Вестник Московского авиационного института. 2022. Т. 29. № 2. С. 197-210.
Aerospace MAI Journal, 2022, vol. 29, no. 2, pp. 197-210.

Научная статья
УДК 004.432.4
DOI: 10.34759/vst-2022-2-197-210

ПРОГРАММИРОВАНИЕ ДОВЕРЕННОЙ ПАМЯТЬ-ЦЕНТРИЧЕСКОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДВИЖЕНИЕМ РОБОТОТЕХНИЧЕСКИХ И МЕХАТРОННЫХ СИСТЕМ

Александр Александрович Зеленский[1] ✉ , Станислав Павлович Ивановский[2], Юрий Владимирович Илюхин[3], Андрей Армович Грибков[4]

[1,2,3,4]Московский государственный технологический университет “СТАНКИН”, Москва, Россия
[1]zelenskyaa@gmail.com ✉
[2]s.ivanovskiy@stankin.ru
[3]y.ilyukhin@stankin.ru
[4]a.gribkov@stankin.ru

Аннотация. Дается обоснование необходимости развития систем управления движением промышленных роботов, станков с ЧПУ и других мехатронных систем. Определяются требования по обеспечению доверия к таким системам с точки зрения функциональной надежности и информационной безопасности. Предлагается перспективная память-центрическая архитектура систем управления движением, обеспечивающая решение основных проблем в области функциональности и доверия к системам управления. Исходя из специфики систем управления с память-центрической архитектурой устанавливаются базовые требования к программированию таких систем управления. Согласно этим требованиям язык программирования должен быть предметно-ориентированным, декларативным (с элементами функционального и логического языка), интерпретируемым (или языком ассемблера), а программа должна реализовывать модель акторов и обеспечивать повышение доверия к системе управления движением. Для удовлетворения заданным требованиям авторами создан предметно-ориентированный декларативный интерпретируемый язык модульной цифровой системы, определены механизм реализации акторной модели посредством метапрограммирования, а также инструменты повышения доверия к системе управления за счет децентрализации управления и локализации данных.

Финансирование: работа выполнена при поддержке Минобрнауки России в рамках выполнения государственного задания (FSFS-2021-0004)

Ключевые слова: промышленный робот, программа, мехатронный, система управления движением, доверие, память-центрический, предметно-ориентированный, декларативный, актор, метапрограммирование

Для цитирования: Зеленский А.А., Ивановский С.П., Илюхин Ю.В., Грибков А.А. Программирование доверенной память-центрической системы управления движением робототехнических и мехатронных систем // Вестник Московского авиационного института. 2022. Т. 29. № 2. С. 197-210. DOI: 10.34759/vst-2022-2-197-210

Original article

PROGRAMMING A TRUSTED MEMORY-CENTRIC MOTION CONTROL SYSTEM FOR ROBOTIC AND MECHATRONIC SYSTEMS

Aleksandr A. Zelenskii[1] ✉, Stanislav P. Ivanovskii[2], Yurii V. Ilyukhin[3], Andrei A. Gribkov[4]

[1,2,3,4]Moscow State University of Technology "STANKIN",
Moscow, Russia
[1]zelenskyaa@gmail.com ✉
[2]s.ivanovskiy@stankin.ru
[3]y.ilyukhin@stankin.ru

Abstract

The article substantiates the need for the development of motion control systems for industrial robots, CNC machines and other mechatronic systems, defines the requirements for ensuring trust in such systems from the viewpoint of functional reliability and information security. One of the most up-to-date trends in the development of motion control systems for digital production is a significant expansion of their functionality for managing complex multi-coordinate nonlinear objects in real time. Practical meeting of the requirements for improving and ensuring the trust of motion control systems of industrial robots, CNC machines and other mechatronic systems can be achieved by improving the architecture of motion control systems, in particular through the application of memory-centric architecture of motion control systems. On the assumption of the specifics of control systems with memory-centric architecture, basic requirements for programming such control systems can be set. According to these requirements, the programming language should be:

- subject-oriented and specialized for motion control;
- declarative with elements of functional and logical language, optimal for setting algorithms of operation, i.e. for distributing tasks between autonomous functional modules of the control system;
- interpreted (or assembly language), ensuring the speed and compactness of the program code, as well as optimal use of shared memory resources of the control system when running in real time.

In addition, the program in the language being defined should implement the model of actors and ensure confidence increasing in the motion control system. To meet the specified requirements, the authors created a domain-oriented declarative interpreted language of a modular digital system. The key elements of the language are a set of syntactic elements, as well as application programming interfaces built from syntactic elements of the language and serving for integration into the language of external libraries (in the same or other languages). The program in language includes the following basic elements: operators, structures and expressions formed from syntactic elements of the language; actors formed as instances of additional programs emulated by the (main) program at startup or during the process of running.

The motion control system, programmed in the language, consists of four main structural components:

- A human-machine interface, through which the program code generated by the human operator, describing the algorithm of operation of the equipment, as well as a configuration file that provides program configuration for the tasks being formulated;
- A central processor responsible for the overall management of the system and distribution of tasks;
- Functional modules, performing data processing of sensitization, computations and control of regulators of actuating devices;
- Communication networks, ensuring communication between the structural elements of a computer, as well as with external devices.

As the result of the research being conducted, the mechanism of implementing the actor model through meta-programming, as well as tools for increasing confidence in the management system through management decentralization and data localization, were determined.

Keywords: industrial robots, program, mechatronic, motion control system, trust, memory-centric, domain-oriented, declarative, actor, meta-programming

Funding: the work was supported by the Russian Ministry of Education and Science as part of the state assignment FSFS-2021-0004

For citation: Zelenskii A.A., Ivanovskii S.P., Ilyukhin Yu.V., Gribkov A.A. Programming a trusted memory-centric motion control system for robotic and mechatronic systems. *Aerospace MAI Journal*, 2022, vol. 29, no. 2, pp. 197-210. DOI: 10.34759/vst-2022-2-197-210

Введение

Одной из первоочередных задач научно-технологического развития в условиях цифровой трансформации экономики является совершенствование систем управления, в частности систем управления движением промышленных и коллаборативных роботов, станков с ЧПУ и других мехатронных систем. Необходимость обеспечения технологической безопасности работы предприятий обрабатывающей промышленности (главного потребителя роботов, станков с ЧПУ и мехатронных систем), в том числе предприятий оборонно-промышленного комплекса, авиастроения, судостроения и других стратегических отраслей, накладывает на научно-технологическое развитие дополнительные требования, связанные с обеспечением доверия к используемым системам управления с точки зрения функциональной надежности и информационной безопасности.

Одним из наиболее актуальных для цифрового производства трендов развития систем управления движением является существенное расширение их функциональных возможностей по управлению сложными многокоординатными нелинейными объектами в реальном времени. Указанный тренд оказывает существенное влияние как на развитие технологического оборудования (в том числе обрабатывающего), так и широкого спектра технологий управления движением в авиастроении [1—3], автомобилестроении, судостроении и производстве других транспортных средств. Но традиционные подходы оказываются недостаточно эффективными, поскольку существующие вычислительные машины уже не обеспечивают требуемого высокого быстродействия и не позволяют сформировать систему управления движением в реальном времени с функциональными возможностями, в полной мере соответствующими требованиям цифрового производства.

