ВЕСТНИК РОССИЙСКОЙ АКАДЕМИИ НАУК, 2023, том 93, № 2, с. 150–161

ОБОЗРЕНИЕ

ПРОГРАММИРОВАНИЕ БЛИЖАЙШЕГО БУДУЩЕГО: КОНЦЕПЦИЯ И ПРАГМАТИКА

© 2023 г. В. П. Ильин[a,b,*]

[a] *Институт вычислительной математики и математической геофизики СО РАН, Новосибирск, Россия*

[b] *Новосибирский государственный технический университет, Новосибирск, Россия*

***** *E-mail: ilin@sscc.ru*

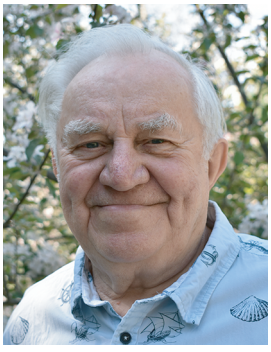
Поступила в редакцию 07.07.2022 г.
После доработки 19.09.2022 г.
Принята к публикации 31.10.2022 г.

В статье рассматриваются концепция, архитектура и научно-организационные проблемы создания интегрированного программного обеспечения нового поколения, предназначенного для предсказательного моделирования в машиностроении, энергетике, материаловедении, биологии, медицине, экономике, природопользовании, экологии, социологии и др. Математические постановки включают в себя междисциплинарные прямые и обратные экстремально ресурсоёмкие задачи, решение которых осуществляется с помощью вычислительных методов и технологий масштабируемого распараллеливания путём гибридного программирования на гетерогенных суперкомпьютерах с распределённой и иерархической общей памятью. Концепция проекта содержит разработку инструментального вычислительного окружения, поддерживающего все стадии крупномасштабного машинного эксперимента: геометрическое и функциональное моделирование, генерацию адаптивных неструктурированных сеток различных типов и порядков, аппроксимацию исходных уравнений, решение возникающих алгебраических проблем, постпроцессинг получаемых результатов, оптимизационные методы для обратных задач, машинное обучение и принятие решений по итогам расчётов. Эффективная функциональность инструментального вычислительного окружения базируется на высокопроизводительных вычислениях и интеллектуализированных средствах работы с большими данными. Архитектура инструментального вычислительного окружения предусматривает автоматизированное расширение состава реализуемых моделей и применяемых алгоритмов, адаптацию к эволюции суперкомпьютерных платформ, дружественные интерфейсы и активное переиспользование внешних программных продуктов, согласованное участие различных групп разработчиков, что в совокупности должно обеспечить длительный жизненный цикл и востребованность создаваемой экосистемы широким кругом пользователей из различных профессиональных областей.

Ключевые слова: предсказательное математическое моделирование, интегрированное вычислительное окружение, инструментальное программное обеспечение, высокопроизводительные методы и технологии, искусственный интеллект.

DOI: 10.31857/S086958732302007X, **EDN:** FCSFWU

Алгоритмы + структуры данных = программы
Н. Вирт, 1976



ИЛЬИН Валерий Павлович – доктор физико-математических наук, главный научный сотрудник лаборатории вычислительной физики ИВМиМГ СО РАН.

Основы программирования были заложены работами леди А. Лавлейс, дочери знаменитого английского поэта Д. Байрона, соратницы изобретателя первой аналитической вычислительной машины, прообраза современной ЭВМ, Ч. Бэббиджа [1]. В силу существовавших в первой половине XIX в. технологических ограничений Бэббиджу не удалось довести свою конструкцию до работоспособного состояния, но благодаря публикациям Лавлейс, содержавшим описание не только самой механической вычислительной ма-

шины, но и принципы программирования её работы, даже с возможностями распараллеливания некоторых операций, научный мир узнал об изобретении, значительно опередившем свой век.

История программирования тесно связана с совершенствованием компьютеров, обусловленным необходимостью решения больших задач, в первую очередь национальной безопасности. Во второй половине XX в. появились электронные вычислительные машины (ЭВМ). Их поколения – на лампах, на транзисторах, на интегральных схемах, на больших и сверхбольших интегральных схемах – сменялись быстро, и каждая такая смена сопровождалась увеличением скорости выполнения арифметических и логических операций, а также примерно пропорциональным увеличением объёмов памяти. Все последние десятилетия продолжался экспоненциальный рост компьютерных мощностей (по закону Мура – в 1000 раз за 11 лет!). Одновременно с материальной основой бурно развивался "софт", их стоимости стали сопоставимыми. Программное обеспечение разделилось на системное (трансляторы и компиляторы с разных языков программирования, операционные системы и всевозможный инструментарий, отвечающий за общую работоспособность компьютера) и прикладное (алгоритмические процедуры и модули, библиотеки и пакеты программ), или проблемно-ориентированное, направленное на решение насущных запросов пользователей. Введённые академиком А.П. Ершовым понятия "компьютерная грамотность", "школьная информатика" косвенно свидетельствовали о том, что программирование обретает черты массовой профессии. (Содержательный исторический анализ становления и развития перечисленных направлений, неразрывно связанных с прикладной и вычислительной математикой, представлен в работах [2, 3].) Во второй половине XX в. формировалось и математическое моделирование в методологических рамках пакетов прикладных программ [4–7]. Кроме того, большой вес приобрела производственно-индустриальная линия программного обеспечения – системы автоматизированного проектирования (САПР), составляющие быстро растущий рынок разнообразных продуктов (CAD, CAM, CAE, PLM), призванных значительно повысить производительность труда в своих областях [8]. Следует отметить, что в таких комплексах, изначально ориентированных на простые инженерные расчёты, всё активнее используется наукоёмкое программирование с математическим моделированием.

Развитие программирования не всегда шло гладко. Можно отметить такой большой проект, как язык и система программирования "Альфа" [9], которые активно разрабатывались и внедрялись в СО АН СССР в 1960–1970-е годы. Хотя для этого русскоязычного варианта Алгола был создан первый в мире оптимизирующий транслятор, проект не смог выжить в мировом окружении разработок на английском языке, де-факто ставшего играть роль эсперанто в программировании.

Ещё одна драматичная история в программировании, на этот раз уже конца XX в., связана с масштабным международным проектом HPF (High Performance Fortran) [10]). Его разработчики не смогли преодолеть внутренние концептуальные противоречия, и проект был закрыт. Поразительно, что проблема создания полноценной программной системы параллельного программирования, несмотря на её архиактуальность, до сих пор остаётся открытой, а существующий инструментарий представляет собой только паллиативные решения параллельной виртуальной машины (например, проект DVM [11]).

