, таблица 1, пункт 38]

3.18

компиляция программы (program compilation): Трансляция программы с языка высокого уровня в форму, близкую к программе на машинном языке.

[ГОСТ 19781—90, таблица 1, пункт 51]

3.19

компилятор (compiler): Программа или техническое средство, выполняющие компиляцию программы.

[ГОСТ 19781—90, таблица 1, пункт 40]

3.20

интерпретация программы (program interpretation): Реализация смысла некоторого синтаксически законченного текста, представленного на конкретном языке.

[ГОСТ 19781—90, таблица 4, пункт 1]

3.21

интерпретатор (interpreter): Программа или техническое средство, выполняющие интерпретацию программы.

[ГОСТ 19781—90, таблица 1, пункт 43]

3.22 **императивное программирование** (imperative programming): Парадигма программирования, которая описывает действия над данными, выражающимися в терминах последовательностей команд.

3.23 **декларативное программирование** (declarative programming): Парадигма программирования, которая описывает действия над данными в виде выражения определений.

3.24

структурное программирование (structured programming): Метод построения программ, использующий только иерархически вложенные конструкции, каждая из которых имеет единственную точку входа и единственную точку выхода.

[ГОСТ 19781—90, таблица 1, пункт 46

3.25

параллельные процессы (parallel processes): Процессы обработки данных, у которых интервалы времени выполнения перекрываются за счет использования различных ресурсов одной и той же системы.

[ГОСТ 19781—90, таблица 1, пункт 85]

3.26

система программирования (programming system): Система, образуемая языком программирования, компиляторами или интерпретаторами программ, представленных на этом языке, соответствующей документацией, а также вспомогательными средствами для подготовки программ к форме, пригодной для выполнения.

[ГОСТ 19781 —90, таблица 1, пункт 21]

3.27

распределенная информационная система (distributed information system): Информационная система, объекты данных и/или процессы которой физически распределяются на две или более компьютерные системы.

[ГОСТ 34.321—96, пункт 2.40]

3.28 **актор** (actor): Отображение в программе сенсора или мехатронного устройства, входящего в состав робота, в виде вычислительной сущности, обменивающейся сообщениями по гибким протоколам в рамках распределенной информационной системы.

3.29 **объект программного управления** (program control object): Совокупность исполнительных устройств и сенсоров, составляющих робот, объединенных компьютерной системой и осуществляющих управление степенями подвижности механической системы.

3.30

сетевая структура (network structure): Множество, частично упорядоченное так, что по крайней мере для некоторых элементов множества существует более одного предшествующего.

[ГОСТ 20886—85, приложение, пункт 4]

3.31

иерархическая структура (hierarchical structure): Множество, частично упорядоченное так, что существует ровно один элемент этого множества, не имеющий предшествующего, а все остальные элементы имеют ровно один предшествующий.

[ГОСТ 20886—85, приложение, пункт 5]

3.32

объектно-ориентированное программирование; ООП (object-oriented programming): Метод построения программ как совокупностей объектов и классов объектов, которые могут вызывать друг друга для выбора и выполнения операций.

Примечание — Объекты состоят из данных и операций над данными.

[ГОСТ 19781—90, таблица 1, пункт 47]

4 Классификация систем программирования

4.1 Общие положения

Программные сенсорные и управляющие системы роботов различают по методам программирования на разных уровнях абстракции (см. 5), по методам взаимодействия с оператором (см. 6), по способам организации программного обеспечения и объектов программного управления (см. 7), по функциональному назначению (см. 8). Данные системы роботов имеют общие классификационные признаки по разделению по времени на этапы исполнения (см. 4.2) и по применяемым программным технологиям (см. 4.3).

4.2 Этапы исполнения программы управления функционированием робота

Программные системы роботов функционируют в трех основных режимах:

- инициализация работы;
- выполнение основного цикла;
- завершение работы.

Этап инициализации работы включает в себя:

- включение основных компонент робота;
- самотестирование и диагностику с информированием оператора об обнаруженных ошибках;
- переход в стартовое состояние программной системы и механических компонент робота.

Этап выполнения основного цикла включает в себя:

- формирование программного управления на основе периодически изменяемой цели работы и/или исполняемой миссии, задаваемых через вышестоящие устройства или от операторов при супервизорном управлении (т. е. отслеживание программных траекторий в режиме контурного управления, перемещение в заданную точку в режиме позиционного управления, анализ состояния среды и робота, выполнение ударных, поисковых и тому подобных движений);
- формирование воздействий на исполнительные органы, включая двигатели, манипуляторы, микрофоны, проекционные устройства и т. п.;
- взаимодействие с исполнительными элементами через драйверы;
- взаимодействие с датчиками обратной связи (сенсорами) и внесение в память данных от них;
- фильтрацию и предобработку получаемой информации, в том числе на основе записей из долговременной памяти;
- мониторинг состояния робота, его компонентов и среды на основе комплексного анализа задач программного управления и показаний датчиков обратной связи;
- интерпретацию данных на основе дедуктивных или иных процедур.