Закон Мура, скорректированная версия которого относится к 1975 году, последние десятилетия ориентировал разработчиков вычислительных машин на решение проблемы производительности за счет повышения быстродействия элементной базы. В настоящее время предел быстродействия электронных компонентов вычислительных машин практически достигнут [4] — в 2020 году появились первые 3-нм микросхемы, дальнейшее уменьшение размеров микросхем на основе используемых физических принципов не представляется возможным.

Практическое удовлетворение, с одной стороны, требований по совершенствованию систем управления движением промышленных роботов, станков с ЧПУ и других мехатронных систем и, с другой стороны, требований по обеспечению доверия может быть достигнуто только на основе совершенствования архитектуры систем управления движением. Основными направлениями её совершенствования являются уменьшение объема обрабатываемого потока данных за счет использования параллельных вычислений, увеличение скорости передачи данных между элементами вычислительной системы за счет применения технологий обработки в памяти (PIM) [5] или обработки вблизи памяти (NMC) [6], устранение очередей при одновременном обращении к одной памяти нескольких вычислительных устройств за счет физического разделения памяти между устройствами.

Перспективным технологическим трендом, в рамках которого реализуются все три перечисленные выше направления совершенствования архитектуры, являются системы управления с память-центрической архитектурой [7]. В ней данные в процессе вычислений не перемещаются между процессором и памятью, а остаются в памяти, в которую интегрируется процессор или аппаратный вычислительный ускоритель.

Система управления, реализующая память-центрическую архитектуру, имеет модульную структуру. Модули в значительной степени автономны, и объем данных, которыми модули обмениваются между собой, несопоставимо меньше объема данных, обрабатываемых внутри мо-

дулей. В результате память-центрическая архитектура позволяет снять существующие ограничения производительности вычислительных систем, называемые: *стена мощности* (the power wall), *стена памяти* (the memory wall) и *стена частоты* (the frequency wall) [8].

Ещё одна важная особенность память-центрической архитектуры — децентрализация хранения и обработки данных, позволяющая локализовать данные, являющиеся потенциальным объектом для угроз информационной безопасности, в пределах отдельных модулей или нескольких функционально связанных модулей, передача данных из которых вовне технически исключается. В результате возможности противодействия угрозам информационной безопасности существенно увеличиваются.

Функциональная надежность системы управления движением с память-центрической архитектурой также существенно выше, чем при использовании других архитектур (например, архитектуры фон Неймана). Это связано с тем, что при децентрализации обработки данных за счет использования нескольких квазиавтономных вычислительных устройств (модулей) отказ одного из них не приводит к выходу из строя всей системы, а может быть локализован в пределах данного устройства с выдачей ошибки обработки данных.

Разработанная в МГТУ «СТАНКИН» концептуальная память-центрическая модель систем управления роботов и станков с ЧПУ [9] и реализованные на её основе опытные образцы [10] на ПЛИС Cyclone IV включают в себя шесть основных блоков: ядро системы управления движением, имеющее базовую техническую реализацию в виде софт процессора NIOS II [11, 12] или RISC-V [13]; блок очувствления, включающий систему технического зрения и модуль обработки аналоговых и дискретных входов; исполнительный блок, включающий в себя модули кинематических, динамических и других вычислений, необходимых для управления движением промышленных роботов и станков с ЧПУ (модули трансформации, интерполяции, разгона/торможения, эквидистантной коррекции, модуль предпросмотра и др.), а также регуляторы исполнительных устройств; интеллектуальный блок, включающий в себя человеко-машинный интерфейс и искусственную нейронную сеть, на базе которой в тестовом режиме реализованы высокопроизводительные интеллектуальные алгоритмы машинного зрения [14]; общая оперативная память системы управления и локальная память акторных модулей и устройств системы; коммуникационная сеть, представляющая собой систему цифровых и аналоговых каналов связи, коммутационного оборудования и устройств для преобразования сигналов, включая модули высокопроизводительного собственного промышленного протокола Ethernet [15], субмикросекундного канала связи [16], известных стандартизованных промышленных Ethernet протоколов, таких как EtherCAT [17], PowerLink [18].

Практическая реализация системы управления с память-центрической архитектурой показала, что существующие подходы к программированию систем управления оказываются несостоятельными, а имеющееся программное обеспечение не может использоваться без существенной переработки. Поэтому с учётом особенностей алгоритмов работы систем с память-центрической архитектурой требуется разработать требования, которым должно удовлетворять программное обеспечение таких систем, и новые подходы к их программированию.

1. Требования к программированию систем управления движением с память-центрической архитектурой

Для полного использования возможностей системы управления движением с память-центрической архитектурой ее программное обеспечение должно удовлетворять нескольким базовым требованиям:

— язык программирования должен быть предметно-ориентированным, специализированным для управления движением;

— поскольку модули системы управления обладают существенной автономией и разнородными свойствами, для задания алгоритмов их работы (распределения задач между модулями) необходимо использовать декларативное программирование, включающее в себя элементы функционального и логического программирования;

— при программировании должна реализовываться акторная инструментальная модель, наилучшим образом соответствующая архитектуре и характеру взаимодействия элементов системы управления движением промышленного робота, станка с ЧПУ или мехатронной системы;

— для обеспечения быстродействия и компактности программного кода, а также оптимального использования ресурсов общей памяти системы управления при работе в режиме реального вре-

мени необходимо, чтобы используемый язык программирования был языком ассемблера (высокого уровня) или интерпретируемым языком;

— значимыми инструментами повышения функциональной надежности и информационной безопасности систем управления движением являются децентрализация управления и локализация обмена данными в отдельных модулях системы управления; данные свойства системы управления должны получить программную реализацию.

Как известно, языки программирования систем управления движением промышленных роботов, станков с ЧПУ и других мехатронных систем делятся в соответствии с областью применения на две основные группы.

• Группа языков высокого уровня, ориентированных на программирование и управление промышленными роботами [19—21]: KRL — KUKA Robot Language (KUKA, Германия); Karel (FANUC, Япония); RAPID (ABB, Швейцария); HrBasic (Hirata, Япония); RCML — Robot Control Meta Language (используют KUKA, FANUC, ABB, YASKAWA, разработчик — ООО «РСМЛ», Россия) и др.

• Группа параметрических языков программирования высокого уровня, ориентированных на программирования устройств с числовым программным управлением: G-code с расширениями; Custom Macro (FANUC, Япония); R Parameter (Siemens, Германия); User Task (Okuma, Япония); Q Routine (Sodick, Япония); APL — Advanced Programming Language [22]; FMS-3000 (МодмашСофт, Россия) и др.