Начало XXI столетия ознаменовалось повсеместным распространением персональных компьютеров и Интернета, качественно изменивших образ жизни и формы деятельности человека. А появление многопроцессорных вычислительных систем с облачными вычислениями и быстрыми сетями передачи больших данных приводит к глобализации моделирования, открывая новый путь получения фундаментальных и прикладных знаний. Как отмечал Дж. Донгарра с коллегами в дорожной карте международного проекта IESP (International Exascale Software Project [12]), мировые тенденции суперкомпьютеризации ставят перед вычислительным сообществом проблему создания в кратчайшие сроки программного обеспечения огромных масштабов. Решение этих задач требует новых концепций и возможно только при широчайшей кооперации. Необходимо отметить и тот факт, что производительность труда программистов в целом катастрофически отстаёт от темпов совершенствования самих компьютеров, и в этом смысле можно говорить о безотлагательности преодоления наблюдающегося мирового кризиса программирования. Здесь главным средством должны стать интеллектуализация и индустриализация технологий наукоёмкого программирования, которые способны качественно поднять уровень автоматизации построения моделей, алгоритмов и их отображения в архитектуре ЭВМ.

Не менее важная проблема – повышение производительности математического и программного обеспечения многопроцессорных вычислительных систем путём масштабируемого распараллеливания (в сильном и слабом смыслах), достижимого за счёт активного внедрения гибридного программирования на гетерогенных кластерных платформах с распределённой и иерархической общей памятью, с возможностя-

ми передачи сообщений между вычислительными узлами, многопотоковых вычислений на многоядерных процессорах, векторизации операций и эффективного применения графических ускорителей (см. аналитические обзоры в статьях [13, 14]). Один из главных путей оптимизации кода заключается в минимизации информационных обменов между вычислительными устройствами, поскольку передача данных не только замедляет расчёты, но и значительно повышает эксплуатационные расходы, поскольку такие операции наиболее энергозатратны.

Важнейшая, ещё не до конца осознанная миссия современных вычислительных ресурсов – предсказательное моделирование процессов и явлений, которое призвано оптимизировать маршруты человеческой деятельности. Примером достижений в этой области может служить краткосрочный (до нескольких дней) прогноз погоды. Впрочем, современные возможности предсказательного моделирования ограниченны, о чём свидетельствуют ошибки учёных в прогнозе волн пандемии коронавируса. Возникает вопрос: каким должно стать программирование, точнее говоря, математическое и программное обеспечение, адекватное стоящим перед нами вызовам, в предстоящие 5, 10, 15 лет?

СИСТЕМНЫЙ ВЗГЛЯД НА ПРИКЛАДНОЕ ПРОГРАММИРОВАНИЕ

Одновременно с экстенсивным развитием мирового компьютерного парка растёт и армия программистов, которые производят огромное количество продуктов, представляющих в совокупности бесценный интеллектуальный потенциал. Какие-то из программ в течение длительного времени имеют своего массового пользователя, а многие отмирают, в том числе из-за смены компьютерных поколений. Эволюция этой большой системы достаточно динамична, поскольку научно-технический прогресс инициирует появление новых практических сверхзадач, а активное развитие теоретической, прикладной и вычислительной математики порождает новые модели, методические подходы, алгоритмы и технологии, в результате чего производство математического программного обеспечения (МПО) приобретает неконтролируемые масштабы. Чтобы в этом хаосе разработок как-то разобраться, необходимо провести систематизирующую классификацию, которая может быть осуществлена по различным системам координат:

• общие направления человеческой деятельности: промышленность и сельское хозяйство, здравоохранение и образование, экология, экономика и финансы, социальные и гуманитарные сферы;

• производственные отрасли: энергетика, машиностроение, материаловедение, природопользование, транспорт, строительство, химические и биологические технологии и т.д.;

• физико-математические направления: электромагнетизм, гидро-газодинамика, упруго-пластичность, тепломассоперенос и фильтрация с фазовыми переходами, кинетические процессы, междисциплинарные прямые и обратные задачи;

• применяемые модели и методики: системы дифференциальных и/или интегральных уравнений, классические и обобщённые постановки, статистическое моделирование и алгоритмы Монте-Карло, дискретные задачи;

• реализуемые алгоритмы и технологии: методы построения сеток, аппроксимации функций, а также дифференциальных и интегральных уравнений, графовые преобразования, вычислительная алгебра, методы оптимизации и решения обратных задач, численное интегрирование динамических систем, распараллеливание алгоритмов;

• целевые компьютерные платформы: персональные компьютеры, серверы, кластеры, графические ускорители, облачные вычисления, мобильные устройства, гетерогенные архитектуры, реконфигурируемые программируемые логические интегральные схемы.

Такая многомерная “манхэттенская” структура свидетельствует об огромном многообразии и суммарном объёме МПО, что обусловливает необходимость тщательной систематизации и планирования разработок для достижения максимальной эффективности их широкого и длительного применения разнопрофильными пользователями.

Для спецификации прикладных программ важны такие категории, как малые, средние и большие задачи. С одной стороны, они характеризуются объёмом реализующего их кода. Условно к малым задачам можно отнести программы длиной в несколько сот строк текста на алгоритмическом языке, к средним – длиной в несколько тысяч, а к большим – длиной в десятки тысяч строк. С организационной точки зрения здесь важно, реализуется ли задача одним программистом, несколькими исполнителями или большим коллективом. Понятно, что с увеличением и усложнением программы достаточно быстро (и нелинейно) растёт трудоёмкость её разработки (кодирование, отладка, верификация, валидация, тестирование, сравнительный анализ качества, оптимизация).

Другой стороной медали для оценки сложности и увеличения задачи является время её счёта на ЭВМ. Обычно малыми считаются задачи со временем решения порядка минуты, средними – со временем в десятки минут, а к большим относятся задачи, решаемые часами, десятками часов,

сутками и более. Любопытно, что здесь многие десятки лет остаётся справедливым инвариант Н.Н. Яненко: такое деление задач на категории не зависит от производительности рассматриваемого поколения компьютеров, поскольку здесь главным оказывается человеческий фактор, связанный с объективной сложностью организации крупномасштабного вычислительного эксперимента.

Объяснить этот феномен можно аналогией с проблемой "щита и меча" в историческом развитии оборонительных и наступательных средств военных технологий. В нашем случае это означает, что очередной скачок скорости машинных вычислений всегда вызывает новые потребности в повышении точности (и ресурсоёмкости) расчётов.