Этап завершения работы включает в себя:

- анализ ошибок, полученных в процессе работы, и их запись в долговременную память для последующего анализа;
- переход в завершающее состояние программной системы и механических компонентов робота.

4.3 Классификационные признаки применяемых программных технологий

Системы программирования классифицируют по следующим признакам:

1) по парадигме программирования:
   а) императивное программирование;
   б) декларативное программирование;
   в) структурное программирование;
2) по наличию встроенных средств параллелизации:
   а) с параллельными процессами;
   б) без параллельных процессов;
3) по наличию поддержки технологии объектно-ориентированного программирования (ООП):
   а) с поддержкой ООП;
   б) без поддержки ООП.

Пример — Наиболее распространенными императивными языками программирования общего назначения с различными встроенными возможностями поддержки ООП в порядке увеличения их выразительных возможностей являются: Алгол 60, Алгол 68, С, С++, С# и т. д.

4.4 Система императивного программирования

Парадигма императивных языков — описание последовательности действий через наборы последовательных команд, списков координат и т. д., изменяющих состояние робота или программы. Такие языки программирования компьютера общего назначения как C, Fortran, Pascal и Python являются типичными представителями данной парадигмы, а программы на них являются ни чем иным как последовательностями инструкций. Аналогично устроены и специализированные языки программирования действий роботов, например, язык KRL, разработанный фирмой KUKA для своих роботов.

В низкоуровневых языках программирования компьютера (например, Ассемблер) состоянием могут быть память, регистры и флаги, а инструкциями — команды, которые поддерживает целевой процессор. В языках более высокого уровня (C, Fortran , Pascal) появились переменные, а инструкции представляют комплексные операции, занимающие сотни строк на языке Ассемблера. В современных высокоуровневых языках программирования компьютера (Python, C#, Lisp) манипулировать разрешается лишь переменными. Как правило, в таких языках (императивные, общего назначения) часто используются оператор присваивания и переменные, что затрудняет отладку программ и приводит к возникновению специфических ошибок. Однако такие языки оказываются значительно ближе декларативных к внутренней архитектуре процессоров, за счет чего средства разработки для них появились раньше и организованы они проще.

4.5 Система декларативного программирования

Парадигма декларативных языков — описание желаемого результата и условий его получения. Разделяют два подхода к построению систем декларативного программирования — логическое программирование и функциональное программирование.

Логическое программирование — это метод построения программы как совокупности логических правил с предварительно определенными алгоритмами для обработки входных данных программы в соответствии с ее правилами. Примером такого подхода является язык Prolog, основанный на исчислении предикатов и автоматическом доказательстве теорем.

В функциональном программировании процесс вычислений трактуется как вычисление значений функций в математическом понимании. Поэтому в программе отсутствуют состояния, а значит и переменные. Это позволяет существенно упростить разработку параллельных программ и приблизить текст программы на языке программирования к математической записи. Отсутствие присваиваний и замена их на операции порождения новых данных приводят к необходимости постоянного выделения памяти, а нестрогая модель вычислений приводит к непредсказуемому порядку вызова функций. Этот

подход наиболее часто используется при решении задач искусственного интеллекта. Примерами языков функционального программирования являются Haskell и Lisp.

Описанные выше подходы могут использоваться одновременно, например, язык ассемблера — для программирования микроконтроллеров робота, язык С — для создания программного обеспечения бортового компьютера и язык Prolog — для построения системы принятия решений.

4.6 Методы построения программ

Непосредственно при программировании могут использоваться приведенные ниже методологии или их комбинации.

4.6.1 Структурное программирование

Структурное программирование (по ГОСТ 19781) представляет метод построения программ, использующий только иерархически вложенные конструкции, каждая из которых имеет единственную точка входа и единственную точку выхода. В структурном программировании используются три вида структур, связанных с передачей управления: последовательная, условного перехода и циклическая.

4.6.2 Объектно-ориентированное программирование

Объектно-ориентированное программирование (по ГОСТ 19781) представляет метод построения программ в виде совокупности объектов и классов объектов, которые могут вызывать друг друга для выбора и выполнения операций.

Примечание — Объекты состоят из данных и операций над данными.

4.6.3 Агентно-ориентированное программирование

Агентно-ориентированное программирование представляет разновидность представления программ, в которой основополагающими концепциями являются понятия агента и его ментальное поведение, зависящее от среды, в которой он находится.