В системах управления движением также используются языки для программируемых логических контроллеров, включенные в международный стандарт IEC 61131-3 [23] (LD —Ladder Diagram, FBD — Function Block Diagram, SFC — Sequential Function Chart, ST — Structured Text), а также языки контроллеров движения (PMAC язык от DeltaTau, Automation Basic от B&R и др.). На этих языках могут быть написаны библиотеки, подключаемые к программному обеспечению систем управления движением.

Язык программирования память-центрической системы управления движением должен относиться к группе языков высокого уровня, ориентированных на программирование и управление роботами и мехатронными системами. При этом он должен быть однозначно интерпретируемым в параметрический язык (язык макропрограммирования), например, в G-code.

Большинство языков программирования, используемых в настоящее время в системах управления движением, реализуют парадигму императивного программирования. Из числа уже названных языков программирования исключениями (декларативными функциональными языками) являются только APL и FBD (специальная модифицированная версия).

Программирование память-центрической системы управления движением, как мы уже определили, должно быть декларативным, т.е. ориентированным на описание в программе задачи и ожидаемого результата, а не последовательности команд, направленных на достижение заданной цели, как это имеет место при императивном программировании.

Как известно, декларативное программирование реализуется на основе двух основных методологий — функционального и логического программирования. Программирование память-центрической системы управления движением должно обеспечивать как функциональное, так и логическое программирование. Это обусловлено широким разнообразием решаемых в процессе управления движением специфических задач. Во-первых, это математические задачи: прямая и обратная задачи кинематики, первая и вторая задачи динамики, интерполяционная задача. Во-вторых, это логические задачи, решение которых позволяет принять управленческие решения на основании результатов работы искусственной нейронной сети или данных, полученных с использованием человеко-машинного интерфейса.

Ключевыми критериями выбора инструментальной модели для описания работы системы управления движением в режиме реального времени являются децентрализация управления, использование параллельных вычислений, асинхронный обмен данными между элементами (модулями) системы управления. Этим критериям в полной мере соответствуют акторная [24, 25] и реакторная модели [26, 27].

В настоящее время существует группа языков, ориентированных на модель акторов: ABCL (Actor-Based Concurrent Language), ActorScript, AmbientTalk и др. Понятие актора используется в некоторых функциональных языках [28]: Erlang, Scala, Elixir и др. За счет использования специальных библиотек расширения акторная модель может быть реализована средствами многих объектно-ориентированных императивных языков общего назначения: C++, C#, Java, Python, Ruby и др.

Другой подход к реализации акторной модели основан на метапрограммировании, при котором в качестве актора выступает не заданный в языке примитив, как в акторно-ориентированных языках, или класс, используемый при объектно-ориентированном программировании, а экземпляр программы, эмулируемый основной программой при старте или в процессе работы. Основная программа при этом приобретает свойства виртуальной машины. Метапрограммирование успешно реализуется во многих языках: C++, C#, Java, Python, Ruby и др.

Работа с акторами в память-центрической системой управления движением реализуется на основе использования метапрограммирования. При этом для каждого модуля системы управления основная программа будет эмулировать отдельную программу в оперативной памяти этого модуля. Это позволит достичь максимальной автономии акторов, реализуемых отдельными программами в памяти отдельных модулей — исходных объектов для моделирования в виде акторов.

Необходимым условием работы системы управления движением в режиме реального времени является исключение из процесса выполнения программы этапа компиляции программного кода. Это может быть достигнуто при использовании интерпретируемого языка или языка ассемблера. Многие языки программирования общего назначения имеют (иногда наряду с компилируемой) интерпретируемую версию, например, APL, BASIC, C, JavaScript, Forth, Lisp, Pascal, Perl, Python, Ruby, VBScript, VBA и др. Языки Forth [29] и Lisp [30] также относят к высокоуровневым языкам ассемблера.

Как известно, отличительной особенностью языков ассемблера является прямое соответствие команд программы и команд машинного кода. В результате синтаксис языков ассемблера привязан к архитектуре вычислительной машины. Это, с одной стороны, делает язык платформенно-зависимым (что в случае узко специализированного языка не имеет значения), а с другой стороны, позволяет за счет рационального использования вычислительных ресурсов и оптимизации вычислений снизить нагрузку на процессор и оперативную память, повысить скорость работы.

Подобными свойствами обладают также некоторые интерпретируемые языки, используемые для программирования систем ЧПУ, оптимизированные под интерпретацию в G-code. При этом такие интерпретируемые языки привязаны к архитектуре вычислительной машины в существенно меньшей степени, чем языки ассемблера.

Язык программирования память-центрической системы управления движением должен быть привязан к архитектуре системы управления движением, поэтому его синтаксис строится под команды ассемблера (соответствующие командам машинного кода), либо предполагает последующую интерпретацию (сразу или через промежуточную интерпретацию в G-code) в команды машинного или байт-кода. В первом случае язык программирования память-центрической системы управления будет языком ассемблера (и, соответственно, в систему необходимо будет включить ассемблер), во втором случае — интерпретируемым языком (и в систему необходимо будет включить интерпретатор или интерпретаторы). Все необходимые функциональные возможности языка в обоих случаях обеспечиваются, а создаваемый программный код будет оптимизированным по производительности (в том числе по быстродействию, что важно при работе в режиме реального времени) и использованию вычислительных ресурсов.

Последнее из сформулированных нами требований — обеспечение программной реализации децентрализации управления и локализации обмена данными [31, 32] — наиболее успешно реализуется в двух случаях: при использовании модели акторов (например, при программировании на язык Erlang [33], разработанном для отказоустойчивых распределённых и параллельных систем реального времени), и при объектно-ориентированном программировании (например, на языке Ruby [34] — самом «объектно-ориентированном» из всех «объектно-ориентированных» языков программирования, в котором все данные являются объектами. В первом случае локализация данных происходит в акторах, во втором случае — в объектах. Децентрализация управления для акторной модели является естественной, а при использовании объектно-ориентированного программирования децентрализация управления может быть реализована имеющимися программными средствами.

Программирование память-центрической системы управления движением относится к первому случаю — как мы уже установили, оно реализует акторную модель за счет использования метапрограммирования.

Следует отметить важную особенность акторной модели на основе метапрограммирования. При её использовании может быть обеспечено

существенное повышение доверия к системе управления как с точки зрения функциональной надежности, так и с точки зрения информационной безопасности. Система управления, построенная из автономных модулей и управляемая отдельными программами, запускаемыми в памяти этих модулей, обладает очень высокой отказоустойчивостью. Обеспечение информационной безопасности достигается за счет адресации сообщений от актора к актору, т.е. от программы к программе, при которой сообщения доступны только тем акторам, которым они адресованы. Кроме того, в каждом акторе (программе) имеется только та информация, которая необходима для его работы. Данные работы системы управления нигде не накапливаются и недоступны из центрального процессора. Часть данных по работе системы управления доступна для чтения оператором через человеко-машинный интерфейс, однако их изменение не предусмотрено.