Ещё один важный фактор уровня сложности программы – объём обрабатываемых или усваиваемых данных, необходимых для проведения расчётов. В специальных случаях используемый компьютер может быть связан с целой системой сбора и обработки результатов натурных измерений: спутниковые данные, сейсморазведка, прогноз погоды и климата и т.д.

Важно отметить и другую особенность МПО как готового продукта. По своей сути технологии компьютерного моделирования призваны повышать производительность труда при проектировании или при эксплуатации каких-то установок или процессов. Это означает, в частности, что "софт", говоря языком экономики, относится к производствам средств производства, то есть к наиболее самоокупаемым индустриальным сферам. Более того, в программном продукте изначально заложена сверхприбыль, так как его многократное копирование происходит практически бесплатно (в то время как, например, в стоимость гвоздя входит цена не только его штамповки, но и металла).

В силу большой практической значимости, МПО должно иметь и высокую стоимость. Обычно его рыночная цена намного превышает затраты на непосредственную разработку, и сегодня не менее половины стоимости суперкомпьютера – это программное обеспечение, которое условно можно разбить на две составляющие. Первая – коммерческий "софт", мировые продажи которого исчисляются миллиардами долларов (не зря долгое время самым богатым человеком на планете был разработчик операционной системы Билл Гейтс), а вторая – свободно распространяемый. Существует даже теория, что сам "софт" должен быть бесплатным, а необходимые доходы его разработчика или владельца – складываться из услуг по эксплуатации.

Очевидно, что определённая часть программного обеспечения ориентируется на задачи обороны страны. Проблемы национальной безопасности – святая святых, рыночные подходы отходят здесь на второй план, но сама востребованность качественного МПО только возрастает.

Другой круг обязательных вопросов заключается в массовой подготовке квалифицированных кадров по разработке и использованию программного обеспечения. За рубежом уже давно обсуждаются и внедряются новые образовательные концепции STEM: Science, Technology, Engineering, Mathematics (имеются в виду различные науки и технологии, внедрение и обязательная математика, то есть фактически моделирование).

И наконец, следует сказать, что решение перечисленных задач государственного значения невозможно методами рынка, который "всё самоорганизует". Нужна обязательная инфраструктура в форме больших пользовательских ассоциаций, фондов или профессиональных сообществ типа SIAM (процветающее общество по индустриальной и прикладной математике, имеющее свои журналы, проводящее конференции, выпускающее книги). Но начало подобной организационной работы, особенно в случае дефицита ресурсов, невозможно без серьёзной правительственной поддержки [15]. Нынешний исторический момент в развитии прикладного программного обеспечения можно сравнить с переходом от ремесленного способа производства товаров к индустриальному, который в своё время обусловил качественный рост производительности труда и внутреннего валового продукта.

ПРОГРАММНЫЕ АСПЕКТЫ ТЕХНОЛОГИЧЕСКИХ СТАДИЙ МОДЕЛИРОВАНИЯ

По мере количественного и качественного развития математического и прикладного программного обеспечения, составляющего инструмент глобализации моделирования, новые методологии непрерывно систематизировались, обобщались и в итоге кристаллизовались в самостоятельную научную дисциплину, уверенно выдвигающуюся на ведущие позиции. Во второй половине XX в. значительную роль в этом процессе сыграли научные школы академиков Г.И. Марчука, А.А. Самарского и Н.Н. Яненко, заложивших концептуальные основы и технологии пакетов прикладных программ, модульных подходов и принципов организации крупномасштабных вычислительных экспериментов. Зачастую новые понятия рождались в жарких философских спорах, например, "является ли программный модуль объективной реальностью?" [16]. Компьютерное сообщество стало производить огромное количество библиотек, инструментария и создаваемых крупными корпорациями пакетов прикладных программ, включая "монстров" матема-

тического моделирования типа NASTRAN, ANSYS, Fenics, NGSolve и многих других, по которым сформировались свои пользовательские ассоциации. Отдельный рынок составляют системы автоматизированного проектирования типа CAD, CAE, CAM, PLM – незаменимые инструменты промышленного прогресса, претерпевающие сейчас конвергенцию с моделирующими технологиями.

Изначально пакеты прикладных программ создавались как проблемно-ориентированные комплексы для конкретных приложений с фиксированными структурами данных и архитектурой, но затем их функциональность искусственно расширялась, что неизбежно приводило к болезням роста технологического развития. Например, успешный коммерческий ANSYS просто скупал своих конкурентов и втискивал их разработки в свои.

Разобраться в структуре прикладного МПО помогает хорошо известная технологическая цепочка Н.Н. Яненко и А.Н. Коновалова, описывающая основные стадии наукоёмкого вычислительного эксперимента [4, 6], современное развитие которой представлено в статье [6], а также во введении к книге [17].

Формирование описания исходной модели решаемой задачи. Эта содержательная информация определяет дифференциальные и/или интегральные уравнения со всеми их коэффициентами, краевыми и начальными данными, а также геометрию расчётной области, которая в общем случае является многомерной, разномасштабной и имеет сложную топологию кусочно-гладкой границы. Здесь фигурируют резко отличающиеся по своим свойствам функциональные объекты, над которыми надо выполнять теоретико-множественные и другие операции, формулировать и решать содержательные оптимизационные проблемы. Значительную помощь здесь могут оказать САПР-разработки, но в целом этот круг вопросов требует создания специальной подсистемы (назовём её моделлер) для поддержки работы с математической постановкой, результатом которой в памяти компьютера должны быть геометрическая и функциональная структуры данных. С позиций программирования, на данной стадии нет проблемы больших данных, так как число объектов в непрерывной постановке – несколько десятков или сотен. Главная задача здесь – обеспечить пользователя наглядным и интеллектуальным входным интерфейсом [18]. Логическая и интеллектуальная сложность этого этапа может быть связана с использованием иерархии моделей для исследуемой задачи с целью возможного последовательного уточнения постановки и численных результатов для более глубокого изучения проблемы.

С одной стороны, суперкомпьютерные возможности накопления и обработки огромных объёмов данных породили в последние десятилетия различные направления Data Science, связанные со статистическим анализом или эмпирическими приёмами (метамоделирование, суррогатная оптимизация, нейронные сети и другие принципы, используемые в машинном обучении). Такие подходы, несомненно, имеют право на существование, но не являются предметом нашей статьи, тем более что это проблема методологии, но не программирования. С другой стороны, предусматриваемая “моделлером” возможность применения иерархии и избыточности различных постановок допускает комбинирование классических математических формулировок с приёмами метамоделирования.