4.6.4 Компонентно-ориентированное программирование

Компонентно-ориентированное программирование представляет методологию, основанную на понятии компонента — независимого модуля программного кода, предназначенного для повторного использования и развертывания, который реализуется в виде множества языковых конструкций (например классов в объектно-ориентированных языках), объединенных по общему признаку и организованных в соответствии с установленными правилами и ограничениями.

4.6.5 Обобщенное программирование

Обобщенное программирование представляет методологию, заключающуюся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание.

4.6.6 Графическая система Р-технологии программирования

Графическая система *Р*-технологии программирования (по ГОСТ 19.005 представляет программу в виде *Р*-схемы — нагруженного по дугам ориентированного графа, изображаемого с помощью вертикальных и горизонтальных линий и состоящего из структур (подграфов), каждая из которых имеет только один вход и один выход. Данная методология является расширением структурного программирования на двумерное пространство.

5 Методы программирования роботов, уровни абстракции

5.1 Общие положения

Конечным результатом исполнения программы роботом является изменение его пространственных или информационных параметров.

Программирование может осуществляться на следующих уровнях абстракции:

- уровень программируемой логики (наиболее низкий уровень абстракции);
- низкий уровень процессора (микропроцессора) или робота;
- высокий уровень процессора или робота.

Программы, составленные на более высоком уровне абстракции с помощью систем программирования, транслируются на более низкие уровни, что в конечном итоге приводит к изменению необходимых параметров робота.

5.2 Уровень программируемой логики

Программирование на уровне программируемой логики может осуществляться с использованием программируемых логических интегральных схем (ПЛИС), заказных ИС, электронных устройств с жесткой логикой или аппаратных средств, например с использованием наборной панели или с перестановкой упоров в роботах с цикловым управлением.

5.3 Низкий уровень процессора и робота

Программирование на низком уровне универсального или специализированного процессора осуществляется на языке Ассемблера или макроязыке.

Целью программирования на низком уровне робота, в отличие от программ, используемых на низком уровне процессора, является не изменение состояния процессора, а изменение состояния робота. Программы низкого уровня для программирования действий робота пишутся на языках программирования компьютера, позволяющих непосредственно управлять портами ввода-вывода компьютерной системы робота. Как правило, данный способ программирования робота используется разработчиком робота и скрыт от конечного пользователя.

Программы низкого уровня, управляющие действиями робота, используются разработчиком робота в качестве основы для реализации операторов языка более высокого уровня (см. 5.4), который предоставляется пользователю для программирования конкретных приложений. Если разработчик не предполагает программирование действий робота пользователем, а сам определяет весь диапазон возможных действий робота, то программу функционирования робота на низком уровне он записывает в постоянную память системы управления робота, к которой у пользователя нет доступа. В частности, таким способом может быть реализовано управление роботом в копирующем режиме или в режиме целеуказания.

5.4 Высокий уровень процессора и робота

Программирование на высоком уровне процессора осуществляется с использованием языков императивного или декларативного программирования, таких как C, Fortran, Pascal, Python, C#, Lisp, Prolog и другие.

Программирование на высоком уровне робота осуществляется с использованием проблемно-ориентированных языков. Большинство крупных производителей роботов используют собственный язык программирования действий робота, обеспечивающий задание:

- жесткой последовательности движений;
- последовательности движений в зависимости от данных, получаемых от сенсорной системы или от аппаратных устройств типа управляющих панелей или джойстиков;
- последовательности действий в терминах решаемых задач.

Примерами языков программирования действий робота высокого уровня являются RAPID (ABB, Швеция), PDL2 (Comau, Италия), Karel (Fanuc, Япония), AS (Kawasaki, Япония), KRL (KUKA, Германия), Inform (Yaskawa, Япония) и др.

Следует отметить, что языки программирования действий робота высокого уровня, в свою очередь, могут классифицироваться в зависимости от степени подробности описания требуемых действий робота, т. е. от степени автономности робота, начиная от простейших языков, подробно описывающих последовательность движений степеней подвижности робота, и до языков, определяющих выполняемую роботом задачу или миссию, включая взаимодействие с другими роботами и людьми.

6 Методы взаимодействия оператора с роботом

6.1 Общие положения

Методы взаимодействия оператора с роботом определяются теми программно-аппаратными устройствами ввода-вывода информации, которые использует оператор для управления роботом и программирования действий робота.

6.2 Взаимодействие с использованием задающих устройств

Данный метод взаимодействия осуществляется с помощью:
- кинематически подобных задающих манипуляторов;
- джойстиков;
- кнопок/тумблеров;
- устройств ввода информации в компьютер;
- педалей;
- силомоментной обратной связи.