Полная защита от угроз информационной безопасности на практике недостижима. Основными каналами реализации угроз информационной безопасности для память-центрической системы управления являются ее входы: модули очувствления (сенсоры, система технического зрения и т.д.), а также загружаемые через человеко-машинный интерфейс программный код (основной и эмулируемых программ) и файл конфигурации основной программы. Указанные каналы реализации угроз должны быть защищены дополнительными программными средствами.

2. Программирование на языке модульной цифровой системы

Все установленные требования к программированию память-центрической системы управления движением могут быть удовлетворены при использовании языка модульной цифровой системы, разработанного в МГТУ «СТАНКИН» и использованного при создании СЧПУ «Перспектива» [35, 36].

СЧПУ «Перспектива» имеет модульный принцип построения. Специально для нее была разработана система унифицированных электронных модулей (Модульная Цифровая Система – «МЦС»), позволяющая строить и в дальнейшем наращивать системы программного управления различным технологическим оборудованием, включая роботы и станки. Системой ЧПУ «Перспектива» были оснащены обрабатывающие центры СА535 (производитель — Станкозавод «Саста», Рязанская обл.), S500 (производитель — НПО «Станкостроение», г. Стерлитамак), КВС В4 (производитель — АО «КЭМЗ», г. Ковров), а также технологические роботы (производитель — ВМЗ АО «АВТОВАЗ»).

Ключевой составляющей анализа языка (в том числе языка программирования) является определение его системно-структурной организации, включая элементы (элементарные операторы) языка, механизмы их связывания и др. [37]. Эффективным инструментом определения системно-структурной организации языка программирования служит построение моделей. Наиболее информативными являются классификационная модель [38], определяющая парадигму, методологию программирования и другие классифицируемые свойства языка; модель синтаксической структуры языка [39,40], определяющая правила и составляющие языковых конструкций, и модель применения языка, позволяющая проследить механизмы применения языка программирования.

Классификационная модель языка (рис. 1) включает в себя следующие критерии классификации (классификационные основания):

— по специализации: предметно-ориентированный — язык программирования систем управления движением роботов, станков с ЧПУ и других мехатронных систем;

— по парадигме программирования: язык декларативного программирования;

— по методологии программирования: язык функционального и логического программирования;

— по реализации акторной модели: за счет метапрограммирования;

— по способу преобразования исходного программного кода в машинный код: интерпретируемый язык программирования;

— по инструментам повышения доверия: за счет децентрализованной архитектуры программ и за счет локализации переменных.

Модель синтаксической структуры языка (рис. 2) является достаточно типичной для языка программирования. Язык программирования строится из констант, переменных (различных типов и структур), операторов и других синтаксических элементов, требуемых для задания всех необходимых алгоритмов работы.

Модель применения языка (рис. 3) показывает наличие выраженного соответствия между

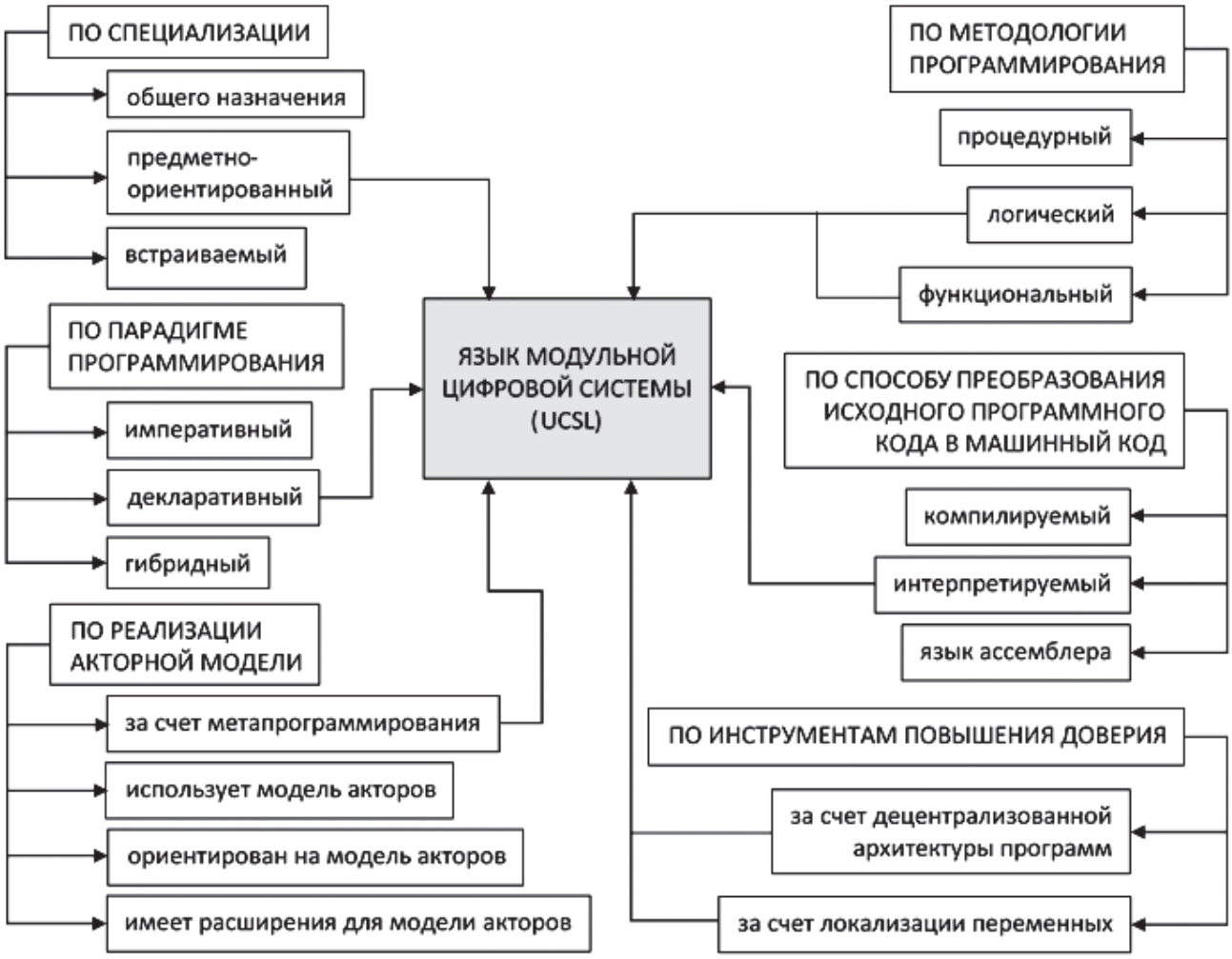

Рис. 1. Классификационная модель языка

элементами языка, с одной стороны, структурными элементами программ на этом языке, с другой стороны, и модулями вычислительной машины (системы управления движением) — с третьей стороны.

Ключевыми элементами языка являются комплекс синтаксических элементов, а также интерфейсы прикладного программирования, построенные из синтаксических элементов языка и служащие для интеграции в язык внешних библиотек (на том же или других языках).

Программа на языке включает в себя следующие основные элементы: операторы, структуры и выражения, формируемые из синтаксических элементов языка; акторы, формируемые в виде экземпляров дополнительных программ, эмулируемых (основной) программой при старте или в процессе работы.

Программный код на языке (рис. 4,*а*) описывает последовательность задач для управляемого робота, станка с ЧПУ или мехатронной системы. В ходе работы программы осуществляется диспетчеризация — задачи, определенные в программном коде, распределяются между акторами: часть задач выполняется акторами, создаваемыми центральным процессором в виде экземпляров программ в памяти центрального процессора или общей памяти, часть задач транслируется в виде фрагментов программного кода на языке через сети связи другим функциональным модулям (акторам в виде программ, управляющим работой других функциональных модулей и находящихся в памяти этих модулей).

Для реализации возможности использования программы для управления системами, образованными различными функциональными модулями, в рамках различных алгоритмов работы, параметры программы должны быть изменяемыми. Для этого требуется файл конфигурации (рис. 4,*б*), текст которого используются для задания настраиваемых параметров программы.

Рис. 2. Модель синтаксической структуры языка

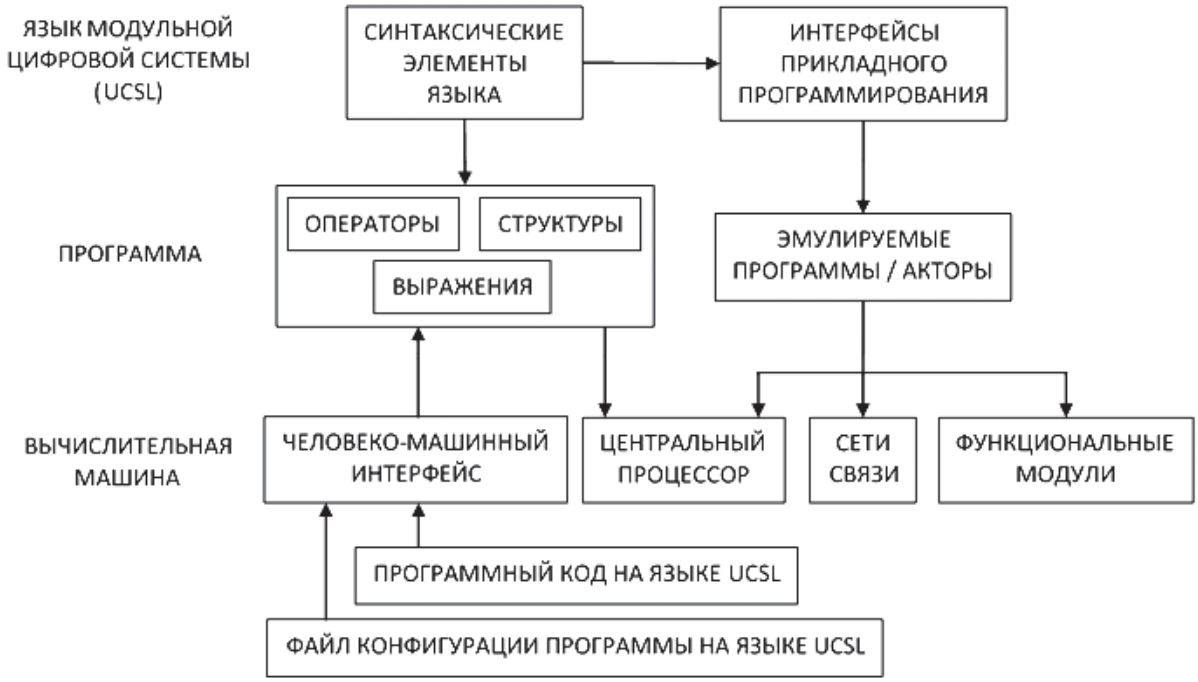

Рис. 3. Модель применения языка

Система управления движением, программируемая на языке (рис. 3), может быть представлена в виде четырех основных структурных составляющих (групп элементов):

– человеко-машинного интерфейса, через который передаются формируемые человеком-оператором программный код, описывающий алгоритм работы (последовательность задач) оборудования, а также файл конфигурации, обеспечивающий настройку программы под формулируемые задачи;

```
//x2@6  // Магазин инструментов в 0-позиции
//x2@7  // Магазин инструментов в A-позиции
//x2@8  // Магазин инструментов в B-позиции
//x2@9  // Магазин инструментов в C-позиции
//y0@8  // Скорость вращения магазина Бит 1
//y0@9  // Скорость вращения магазина Бит 2
//y2@6  // Вращение магазина вперед
//y2@7  // Вращение магазина назад
//YC3@10 // Заявка на поиск 1 положения магазина
//YC3@11 // Заявка на вращение магазина вперед
//YC3@12 // Заявка на вращение магазина назад
//YC3@13 // Заявка на включение освещения РЗ
…
// Цикл поиска 1 позиции в магазине
        IF ($YC3@10)
                IF ($TecPos==0)
                        IF ($YC3@12)
                                y2@6=0
                                y2@7=1
                        ELSE
                                y2@7=0
                                y2@6=1
                        END IF
                        y0@9=0
                        y0@8=1
                ELSE IF (x2@7 && x2@8 && x2@9)
                        y2@6=y2@7=y0@8=y0@9=0
                        $YC3@10=$YC3@11=$YC3@12=0
                END IF
        END IF
…
```

а)

```
// Определения
EXTERN NETWORK // Параметры Ethernet
EXTERN PIO     // Параметры модуля ввода-вывода
EXTERN TOOL    // Параметры инструмента
EXTERN CONTROL // Регистры управления
EXTERN CHN_CFG // Регистры управления каналом
EXTERN SYS_CFG // Параметры системы
...
// Параметры модулей ввода-вывода
// {идентификатор, тип (0-5), входов, выходов}
// Дискретный модуль - 16 входов / 16 выходов:
PIO xbus_d1616 = { 0x44213201, 2, 1, 1 }
// Дискретный модуль - 32 входа:
PIO xbus_d3200 = { 0x44213204, 4, 1, 0 }
// Дискретный дифф. модуль - 8 входов / 8 выходов:
PIO xbus_d0808 = { 0x44213203, 4, 1, 1 }
// Аналоговый модуль - 16 бит, 16 входов / 16 выходов:
PIO xbus_a0606 = { 0x44213202, 3, 6, 6 }
// Модуль безопасности - 32 входа / 32 выхода:
PIO xbus_safety = { 1, 4, 1, 1 }
// Таблица инструментов
// {вылет Z, смещение X, ориентация, диаметр}
SC.tool_table = { , ...
        {-128.0981, 0, 0, 10 }, ...   // Фреза
        {-89.3152, 0, 0, 4 }, ...     // Фреза
        {-108.9593, 0, 0, 6.1 }, ...  // сверло
        {-89.3152, 0, 0, 4.1 }, ...   // Фреза
        {-89.3152, 0, 0, 4.6 }, ...   // Фреза
}
INT tool_table[1, 8]
...
```

б)

Рис. 4. Фрагменты программного кода (*а*) и файла конфигурации (*б*) на языке

– центрального процессора (ядра), ответственного за общее управление системой и распределение задач;

– функциональных модулей, выполняющих обработку (под управлением собственных программ, находящихся в памяти этих модулей) данных очувствления, вычисления (траекторий рабочих органов, их кинематики, динамики, режимов работы исполнительных устройств и т.д.) и управление регуляторами исполнительных устройств;

– сетей связи, обеспечивающих (при необходимости — под управлением собственных программ, находящихся в общей памяти системы) связь между структурными элементами вычислительной машины, а также с внешними устройствами (датчиками и исполнительными механизмами).

Выводы

На основании проведенного исследования можно сделать следующие выводы.

1. Развитие производства в условиях цифровой трансформации экономики требует первоочередного совершенствования систем управления роботами, станками и другими мехатронными системами. Обеспечение технологической безопасности и обороноспособности страны накладывает на это развитие дополнительные требования по повышению доверия к системам управления. Совершенствование систем управления роботов, станков с ЧПУ и других мехатронных систем, расширение их функциональных возможностей и обеспечение доверия к таким системам может быть достигнуто за счет создания систем управления с память-центрической архитектурой.

2. Для полного использования возможностей память-центрической системы управления программирование таких систем должно выполняться на предметно-ориентированном интерпретируемом языке, в соответствии с парадигмой декларативного программирования (с элементами функционального и логического программирования), реализовывать модель акторов, а также обеспечивать повышение доверия к системе управления движением за счёт децентрализованной архитектуры программ и локализации переменных.

3. Указанные требования к программированию память-центрической системы управления могут быть удовлетворены при использовании разработанного авторами предметно-ориентированного декларативного интерпретируемого языка модульной системы, реализации акторной модели посредством метапрограммирования, а также использовании инструментов децентрали-

зации управления и локализации данных, наилучшим образом совместимых с акторной моделью с эмулируемыми программами в качестве акторов.

Список источников

1. *Погосян М.А., Верейкин А.А.* Управление положением и движением летательных аппаратов в системах автоматической посадки: Аналитический обзор // Вестник Московского авиационного института. 2020. Т. 27. № 3. С. 7-22. DOI: 10.34759/vst-2020-3-7-22
2. *Кульков В.М., Юн С.У., Фирсюк С.О.* Метод управления движением малых космических аппаратов с использованием надувных тормозных устройств для торможения при орбитальном полете до входа в атмосферу // Вестник Московского авиационного института. 2020. Т. 27. № 3. С. 23-36. DOI: 10.34759/vst-2020-3-23-36
3. *Фёдоров А.В., Хоанг В.Т.* Программный комплекс для проектирования алгоритмов управления движением сервисного модуля на геостационарной орбите // Вестник Московского авиационного института. 2020. Т. 27. № 4. С. 192-205. DOI: 10.34759/vst-2020-4-192-205
4. *Leiserson C.E., Thompson N.C., Emer J.S.* et al. There's plenty of room at the Top: What will drive computer performance after Moore's law? // Science. 2020. Vol. 368. No. 6495. DOI: 10.1126/science.aam9744
5. *Ghose S., Hsieh K., Boroumand A., Ausavarungnirun R., Mutlu O.* Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions. 2018, 45 p. URL: CoRR abs/1802.00320
6. *Singh G., Chelini L., Corda S.* et al. Near-Memory Computing: Past, Present, and Future // Microprocessors and Microsystems. 2019. DOI: 10.48550/arXiv.1908.02640
7. *Каляев И.А., Заборовский В.* Искусственный интеллект: от метафоры к техническим решениям // Control Engineering Россия. 2019. № 5(83). С. 26-31. URL: https://controleng.ru/wp-content/uploads/8326.pdf
8. Cell Broadband Engine Programming Tutorial. Version 2.0. IBM Systems and Technology Group. 2006б 185 p. URL: https://arcb.csc.ncsu.edu/~mueller/cluster/ps3/CBE_Tutorial_v2.0_15December2006.pdf
9. *Зеленский А.А., Илюхин Ю.В., Грибков А.А.* Память-центрические модели систем управления движением промышленных роботов // Вестник Московского авиационного института. 2021. Т. 28. № 4. С. 245-256. DOI: 10.34759/vst-2021-4-245-256
10. *Зеленский А.А., Абдуллин Т.Х., Илюхин Ю.В., Харьков М.А.* Высокопроизводительная цифровая система на основе ПЛИС для управления движением многокоординатных станков и промышленных роботов // СТИН. 2019. № 8. С. 5-8.
11. Nios® II Processor Reference Guide. Intel Corporation, 2020, 230 p. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf
12. Nios® II Software Developer Handbook. Intel Corporation, 2021. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2sw_nii5v2gen2.pdf
13. *Waterman A., Asanović K.* The RISC-V Instruction Set Manual. Volume I: User-Level ISA. Document Version 2.2. 2017, 145 p. URL: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf
14. *Зеленский А.А., Франц В.А., Семенищев Е.А.* Алгоритм планирования траектории рабочего органа манипулятора для привязки базисных систем координат с использованием технического зрения // Вестник машиностроения. 2019. № 10. С. 3-7.
15. *Зеленский А.А., Шадрин Н.Г., Абдуллин Т.Х., Харьков М.А.* Высокоскоростная промышленная сеть реального времени киберфизических систем // Вестник компьютерных и информационных технологий. 2019. №11 (185). С. 46-52.
16. *Харьков М.А., Ивановский С.П., Зеленский А.А., Абдуллин Т.Х.* Распределенная система управления электроавтоматикой станков, промышленных роботов и автоматизированных комплексов на основе высокопроизводительного интерфейса связи // Вестник МГТУ Станкин. 2018. № 1(44). С. 91-95.
17. EtherCAT Communication Manual. Cat. No. Q179-E1-01. OMRON Corporation, Industrial Automation Company. 2010. URL: https://www.tecnical.cat/PDF/OMRON/Vision/Q179-E1-01.pdf
18. Ethernet POWERLINK Communication Profile Specification. EPSG, 2016. URL: https://www.ethernet-power-link.org/fileadmin/user_upload/dokumente/downloads/technical_documents/epsg_ds_301_v-1-3-0__4_.pdf
19. *Mühe H., Angerer A., Hoffmann A., Reif W.* On reverse-engineering the KUKA Robot Language // 1st International Workshop on Domain-Specific Languages and models for ROBotic systems. 2010. URL: https://arxiv.org/pdf/1009.5004.pdf
20. HrBasic Reference Manual Ver. 5.50. Hirata 2004-2005. URL: https://www.hirata.de/fileadmin/content/05_Kontakt/Support_Download/Programmierhandbuch_fuer_HR-Basic_Syntax_Version_2_10-2008__H-XXXX-1E.pdf
21. *Сутормин Д.К., Тюлькин М.В.* Robot Control Meta Language. Метаязык для роботов. — Пермь, Издательский центр «Титул», 2015. — 72 с.
22. *Finkel R.* Advanced programming languages design. — University of Kentucky. Addison-Wesley Publishing Company, Inc. 1996. — 363 p. URL: https://www.researchgate.net/publication/220692467_Advanced_programming_language_design
23. IEC 61131-3 International standard. Second edition 2003-01. Programmable controllers. Part 3: Programming languages. URL: https://d1.amobbs.com/bbs_upload782111/files_31/ourdev_569653.pdf

24. *Burgin M.* Systems, Actors and Agents: Operation in a multicomponent environment. 2017. DOI: 10.48550/arXiv.1711.08319
25. *Rinaldi L., Torquati M., Mencagli G., Danelutto M., Menga T.* Accelerating Actor-based Applications with Parallel Patterns // 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (13-15 February 2019; Pavia, Ital). DOI: 10.1109/EMPDP.2019.8671602
26. *Shah V., Salles M.* Reactors: A Case for Predictable, Virtualized Actor Database Systems // International Conference on Management of Data SIGMOD'18 (10—15 June 2018; Houston TX USA), pp. 259—274. DOI: 10.1145/3183713.3183752
27. *Lohstroh M., Menard C., Bateni S., Lee E.* Toward a Lingua Franca for Deterministic Concurrent Systems // ACM Transactions on Embedded Computing Systems. 2021. Vol. 20. No. 4. Article 36. DOI: 10.1145/3448128
28. *Batko P., Kuta M.* Actor model of Anemone functional language // The Journal of Supercomputing. 2018. Vol. 74, pp. 1485-1496. DOI: 10.1007/s11227-017-2233-1
29. *Brodie L.* Thinking Forth: A Language and Philosophy for Solving Problems. — Punchy Publishing, 2004. — 316 p. URL: https://jztkft.dl.sourceforge.net/project/thinking-forth/reprint/rel-1.0/thinking-forth-color.pdf
30. *Steele L.* Common LISP. The Language. — 2nd Edition. — Thinking Machines, Inc., 1990. — 1029 p. URL: https://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html
31. *Porter J., Menascé D., Gomaa H.* Decentralized Software Architecture Discovery in Distributed Systems. Technical Reports. George Mason University, Department of Computer Science, 2016. URL: https://cs.gmu.edu/media/techreports/GMU-CS-TR-2016-2.pdf
32. *Porter J., Menascé D., Gomaa H.* DeSARM: A Decentralized Mechanism for Discovering Software Architecture Models at Runtime in Distributed Systems. MoDELS@Run.time, 2016. URL: http://ceur-ws.org/Vol-1742/MRT16_paper_3.pdf
33. Getting Started with Erlang User's Guide. Version 12.0.2. Ericsson AB, 2021. URL: https://erlang.org/doc/getting_started/users_guide.html
34. ISO/IEC 30170:2012. Information technology — Programming languages — Ruby. 2012, 313 p. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:30170:ed-1:v1:en
35. *Зеленский А.А., Порядин Д.В., Морозкин М.С.* Интерпретатор ядра СЧПУ. Свидетельство о регистрации программы для ЭВМ RU2020612803. Бюл. № 3, 03.03.2020.
36. *Зеленский А.А., Порядин Д.В., Морозкин М.С., Купцов В.Р.* Графический интерфейс системы ЧПУ «Перспектива». Свидетельство о регистрации программы для ЭВМ RU2020612698. Бюл. № 3, 28.02.2020.
37. *Raphael B.* The structure of programming languages // Communications of the ACM. 1966. Vol. 9. No. 2, pp 67—71. DOI: 10.1145/365170.365175
38. *Volkova V., Kozlov V., Mager V., Chernenkaya L.* Classification of methods and models in system analysis // XX IEEE International Conference on Soft Computing and Measurements – SCM (24—26 May 2017; St. Petersburg, Russia). DOI: 10.1109/SCM.2017.7970533
39. *Hilfinger P.* A Model of Programming Languages. University of California, Department of Electrical Engineering and Computer Sciences, Computer Science Division. 1998. URL: https://people.eecs.berkeley.edu/~jrs/61bf98/reader/ucb/java-model.pdf
40. *Martini S.* The Standard Model for Programming Languages: The Birth of a Mathematical Theory of Computation // OpenAccess Series in Informatics (OASIcs). 2020. Vol. 86, 8:1-8:13. DOI: 10.4230/OASIcs.Gabbrielli.8

References

1. Pogosyan M.A., Vereikin A.A. Position and motion control of aerial vehicles in automatic landing systems: analytical review. *Aerospace MAI Journal,* 2020, vol. 27, no. 3, pp. 7-22. DOI: 10.34759/vst-2020-3-7-22
2. Kul'kov V.M., Yoon S.W., Firsyuk S.O. A small spacecraft motion control method employing inflatable braking units for deceleration while orbital flight prior to the atmospheric entry. *Aerospace MAI Journal,* 2020, vol. 27, no. 3, pp. 23-36. DOI: 10.34759/vst-2020-3-23-36
3. Fedorov A.V., Hoang V.T. Software package for motion control algorithms design of service module in geostationary orbit. *Aerospace MAI Journal,* 2020, vol. 27, no. 4, pp. 192-205. DOI: 10.34759/vst-2020-4-192-205
4. Leiserson C.E., Thompson N.C., Emer J.S. et al. There's plenty of room at the Top: What will drive computer performance after Moore's law? *Science*, 2020, vol. 368, no. 6495. DOI: 10.1126/science.aam9744
5. Ghose S., Hsieh K., Boroumand A., Ausavarungnirun R., Mutlu O. *Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions.* 2018, 45 p. URL: CoRR abs/1802.00320
6. Singh G., Chelini L., Corda S. et al. Near-Memory Computing: Past, Present, and Future. *Microprocessors and Microsystems*, 2019. DOI: 10.48550/arXiv.1908.02640
7. Kalyaev I.A., Zaborovskii V. *Control Engineering Rossiya*, 2019, no. 5(83), pp. 26-31. URL: https://controleng.ru/wp-content/uploads/8326.pdf

8. *Cell Broadband Engine Programming Tutorial. Version 2.0.* IBM Systems and Technology Group. 2006б 185 p. URL: https://arcb.csc.ncsu.edu/~mueller/cluster/ps3/CBE_Tutorial_v2.0_15December2006.pdf
9. Zelenskii A.A., Ilyukhin Yu.V., Gribkov A.A. Memory-centric models of industrial robots control systems. *Aerospace MAI Journal*, 2021, vol. 28, no. 4, pp. 245-256. DOI: 10.34759/vst-2021-4-245-256
10. Zelenskii A.A., Abdullin T.Kh., Ilyukhin Yu.V., Khar'kov M.A. *STIN*, 2019, no. 8, pp. 5-8.
11. *Nios® II Processor Reference Guide. Intel Corporation,* 2020, 230 p. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf
12. *Nios® II Software Developer Handbook. Intel Corporation,* 2021. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2sw_nii5v2gen2.pdf
13. Waterman A., Asanović K. *The RISC-V Instruction Set Manual. Volume I: User-Level ISA. Document Version 2.2.* 2017, 145 p. URL: https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf
14. Zelenskii A.A., Frants V.A., Semenishchev E.A. *Vestnik mashinostroeniya*, 2019, no. 10, pp. 3-7.
15. Zelenskii A.A., Shadrin N.G., Abdullin T.Kh., Khar'kov M.A. *Vestnik komp'yuternykh i informatsionnykh tekhnologii*, 2019, no. 11 (185), pp. 46-52.
16. Khar'kov M.A., Ivanovskii S.P., Zelenskii A.A., Abdullin T.Kh. *Vestnik MGTU Stankin*, 2018, no. 1(44), pp. 91-95.
17. *EtherCAT Communication Manual. Cat. No. Q179-E1-01.* OMRON Corporation, Industrial Automation Company, 2010. URL: https://www.tecnical.cat/PDF/OMRON/Vision/Q179-E1-01.pdf
18. *Ethernet POWERLINK Communication Profile Specification*. EPSG, 2016. URL: https://www.ethernet-power-link.org/fileadmin/user_upload/dokumente/downloads/technical_documents/epsg_ds_301_v-1-3-0__4_.pdf
19. Mühe H., Angerer A., Hoffmann A., Reif W. On reverse-engineering the KUKA Robot Language. *1st International Workshop on Domain-Specific Languages and models for ROBotic systems,* 2010. URL: https://arxiv.org/pdf/1009.5004.pdf
20. *HrBasic Reference Manual Ver. 5.50.* Hirata 2004-2005. URL: https://www.hirata.de/fileadmin/content/05_Kontakt/Support_Download/Programmierhandbuch_fuer_HR-Basic_Syntax_Version_2_10-2008__H-XXXX-1E.pdf
21. Sutormin D.K., Tyul'kin M.V. *Robot Control Meta Language. Metayazyk dlya robotov* (Robot Control Meta Language. A meta-language for robots). Perm, Titul, 2015, 72 p.
22. Finkel R. *Advanced programming languages design*. University of Kentucky. Addison-Wesley Publishing Company, Inc. 1996, 363 p. URL: https://www.researchgate.net/publication/220692467_Advanced_programming_language_design
23. *IEC 61131-3 International standard. Second edition 2003-01. Programmable controllers. Part 3: Programming languages.* URL: https://d1.amobbs.com/bbs_upload782111/files_31/ourdev_569653.pdf
24. Burgin M. *Systems, Actors and Agents: Operation in a multicomponent environment.* 2017. DOI: 10.48550/arXiv.1711.08319
25. Rinaldi L., Torquati M., Mencagli G., Danelutto M., Menga T. Accelerating Actor-based Applications with Parallel Patterns. *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (13-15 February 2019; Pavia, Italy)*. DOI: 10.1109/EMPDP.2019.8671602
26. Shah V., Salles M. Reactors: A Case for Predictable, Virtualized Actor Database Systems. *International Conference on Management of Data SIGMOD'18 (10—15 June 2018; Houston TX USA)*, pp. 259—274. DOI: 10.1145/3183713.3183752
27. Lohstroh M., Menard C., Bateni S., Lee E. Toward a Lingua Franca for Deterministic Concurrent Systems. *ACM Transactions on Embedded Computing Systems,* 2021, vol. 20, no. 4. Article 36. DOI: 10.1145/3448128
28. Batko P., Kuta M. Actor model of Anemone functional language. *The Journal of Supercomputing*, 2018, vol. 74, pp. 1485-1496. DOI: 10.1007/s11227-017-2233-1
29. Brodie L. *Thinking Forth: A Language and Philosophy for Solving Problems.* Punchy Publishing, 2004, 316 p. URL: https://jztkft.dl.sourceforge.net/project/thinking-forth/reprint/rel-1.0/thinking-forth-color.pdf
30. Steele L. *Common LISP. The Language.* 2nd Edition. Thinking Machines, Inc., 1990, 1029 p. URL: https://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html
31. Porter J., Menascé D., Gomaa H. *Decentralized Software Architecture Discovery in Distributed Systems.* Technical Reports. George Mason University, Department of Computer Science, 2016. URL: https://cs.gmu.edu/media/techreports/GMU-CS-TR-2016-2.pdf
32. Porter J., Menascé D., Gomaa H. *DeSARM: A Decentralized Mechanism for Discovering Software Architecture Models at Runtime in Distributed Systems.* MoDELS@Run.time, 2016. URL: http://ceur-ws.org/Vol-1742/MRT16_paper_3.pdf
33. *Getting Started with Erlang User's Guide. Version 12.0.2.* Ericsson AB, 2021. URL: https://erlang.org/doc/getting_started/users_guide.html
34. *ISO/IEC 30170:2012. Information technology — Programming languages — Ruby.* 2012, 313 p. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:30170:ed-1:v1:en
35. Zelenskii A.A., Poryadin D.V., Morozkin M.S. *Svidetel'stvo o gosudarstvennoi registratsii programm dlya EVM "Interpretator yadra SChPU" RU2020612803, 03.03.2020* (Certificate of state registration of computer programs "CNC core Interpreter" RU2020612803, 03.03.2020).

36. Zelenskii A.A., Poryadin D.V., Morozkin M.S., Kuptsov V.R. *Svidetel'stvo o gosudarstvennoi registratsii programm dlya EVM "Graficheskii interfeis sistemy chpu PERSPEKTIVA" RU2020612698, 28.02.2020* (Certificate of state registration of computer programs " Graphical interface of the Perspective CNC system" RU2020612698, 28.02.2020).
37. Raphael B. The structure of programming languages. *Communications of the ACM*, 1966, vol. 9, no. 2, pp 67–71. DOI: 10.1145/365170.365175
38. Volkova V., Kozlov V., Mager V., Chernenkaya L. Classification of methods and models in system analysis. *XX IEEE International Conference on Soft Computing and Measurements – SCM (24—26 May 2017; St. Petersburg, Russia)*. DOI: 10.1109/SCM.2017.7970533
39. Hilfinger P. *A Model of Programming Languages*. University of California, Department of Electrical Engineering and Computer Sciences, Computer Science Division, 1998. URL: https://people.eecs.berkeley.edu/~jrs/61bf98/reader/ucb/java-model.pdf
40. Martini S. The Standard Model for Programming Languages: The Birth of a Mathematical Theory of Computation. *OpenAccess Series in Informatics (OASIcs),* 2020, vol. 86, 8:1-8:13. DOI: 10.4230/OASIcs.Gabbrielli.8

Статья поступила в редакцию 01.03.2022; одобрена после рецензирования 04.03.2022; принята к публикации 10.03.2022.

The article was submitted on 01.03.2022; approved after reviewing on 04.03.2022; accepted for publication on 10.03.2022.