Первый ресурсоёмкий алгоритмический этап решения задачи – превращение её непрерывной модели в дискретную, то есть построение расчётной сетки, от качества которой будет в значительной степени зависеть успех всего моделирования. Несмотря на кажущуюся простоту своих примитивов и операций над ними, сеточный трёхмерный конструктор, или генератор, – это в общем случае достаточно сложная программа, реализация которой требует немалых интеллектуальных ухищрений, усугубляемых зачастую большими объёмами данных, так как современные требования к точности заставляют доводить число узлов сетки до десятков и сотен миллиардов. Многообразие типов конечных элементов, или объёмов, а также их упорядоченностей открывает большие возможности для реализации сеток, адаптированных к особенностям задачи, но всё это ведёт к росту вычислительной сложности. Особенно трудоёмки динамически перестраивающиеся сетки, а также неструктурированные, в которых определение соседних объектов можно задать только их перечислением. В Интернете имеется достаточно большое число как дорогих, так и бесплатных сеточных генераторов (Netgen, GMesh и др.), однако проблема эта далеко не закрыта. Здесь большое поле деятельности как для эмпирики, так и для теоретических исследований в области вычислительной геометрии и топологии [17, 19].

Наиболее эффективные алгоритмы используют локальные сгущения, последовательности вложенных сеток, преобразования сеточных графов, а также декомпозиции расчётных областей с параметризованными пересечениями подобластей, в том числе формируемых распределённым образом в памяти различных процессоров. Перечисление перспективных технологий можно бы продолжить, но все они должны заканчиваться одним – формированием сеточной структуры данных, которая в совокупности с функциональной структурой данных на дискретном уровне полностью отображает постановку исходной за-

дачи. Основная проблема программирования конструкторов многомерных сеток – это, по-видимому, тщательное проектирование структур данных, от качества которых в значительной степени зависит экономичность решения задачи в целом.

Сеточная аппроксимация непрерывной модели. На этой стадии сильно различаются случаи стационарной или нестационарной задачи и их дифференциального и/или интегрального представления (конечно, существенно влияет также линейность или нелинейность исходных уравнений). Естественно, ресурсоёмкость алгоритмов растёт при увеличении плотности матриц, а также при необходимости многократного перевычисления их элементов. Программирование аппроксимационных методов значительно упрощается для однородных схем, когда для всех конечных элементов или узлов расчётные формулы одинаковы. Однако этого почти не бывает в наиболее продвинутых методах, когда, например, используются асимптотические свойства решения в окрестностях сингулярных точек.

Для многих схем конечных объёмов, конечных элементов и разрывных алгоритмов Галёркина различных порядков точности, к счастью, имеются, универсальные и естественным образом распараллеливаемые поэлементные технологии с экономичным вычислением локальных матриц и сборкой глобальной матрицы для всей дискретной задачи. Это позволяет единообразным способом автоматизировать построение алгоритмов аппроксимации для широкого класса исходных алгоритмов и открывает методологическую основу для создания нового типа программного продукта – библиотеки аппроксиматоров на представительном семействе разных сеток. Результат этого расчётного этапа состоит в формировании алгебраической структуры данных [20].

Важно отметить общую современную тенденцию аппроксимационных методов, обусловленную необходимостью минимизации дорогих коммуникационных операций – переход от простых формул первого или второго порядка к схемам повышенной точности, что позволяет резко сократить размеры сеток, объёмы используемой памяти и информационных обменов. Однако для математика-программиста это сопряжено с психологическими и техническими неприятными процедурами вывода и реализации длинных арифметических выражений, исчисляемых десятками и более страниц. Здесь очень поможет искусственный интеллект в форме систем автоматизированных аналитических преобразований типа Maple. Более того, такие средства уже имеются в широко распространённом языке Python.

Решение задач вычислительной алгебры. Эта стадия наиболее актуальна с точки зрения производительности МПО, так как при решении всевозможных проблем моделирования около 80% машинного времени уходит на решение системы линейных алгебраических уравнений, поскольку объёмы необходимых арифметических итераций и памяти растут нелинейно с увеличением числа степеней свободы. Именно по этой причине данная область математики наиболее продвинута программистами (см. представительный обзор [21] по имеющимся в открытом доступе разработкам).

Системы линейных алгебраических уравнений очень разнообразны: вещественные и комплексные, плотные и разреженные, эрмитовые и неэрмитовые, вырожденные и невырожденные и т.д. Все эти свойства существенно влияют на вычислительную схему, структуру данных и технику программирования, особенно при распараллеливании алгоритмов. Важна роль и сжатых форматов хранения матриц, для которых существуют определённые стандарты и соответствующие переводчики между ними. В значительной степени здесь помогает относительная ограниченность типовых матричных и векторных операций, для которых имеются доступные стандартные реализации, например, высококачественные библиотеки BLAS, SPARSE BLAS, MKL INTEL и многие другие [22].

С ростом мощности суперкомпьютеров всё более востребованными становятся большие системы линейных алгебраических уравнений (с порядками 10^9-10^{11} и выше), для которых наиболее экономичны итерационные процессы. Среди последних лидерами становятся предобусловленные методы в подпространствах Крылова с различными ортогональными, проекционными и вариационными свойствами. Для их ускорения применяются многообразные алгоритмы приближенной факторизации матриц, асимптотически оптимальные по порядку многосеточные подходы, а также декомпозиции областей, наиболее приспособленные к масштабируемому распараллеливанию алгоритмов.

Исследование методов, судя по потоку публикаций, продолжается, ежегодно появляются новые усовершенствования для алгебраических решателей. По мере их реализации мы получаем библиотеки программ с избыточным функциональным наполнением, позволяющим выбирать наилучший алгоритм решения конкретной задачи, что, в свою очередь, требует новых аналитических и технологических подходов к развитию МПО.

Решение обратных задач. В определённом смысле это главная цель моделирования, заключающаяся в отыскании неизвестных параметров модели или в оптимизации какого-либо динамического процесса. Универсальный подход в данном случае состоит в формировании последовательности параметризированных прямых задач,

а также в определении на их решениях целевого функционала, который требуется минимизировать за счёт подбора параметров. Итерационные процедуры их последовательного вычисления с многомерным решением прямых задач и составляют методы оптимизации, они активно развиваются в последние десятилетия. Основные достижения здесь связаны с алгоритмами внутренних точек, последовательного квадратичного программирования, доверительных интервалов и статистических подходов [23]. При использовании наиболее эффективных методов минимизации требуется вычислять производные целевого функционала по параметрам, или функции чувствительности. При этом для их определения приходится формировать и решать новые дополнительные задачи, зачастую с плохими свойствами.

В данных приложениях, в первую очередь, различаются проблемы локальной и глобальной оптимизации, которые связаны с существованием одного или нескольких минимумов целевого функционала. Если прямые задачи достаточно ресурсоёмки, то и реализация методов оптимизации сложна и длительна, особенно при "овражном" характере целевого функционала. Его формирование – технология на грани искусства, требующая зачастую серьёзного вычислительного опыта решения задач из области конкретных приложений. В таких ситуациях хороший эффект достигается при организации человеко-машинного взаимодействия.

Постобработка, визуализация и анализ результатов. Пользователю МПО необходимо оценить качественные и количественные характеристики расчётных результатов в показателях, естественных для исследуемой предметной области. Для убедительности представим себе машинный эксперимент по моделированию критических термоупругих напряжений спускаемого космического аппарата, используя метод конечных элементов на пространственной сетке с миллиардом узлов. Пусть при этом рассчитывается миллион временны́х шагов, а результат каждого из них – коэффициенты разложения вычисляемых функций температуры, давления и напряжения по десяти базисным функциям (в каждом конечном элементе). Пользователю требуется наглядно изобразить динамическую цветную 3D-картину, позволяющую сразу оценить слабые места. Очевидно, что быстро получить такую виртуальную реальность – очень важная, но весьма ресурсоёмкая задача, для решения которой создаются сверхбыстрые графические вычислительные ускорители типа NVIDIA. Здесь высокой производительности удаётся достичь за счёт относительно простой логики при выполнении большого объёма арифметических действий. Это обстоятельство также благоприятствует автоматизации построения операций на данном технологическом этапе (например, путём создания специализированного языка программирования). В последние десятилетия для этих целей разрабатывают специальные программно-аппаратные средства, обеспечивающие даже эффект присутствия. Наряду с такой формой интеллектуального пользовательского интерфейса важны и чисто программные решения, когда на основе анализа обработанных данных автоматизируется управление дальнейшим ходом вычислительного процесса.

Квинтэссенция машинного эксперимента – принятие решений по результатам исследования. На текущий момент опыт программирования в данной области искусственного интеллекта пока недостаточен, и мы ограничимся общим замечанием, что рассматриваемая здесь тематика – прерогатива математической логики, семантического моделирования, онтологических подходов и когнитивных технологий. Очевидно, что первых результатов для применения к математическому моделированию следует ожидать в частных решениях с игровыми подходами и ограниченными сценариями.

При использовании традиционного проблемно-ориентированного подхода программная реализация конкретной прикладной задачи предполагает последовательное прохождение всей выше рассмотренной технологической цепочки. Однако если рассматривать цель создания МПО в целом для разных классов приложений математического моделирования, то может оказаться более предпочтительным методо-ориентированный принцип, на котором мы остановимся в следующем разделе.

ЭВОЛЮЦИЯ ТЕХНОЛОГИЙ: ОТ ПАКЕТОВ ПРИКЛАДНЫХ ПРОГРАММ К ИНТЕГРИРОВАННЫМ ВЫЧИСЛИТЕЛЬНЫМ ОКРУЖЕНИЯМ

Несколько десятилетий назад в мировой практике прикладного программирования начала складываться тенденция к переходу от разработок пакетов прикладных программ к интегрированным вычислительным окружениям. Примером может служить проект DUNE (Distributed Unified Numerical Environment [24]), инициированный немецкими университетами, но затем активно расширявшийся за счёт добровольных участников международного консорциума. Среди аналогичных открытых к участию проектов можно отметить Open FOAM [25], а также INMOST (Институт вычислительной математики им. Г.И. Марчука, руководитель работ член-корреспондент РАН Ю.В. Василевский [26]). В статье [27] изложена концепция интегрированного вычислительного окружения, функциональное наполнение которого реализовывалось в Институте вычислитель-

ной математики и математической геофизики СО РАН в рамках базовой системы моделирования.

Технологические спецификации интегрированных вычислительных окружений. Предлагаемая архитектура разработки программ основывается на том положении, что каждая из семи описанных в предыдущем разделе технологических стадий реализуется соответствующей автономной подсистемой, ориентированной на поддержку своего класса алгоритмов и взаимодействующей с остальными подсистемами посредством согласованных структур данных, в том числе перечисленных выше. Такие подсистемы могут создаваться независимыми группами математиков-программистов, специализирующихся на достаточно узких направлениях работ и достигающих за счёт этого высокой профессиональной квалификации. Развиваемые при этом вычислительные процедуры, модули или функции специфицируются по содержанию и форме входными/выходными интерфейсами и интегрируются в необходимый прикладной программный комплекс средствами сборочного, фрагментарного или модульного программирования, по аналогии с интеллектуальным конструктором LEGO. Цель такого подхода к организации работ – кардинальное повышение совокупной производительности труда разработчиков, которая должна оцениваться пользователями программного продукта по соотношению цены и качества или срока разработки и производительности полученного кода.

С точки зрения создания не конкретного пакета прикладных программ, а решения отраслевых проблем МПО, можно сформулировать следующие технические требования к интегрированному вычислительному окружению:

• гибкое расширение состава реализуемых моделей, применяемых алгоритмов и технологий;

• адаптация к эволюции компьютерных платформ, представляющих высокий интеллектуальный потенциал;

• унифицированные структуры данных и эффективное переиспользование внешних программных продуктов;

• интеллектуальные внутренние и пользовательские входные/выходные интерфейсы;

• согласованное участие различных групп разработчиков.

Соблюдение перечисленных условий обеспечит длительный жизненный цикл проекта и его востребованность со стороны широкого круга разнопрофильных пользователей. Другими словами, прикладное программное обеспечение нового поколения должно представлять собой непрерывно развивающийся инструментальный комплекс для поддержки процесса безостановочного появления современных актуальных задач, математических моделей, вычислительных методов, информационных технологий, суперкомпьютеров и их массовых пользователей. Что касается эксплуатационных спецификаций МПО, то здесь в первую очередь следует назвать такие очевидные качества, как широта и адекватность заложенных моделей, оперативность получения результатов (например, счёт предпочтительней на минуты, а не на часы или сутки), а также удобство и надёжность использования. Важно подчеркнуть, что цель создания интегрированного вычислительного окружения заключается не только в повышении производительности труда отдельной группы разработчиков или пользователей МПО, но и в обеспечении интересов всего вычислительного сообщества, занимающегося математическим моделированием в рамках предлагаемой концепции проектных технологий.

Весьма серьёзный и содержательный вопрос качества программирования – обеспечение высокой производительности вычислений, то есть быстродействия ЭВМ при выполнении определённого класса алгоритмов для решения каких-то типов задач. Как уже видно из данной формулировки, само это понятие сложное и неоднозначное. Общепринятыми количественными критериями качества распараллеливания служат величины ускорения и эффективности использования процессоров:

$$S_p = T_1/T_p, \quad E_p = S_p/p,$$

где T_1 и T_p – время выполнения алгоритма (или решения задачи) на одном и на p вычислительных устройствах соответственно. При этом время работы компьютера складывается из временны́х затрат на арифметические операции и информационные обмены, которые качественно оцениваются формулами

$$T = T_a + T_c, \quad T_a = \tau_a N_a, \quad T_a = N_0(\tau_o + \tau_c N_c).$$

Здесь $\tau_a \ll \tau_c \ll \tau_o$ – среднее время выполнения одного арифметического действия, передачи одного числа и задержки одной транзакции соответственно, а N_a, N_c, и N_0 – количество выполненных арифметических операций, пересланных чисел и обменов данными.

Хотя приведённые соотношения очень грубы, они позволяют давать некоторые рекомендации по программированию алгоритмов на всех описанных выше стадиях моделирования: надо стараться минимизировать как объём пересылаемой информации, так и число таких передач, для чего целесообразно буферизовать файлы обмена, а также по возможности совмещать во времени пересылку данных с арифметическими действиями. Эти аспекты важны ещё и потому, что коммуникации – наиболее энергозатратные операции, они серьёзно сказываются на стоимости эксплуатационных расходов суперкомпьютеров.

Отметим, что технологии программирования могут быть разными при стремлении к масштабируемому параллелизму в сильном или в слабом смыслах. Первый означает сокращение времени счёта конкретной задачи пропорционально увеличению количества процессоров, второй – сохранение временны́х затрат при одновременном пропорциональном изменении трудоёмкости задачи и числа арифметических устройств. Также справедливости ради надо подчеркнуть, что мы говорим о производительности решения только одной задачи. Однако с точки зрения администратора вычислительного центра коллективного пользования оптимальные решения по распараллеливанию вычислений на потоке задач могут быть совсем другими.

Стратегии и тактики распараллеливания, конечно, определяются архитектурой суперкомпьютеров, которая, по-видимому, в ближайшее десятилетие не сильно отойдёт от нынешней гетерогенной структуры – соединённые шинами многопроцессорные узлы с центральными многоядерными устройствами CPU и специализированными ускорителями. Некоторые актуальные вопросы распараллеливания вычислений в рамках интегрированных вычислительных окружений представлены в статье [28]. Высокопроизводительные алгоритмы базируются на декомпозиции областей, в широком смысле этого слова, а их реализация осуществляется средствами гибридного программирования: межузловая передача сообщений на распределённой памяти, многопотоковые вычисления на иерархической общей памяти и векторизация операций (системы типа MPI, OpenMP и AVX соответственно). Эти инструменты выглядят сейчас достаточно архаично, и не случайно, что эффективность распараллеливания при их использовании зачастую достигает только 10–20%. В литературе имеется немало описаний языков параллельного программирования (например, CHAPEL), но они пока не доведены до уровня массового производственного применения, и эта проблема остаётся одной из ключевых на ближайшее пятилетие. В этой связи можно упомянуть такие перспективные исследования, как реализация максимального распараллеливания с помощью построения Q-детерминанты, которая представлена в работе [29], и автоматизированный синтез параллельных программ в [30].

Концепция и структура базы знаний математического моделирования. Не затрагивая когнитивные и философские аспекты, под базой знаний некоторой математической области будем понимать совокупность информации (базу данных) и программных инструментов, содержащих опыт исследований и правила вывода. Говоря более конкретно, мы рассматриваем интеллектуализацию технологий моделирования как создание информационно-вычислительного окружения, которое удовлетворяет определяющим свойствам базы знаний, включая все необходимые компоненты программной системы, обеспечивающей эффективное проведение и применение суперкомпьютерных экспериментов по изучению процессов и явлений. Подчеркнём, что цель формирования такого искусственного интеллекта – качественное повышение уровня человеко-машинного взаимодействия (но не замена *Homo sapiens* роботом!), о котором говорилось в пионерной работе А.П. Ершова и Г.И. Марчука [31]. Прототипом здесь можно считать возглавляемый Дж. Донгаррой и В.В. Воеводиным проект AlgoWiki [32] (в применении к параллельным алгоритмам и программам). Из профессиональных исследований на данную тему укажем также на работу [33] о роли онтологий и когнитивных технологий в проектировании.

В соответствии с представленным выше содержанием этапов жизненного цикла интегрированного вычислительного окружения определим базу знаний математического моделирования как структуру со следующими основными компонентами:

• конструктор моделей, включая описания геометрических и функциональных, или материальных, объектов, а также средств их редактирования и модифицирования, хранения типовых объектов, готовых (сформированных по правилам) методических или специальных примеров и практических задач;

• библиотеки алгоритмов (генераторы сеток, аппроксиматоров, алгебраических решателей, методов оптимизации и т.д.), в том числе как собственные разработки в рамках технологий проектных интегрированных вычислительных окружений, так и сторонние продукты;

• архивы расчётных заданий и результатов решения методических и практических задач со средствами их анализа и визуализации;

• пользовательская документация с демонстрационными и учебными версиями компонентов МПО;

• системы автоматизации, верификации и тестирования алгоритмов и программ;

• конфигуратор исполняемых программ из наборов вычислительных модулей, процедур и функций на основе технологий сборочного, модульного или фрагментарного программирования;

• систематизированные библиография, глоссарии и справочные материалы со ссылками на доступные в Интернете источники;

• система(ы) управления (статического и/или динамического) вычислительными процессами и сценариями;

• система принятия решений по результатам расчётов.

Одним из главных выводов, который должна уметь делать база знаний математического моделирования, – выбор наиболее предпочтительного алгоритма из доступного набора методов для решения конкретной задачи или подзадачи. Фактически принятие решения осуществляется, конечно, по заранее заложенным правилам, на основе сформулированных онтологий для изучаемых предметных областей. Определение же оптимального алгоритма чаще всего зависит от многих условий и само может представлять слишком сложную проблему. В более общем плане о базе знаний математического моделирования можно сказать, что это – актуальная форма для методологии глубокого машинного изучения как орудия познания (deep learning, machine learning – одни из самых распространённых понятий в работах по искусственному интеллекту [34, 35]).

Очевидно, чем больше требований к функциональности и интеллектуальности закладывается в процесс глубокого изучения объектов, тем сложнее требуется системное наполнение разрабатываемого МПО. Одна из ключевых проблем относится к языкам программирования, мода на которые меняется достаточно быстро. Если в XX в. велись острые дискуссии между сторонниками Алгола и Фортрана, продвинутых алгоритмических языков ADA и PL-I, то сейчас эти споры ушли в прошлое. Однако вопросы многоязыковости систем программирования имеют общий характер и остаются актуальными, ожидая своего эффективного решения в интегрированных вычислительных окружениях (то же самое относится и к многоплатформенности, то есть к возможности работы в различных многопользовательских операционных окружениях). Другие лингвистические аспекты связаны с интеллектуализацией, или “очеловечиванием”, языков естественного типа для математиков-программистов и многочисленных пользователей, далёких от информационных технологий, для которых общение с ЭВМ идеально как “программирование без программирования”. В книге [36] такая перспектива отождествляется с переходом от “палеоинформатики к неоинформатике” и образно связывается с созданием “фабрики языков”. Некоторый положительный опыт в данной области уже имеется, примером чему может служить система программирования для описания научных интерфейсов SIDL (Scientific Interface Definition Language) [37].

В больших прикладных разработках возникает серьёзная проблема автоматизации построения единой внутренней структуры, состоящей из многочисленных программных фрагментов. В этой области в течение многих лет развиваются такие известнее проекты, как DCOM (Distributed Component Object Management), CCA (Common Component Architectures), инструментальный комплекс CCAFFENE [38] для осуществления технологических операций в распределённой памяти, а также методология сервис-ориентированных архитектур SOA (Service Oriented Architecture), предполагающая кардинальное повышение производительности труда программистов-разработчиков и сокращение сроков трудоёмких процессов отладки программ, верификации и тестирования алгоритмов.

В последние годы огромную популярность набирает система веб-сервисов GitHub (а также её общедоступная версия GitLab) для многопользовательской поддержки всевозможных программных продуктов, де-факто сформировавшая международную социальную сеть армии разработчиков. В частности, она предполагает создание дерева репозитариев, как закрытых, так и открытых для сотрудничества и кооперативных проектов, в том числе в рамках облачных вычислений [39]. Этот успешный проект демонстрирует, в частности, бесконфликтное сосуществование коммерческих и бесплатных разработок, что свидетельствует о преувеличенной в некоторых публикациях фетишизации принципов открытых систем, которые, несомненно, имеют важное значение, но скорее для отношений собственности и маркетинга, а не для решения фундаментальных проблем и разработки технологий программирования.

* * *

Программирование как основа супервычислений и математического моделирования с нарастанием темпов 4-й индустриальной революции обретает всё больший вес в научно-технической и социально-гуманитарной сферах, играя жизнеобеспечивающую роль кровеносной или нервной системы мирового сообщества. В этой связи интересно посмотреть на МПО с точки зрения важнейшей проблемы современности – устойчивого развития. Следует различать лицевую и оборотную стороны медали. Последняя связана с тем, что программы для суперкомпьютерного моделирования в целом – это обширнейшая экосистема, отличающаяся многообразием решаемых фундаментальных и прикладных задач с большим количеством вычислительных модулей, объёмом перерабатываемых данных и числом разнопрофильных пользователей. Как добиться успешного длительного существования такой структуры, эффективных инноваций в ней – об этом шла речь в предыдущих разделах статьи. Здесь же упомянем ещё одну важную проблему – она касается философии, теории и практики управления большими программными проектами. В книге “Мифиче-

ский человеко-месяц” [40] талантливый анализ фундаментальных и прагматических вопросов представлен Ф. Бруксом – одним из руководителей разработок операционной системы IBM. Позднее подробный обзор методологий своего времени был проведён в работе [41]. Из современных подходов к менеджменту программных продуктов популярностью пользуются гибкие принципы Agile для крупных и средних по своему размеру команд разработчиков.

Важно отметить, что проблема активных массовых разработок МПО в целом (академического, образовательного, коммерческого) вследствие своей масштабной наукоёмкой и возрастающей практической значимости требует творческой консолидации и серьёзной научно-организационной работы, некоторые аспекты которой рассмотрены в статье [15]. А поскольку затрагиваются общенациональные интересы, вопросы стратегического развития МПО нуждаются в государственном планировании, государственной поддержке.

Что касается прогностической миссии программирования, то наряду с другими областями научных знаний математическое моделирование должно помочь построить дорожную карту устойчивого развития цивилизации. В 2015 г. Генеральная Ассамблея ООН приняла Декларацию “Преобразование нашего мира. Повестка в области устойчивого развития на период до 2030 года”. Этот документ включает 17 целей. Очевидно, что глобальная по своим масштабам миссия выполнима только при активном вовлечении комплексных научных исследований в новые экономические, политические, социальные и гуманитарные сферы с использованием суперкомпьютерных вычислительно-информационных технологий. Формально планетарные процессы (межгосударственные конфликты, климатические риски, пандемии, природные и техногенные катастрофы, демографические проблемы и др.) могут быть описаны и изучены математически как некоторые динамические системы. Однако здесь мы зачастую вторгаемся в область менталитета, где предсказательное моделирование должно включать в себя такие субъективные категории, как пока ещё трудно прогнозируемый человеческий фактор. Генеральная линия развития видится на путях такой, казалось бы, абстрактной науки, как математическая логика, закладывающей основы семантического моделирования, систем принятия решений, более глубокого понимания коллективного разума *Homo sapiens*.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации (код проекта FSUN-2020-0012).

ЛИТЕРАТУРА

1. *Ильин В.П.* Вычислительная информатика: открытие науки. Новосибирск: Наука, 1991.
2. *Любимский Э.З., Поттосин И.В., Шура-Бура М.Р.* От программируемых программ к системам программирования (российский опыт) // История информатики в России. Учёные и их школы, М.: Наука, 2003. С. 252–261.
3. *Ильин В.П.* Сибирская информатика: школы Г.И. Марчука, А.П. Ершова, Н.Н. Яненко // История информатики в России. Учёные и их школы. М. Наука, 2003. С. 340–363.
4. *Ершов А.П., Ильин В.П.* Пакеты программ – технология решения прикладных задач. Новосибирск: ВЦ СО АН СССР. Препринт № 121, 1978.
5. *Самарский А.А., Михайлов А.П.* Математическое моделирование. М.: Физматгиз, 2002.
6. *Яненко Н.Н., Коновалов А.Н.* Некоторые вопросы теории модульного анализа и параллельного программирования для задач математической физики и механики сплошной среды // Современные проблемы математической физики и вычислительной математики. М.: Наука, 1982. С. 200–217.
7. *Яненко Н.Н., Рычков А.Д.* Актуальные проблемы прикладной математики и математического моделирования. Новосибирск: Наука, 1982.
8. *Cottrell J., Hughes T., Bazilevs Y.* lsogeometric Analysis: Towards Integration of CAD and FEA. Wiley, Singaporer, 2009.
9. *Ершов А.П.* Альфа-язык. Энциклопедия кибернетики / Ред. В.М. Глушков. Киев: Глав. ред. УСЭ, 1974. С. 111–113.
10. *Benkner S., Lonsdale D., Zuma H.P.* The HPF – Project: Supporting HPF for Advanced Industrial Application // Proceedings EuroPAR 99. V. 1685.
11. DVM Systems. http//www.keldush.ru /dvm
12. IESP/www/exascale.org/iesp
13. *Il'in V.P.* Problems of Parallel Solution of Large Systems of Linear Algebraic Equations // J. Math. Sci. 2016. V. 216. № 6. P. 795–804.
14. *Dongarra J., Grigori L., Higham N.J.* Numerical Algorithms for High Performance Computational Science // Phil. Trans. R. Soc. 2020. A 378.
15. *Ильин В.П.* Как реорганизовать вычислительные науки и технологии // Вестник РАН. 2019. № 2. С. 232–242.
16. *Ильин В.П.* Математическое моделирование: философия науки // Сб. науч.-попул. статей “Математика, механика и информатика”, 2017. С. 8–16.
17. *Ильин В.П.* Математическое моделирование. Ч. 1. Непрерывные и дискретные модели. Новосибирск: Изд-во СО РАН, 2017.
18. *Il'in V.P.* Artificial Intelligence Problems in Mathematical Modeling // Springer Nature Switzerland AG 2019 / V. Voevodin, S. Sobolev (eds.). RuSCDays 2019. CCIS 1129. P. 505–516.
19. *Il'in V.P.* Integrated Computational Environment for Grid Generation Parallel Technologies // Springer Nature Switzerland AG 2020 / L. Sokolinsky, M. Zymbler (eds.). 2020. V. CCIS 1263. P. 58–68.

20. *Бутюгин Д.С., Ильин В.П.* CHEBYSHEV: принципы автоматизации построения алгоритмов в интегрированной среде для сеточных аппроксимаций начально-краевых задач // Труды Международной конференции ПАВТ'2014. Челябинск: изд-во ЮУрГУ, 2014. С. 42–50.
21. *Dongarra J.* List of freely available software for linear algebra on the web (2006). http://netlib.org/utk/people/JackDongarra/la-sw.htm
22. *Il'in V.P.* On an Integrated Computational Environment for Numerical Algebra // Springer Nature Switzerland AG 2019 / L. Sokolinsky, M. Zymbler (eds.). PCT 2019. V. CCIS 1063. P. 91–106.
23. *Il'in V.P.* The Integrated Computational Environment for Optimization of Complex Systems // Proceed. 2019 15th International Asian School-Seminar "Optimization Problems of Complex Systems (OPCS-2019)". P. 65–67.
24. *Bastian P., Blatt M., Dedner A. et al.* The Dune Framework: Basic Concepts and Recent Developments, Computers and Mathematics with Applications, 2020. DOI.org/10.1016/j.camwa.2020.06.007
25. OpenFOAM. https://www.openfoam.com/
26. INMOST: A Toolkit for Distributed Mathematical Modeling. https://www.inmost.org
27. *ll'in V.P.* The Conception, Requirements and Structure of the Integrated Computational Environment. B Supercomputing 4 th Russian Supercomputing Days, RuSCDays 2018. Revised Selected Papers. P. 653–665 (Communications in Computer and Information Science). V. 965. Springer-Verlag GmbH and Co. KG.
28. *Il'in V.* Parallel Intelligent Computing in Algebraic Problems // Sokolinsky L., Zymbler M. (eds). Parallel Computational Technologies. PCT 2021. Communications in Computer and Information Science. V. 1437. Springer, Cham.
29. *Aleeva V.* Designing Parallel Programs on the Base of the Conception of ?-Determinant // Supercomputing. RuSCDays 2018 (Communications in Computer and Information Science (CCIS)). 2019. V. 965. V. Voevodin, S. Sobolev (eds.). Springer, Cham. P. 565–577.
30. *Akhmed-Zaki D., Lebedev D., Malyshkin V., Perepelkin V.* Automated Construction of High Performance Distributed Programs in LuNA System // Parallel Computing Technologies. PaCT 2019 (Lecture Notesin Computer Science. V. 11657) // V. Malyshkin (ed.). Springer, Cham. P. 3–9. https://doi.org/10.1007/978-3-030-25636-4_1
31. *Ershov A.P., Marchuk G.I.* Man-machine Interaction iп Solving a Certain Class of Differential Equations // Proceed. IFIP Congress. New York, 1965. P. 550– 551.
32. *Antonov A., Dongarra J., Voevodin V.* AlgoWiki Project as an Extension of the Top500 Methodology // Supercomputing Frontiers and Innovations. 2018. V. 5. № 1. P. 4–10. https://doi.org/10.14529/jsfi18010
33. *Borgest N.M.* Key Terms the Ontology of Designing Review // Anal. Gen. Ontol. Des. 2013. V. 3. № 9. P. 9–31.
34. *LeCun Y., Bengio Y., Hinton G.* Deep learning // Nature. 2015. V. 521. P. 436–444. https://doi.org/10.1038/nature14539
35. *Weinan E.* Machine Learning and Computational Mathematics // Commun. Comput. Phys. 2020. V. 28. № 5. P. 1639–1670. https://doi.org/10.4208/cicp.oa-2020-0185
36. *Kleppe A.* Software Language Engineering: Creating Domain Specific Language Using Metamodels. N.Y.: Addison Wesley, 2008.
37. *Kohn S., Kumfert G., Painter J., Ribben C.* Divorcing Language Dependencies from a Scientific software Library // http://computation. .llnl.gov /casc /components/docs/2001statpp.pdf
38. *Allan B., Armstrong R., Wolfe A. et al.* The CCA Core specification in a Distributed Memory // SPMD Framework Concurrent Practice and Expedience. 2002. V. 14. P. 323–345.
39. *Feoktistov A., Kostromin R., Sidorov I.A., Gorsky S.A.* Development of distributed subject-oriented applications for cloud computing through the integration of conceptual and modular programming // Proceed. of the 41st International Conference on Information and Communication Technology, Electronics and Microelectronics. IEEE, 2018. P. 256–261.
40. *Брукс Ф.* Мифический человеко-месяц, или Как создаются программные системы. М.: Символ-Плюс, 2010.
41. *Скопин И.Н.* Основы менеджмента программных проектов. Курс лекций. Учебное пособие. М.: ИНТУИТ. РУ, 2004.