6.3 Звуковое взаимодействие

Данный метод взаимодействия осуществляется с помощью:
- звуковых сигналов;
- речевых команд;
- синтеза речи;
- распознавания речевых команд.

6.4 Визуальное взаимодействие

Данный метод взаимодействия осуществляется с помощью:
- жестов;
- визуальных сигналов;
- распознавания выражения лица (мимики);
- положения глазных яблок;
- отображения действий робота на экране монитора через графический интерфейс пользователя (GUI).

6.5 Взаимодействие через церебральные интерфейсы

Взаимодействие через церебральные интерфейсы (нейрокомпьютерный интерфейс или интерфейс мозг — компьютер) заключается в подключении управляющих контуров робота непосредственно к центральной нервной системе человека за счет электрического или электромагнитного контакта. При этом осуществляется считывание мозговых волн различной природы для использования в контуре управления.

6.6 Взаимодействие, использующее иные биометрические показатели

К иным биометрическим показателям относятся:
- электрическое напряжение в мышцах человека;
- удельная электрическая емкость кожи человека;
- проводимость кожи (миограммы).

6.7 Комплексное взаимодействие

Данный метод взаимодействия основывается на одновременном использовании разных взаимодействий, например:
- элементы эмоционального коммуникативного поведения;
- мультимодальные интерфейсы.

7 Организация взаимодействия программного обеспечения и исполнительных механизмов роботов, включая организацию параллельных вычислений

7.1 Способы программирования распределенных систем

Сетевое взаимодействие строится на основе задания основных видов топологии сети:
- без заданной топологии — программно регистрируется единственный робот;
- без явной топологии — роботы не объединены в структуру, но могут взаимодействовать, обмениваться данными по схеме «любой с любым». Таким образом, в конкретные моменты времени

эта топология может становиться как иерархической — в виде дерева связей, в котором все элементы связаны единственным образом и при этом назначена корневая вершина, так и в виде произвольной графовой,

- с иерархической топологией — роботы объединены в иерархическую структуру в виде дерева;
- с графовой топологией — роботы объединены в графовую структуру.

При любых топологиях возникает задача сетевого взаимодействия. Для описания взаимодействия роботов в сети может быть использована вычислительная модель акторов. Мультиагентные технологии являются частным случаем модели вычислений акторов, с дополнительной интеллектуализацией.

Каждый сенсор и устройство, а также робот в целом, отображаются в программе в актора. Актор является вычислительной сущностью, которая в ответ на полученное сообщение может одновременно:

- отправить конечное число сообщений другим акторам;
- создать конечное число новых акторов,
- выбрать тип поведения, который будет использован для следующего сообщения в свой адрес.

Может существовать произвольная последовательность вышеописанных действий, и все они могут выполняться параллельно.

7.2 Протоколы взаимодействия

Для обеспечения непрерывного мониторинга состояния робота и его компонент, а также анализа после выполнения основных работ, необходимо реализовать ведение протокола работы и обеспечить доступ к нему как во время выполнения основного цикла управления, так и по завершении работ.

Протокол работы (или шлейф данных) должен включать в себя:

- метку времени,
- данные об исполняемой роботом команде или группе команд,
- внутреннее состояние робота, выраженное в композиции состояний его подсистем и сенсорных данных.

При этом программная система может быть организована с параллельными вычислениями. Параллелизм может быть и неявным.

На уровне языков программирования поддержку параллелизма реализуют следующими способами:

- на уровне операционной системы: потоки и процессы;
- на уровне языка: библиотеки, например pthread, MPI;
- на уровне языка: директивы компилятора, например OpenMP в C++;
- на уровне языка: парадигма, например реализация модели акторов в Erlang;
- на уровне алгоритма управления.

Библиография

[1] ИСО 8373:2012 (ISO 8373:2012) Роботы и робототехнические системы. Словарь (Robots and robotic devices — Vocabulary)

УДК 621.865.8:007.52:62-529:006.354 ОКС 25.040.30 ОКП 38 8600

Ключевые слова: роботы, робототехнические устройства, промышленные роботы, сервисные роботы, программирование, языки программирования, методы программирования роботов, взаимодействие робота с оператором

Редактор *Е.В. Лукьянова*
Технический редактор *И.Е. Черепкова*
Корректор *Р.А. Ментова*
Компьютерная верстка *А.Н. Золотаревой*

Сдано в набор 11.12.2018. Подписано в печать 24.12.2018 Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 1,86. Уч.-изд. л. 1,68.
Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
gost.e-ecolog.ru gost.e-ecolog.ru

Создано в единичном исполнении ФГУП «СТАНДАРТИНФОРМ»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru