ПРОГРАММИРОВАНИЕ, 2016, № 1, с. 10–25

УДК 681.3.06

О ПРОИЗВОДИТЕЛЬНОСТИ И ИНТЕЛЛЕКТУАЛЬНОСТИ СУПЕРКОМПЬЮТЕНОГО МОДЕЛИРОВАНИЯ *

© 2016 г. В.П. Ильин[1, 2], И.Н. Скопин[1, 2]

[1] *Институт вычислительной математики и математической геофизики СО РАН*
630090 г. Новосибирск, пр-т Академика Лаврентьева, 6
[2] *Новосибирский государственный университет*
630090 г. Новосибирск, ул. Пирогова, 2
E-mail: iskopin@gmail.com

Поступила в редакцию 12.02.2014

Традиционно понятие суперкомпьютерных технологий связывается с отображением алгоритмов на архитектуру ЭВМ, что при бурном росте вычислительных мощностей означает необходимость примерно такого же увеличения быстродействия разрабатываемых алгоритмов и программ. В то же время хорошо известно, что темпы наращивания "компьютерных мышц" намного превосходят скорость повышения производительности труда разработчиков прикладного программного продукта, и это становится узким местом в эволюции компьютерных инноваций. Единственный путь в разрешении этой проблемы — это автоматизация построения моделей, алгоритмов и программ, что напрямую означает кардинальное изменение уровня искусственного интеллекта в суперкомпьютерных технологиях. В данной работе именно с этих позиций рассматриваются основные вычислительные стадии в математическом моделировании различных процессов и явлений, отмечаются аспекты высокой логической сложности современных высокопроизводительных методов расчета реальных "больших" практических задач и предлагаются некоторые интеллектуальные решения возникающих проблем.

1. ВВЕДЕНИЕ

Построение наукоемкого программного обеспечения для решения сверхбольших задач математического моделирования на современных ЭВМ постпетафлопсного уровня сталкивается с двумя существенно отличающимися друг от друга проблемами алгоритмического характера. Первая из них традиционно ставится во главу угла и заключается в отображении алгоритмов на архитектуру многопроцессорной вычислительной системы (МВС) с целью достижения максимального быстродействия, или минимизации вычислительной сложности методов, при решении реализуемой задачи. Второй аспект связан с автоматизацией построения алгоритмов и программ, что определяет, в конечном счете, производительность труда прикладного программиста, темпы роста которой, как известно, намного отстают от скорости наращивания мощностей компьютерного оборудования. Более того, как отмечается в "дорожной карте" экспертов IESP (International Exascale Software Project [1]), предстоящее в ближайшие годы вступление в эру экзафлопсных ЭВМ с сотнями миллионов ядер означает переход количества в качество и ставит перед мировым вычислительным сообществом проблему-вызов создания нового поколения программного обеспечения (X-Stack), способного обеспечить эффективное использование экстремально больших вычислительных ресурсов.

Кардинальное решение в данном направлении видится, прежде всего, как значительное повышение уровня интеллектуальности разработки прикладных программ, их пакетов и комплексов. Мы рассматриваем уровни интеллектуальности

*Работа поддержана грантом РНФ № 14-07-0048.5.

как степень творческой составляющей в работе экспертов прикладных областей, отвечающих за соответствие замысла предлагаемой программной системы реальным потребностям пользователей: математиков, конструирующих формализованные постановки задач и алгоритмы, и программистов, которые должны реализовать требуемое программное обеспечение. Творчество в разных его аспектах для указанных разработчиков всегда сопровождает их деятельность, когда создается принципиально новый продукт. Вместе с тем, в противоположность технологии, следование которой в идеале гарантирует своевременное получение продукта должного качества, творчество в производстве — реальный источник неопределенности и риска. По этой причине преодоление обозначенных выше проблем требует создания не технологий, а технологической поддержки, обеспечивающей максимально возможную автоматизацию рутинных процессов, и свободу принятия творческих решений в условиях предоставления полной и достоверной информации, связанной с каждым из альтернативных вариантов. Такая поддержка строится на базе общих средств разработки и специализированных инструментов, предусматриваемых для решения задач определенного класса.

Примером хорошо организованной технологической поддержки может служить конструирование компиляторов. После появления работ Д. Кнута [2, 3] с весьма высоким уровнем интеллектуальности стала возможной стандартизация процесса разработки, на основе которой построены различные инструменты поддержки конструирования front end'a — начальных фаз компиляции (см., например, [4]), не предполагающие от разработчика творческих усилий. В то же время, такой же технологичности разработки фаз back end'a, связанных с архитектурно зависимой оптимизацией, достичь не удается, в частности, из-за многообразия и сложности вариантов задания оптимального кода. Эта часть компилятора требует от разработчиков высокий уровень интеллектуальных способностей.[1]

[1]В публикации [5] приведены примеры других программных решений, анализ которых показывает, что при соответствующей интеллектуальной проработке они могут привести к появлению новых методов повышения робастности распределенных вычислительных систем.

Следует подчеркнуть, что интеллектуальность, связанная с проработкой хороших идей, теорий, методов и алгоритмов и привнесением их в инструмент программирования, делает поддержку разработки более технологичной, не требующей в проекте той творческой работы, которая выполнена заранее. Эта интеллектуальность исходит из специфики области прикладной программы. Примерами такой специализированной поддержки могут служить программные инструменты, преодолевающие логическую сложность работ: решение "больших" функциональных систем уравнений, оперирование с громоздкими формулами, построение алгоритмов с повышенной точностью вычислений, усилением устойчивости вычислительных процессов и другими математическими особенностями моделей или алгоритмов, определяющими наукоемкость прикладного программного обеспечения. А вот степень автоматизации этих аспектов характеризует уровень интеллектуализации соответствующих вычислительно-информационных технологий.

Наряду с проблемно-ориентированной специализированной поддержкой, свой вклад в технологичность проектной деятельности вносят технические, системные и организационно-методические общеупотребительные средства, и методы разработки, которые не связаны со спецификой проекта. Каждый из видов поддержки проектной деятельности развивается с внедрением своих достижений, основанных на актуализации своих интеллектуальных проработок. Какая из них вносит больший вклад в интеллектуальность проекта, сказать трудно. Заметим, что при разработке проекта общая поддержка создает условия для эффективного применения специализированной поддержки, и этот комплекс повышает уровень технологической поддержки проектов, а значит, и производительности, и качества труда программистов.

В настоящей работе мы, в основном, обсуждаем и оцениваем интеллектуальность специализированной поддержки и говорим о других аспектах технологизации проектной деятельности лишь в той мере, в какой она способствует решению задач вычислительного программирования, связанных с математическим моделированием.

Работа построена следующим образом. Раздел 2 представляет обзор процессов, так или иначе выполняемых при организации крупномасштабных компьютерных экспериментов. Далее, в разделе 3 обсуждаются методические аспекты стадий постановки и проведения вычислительных экспериментов, которые в конкретных проектах реализуются как этапы разработки математических моделей, проведения расчетов и принятия решений на основе моделирования. Четвертый раздел описывает возможные пути автоматизации построения алгоритмов, как средства кардинального повышения производительности труда в функциональном и системном наполнении пакетов прикладных программ (ППП). Завершает статью обсуждение актуальных проблем интеллектуализации и автоматизации построения алгоритмов.

2. ПОСТАНОВКА И ПРОВЕДЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Ситуация роста требуемого уровня специализированной интеллектуальности характерна для сфер производства программного обеспечения, где одним из основных критериев качества является продуктивность использования ресурсов МВС при вычислениях. Одной из наиболее важных таких областей является рассматриваемый в настоящей работе класс задач, описывающих реальные практические процессы и явления. В этих приложениях используются междисциплинарные прямые и обратные начально-краевые задачи для систем дифференциальных и интегральных уравнений (или соответствующих вариационных постановок). Обычно задачи ставятся в многомерных расчетных областях со сложной конфигураций многосвязных кусочно-гладких границ и контрастными материальными свойствами сред, что обуславливает потребности развитой математизации моделирования.

Технологические цепочки постановки и проведения крупномасштабных вычислительных экспериментов включают большое разнообразие работ. На самом общем уровне рассмотрения можно выделить следующие стадии, или этапы вычислительных проектов, характеризующие разработку моделей и проведение расчетов:

- *геометрическое и функциональное моделирование*, в ходе которого конкретизируется расчетная область и ее подобласти со свойствами решаемых уравнений, определяющими требования к применению расчетных схем;
- *дискретизация задачи моделирования*, представленной на предыдущей стадии в непрерывной постановке. Этот этап предусматривает генерацию сеток, в том числе адаптивных неструктурированных, учитывающих особенности поставленной проблемы;
- *аппроксимации* высоких порядков исходных соотношений уравнениями на сеточной структуре, полученной на предыдущей стадии, в результате чего постановка задачи алгебраизируется: задача получает представление в виде системы линейных и нелинейных уравнений, которые часто оказываются трудоемкими для решения и плохо обусловленными;
- *решение алгебраических задач*, в том числе систем уравнений, построенных при аппроксимации, с использованием методов, учитывающих особенности возникающих матриц и позволяющих преодолевать проблемы большой размерности, трудоемкости и плохой обусловленности;
- *постобработка* с целью приведения первичных результатов расчета к формату, пригодному для дальнейшего анализа и использования.

При решении обратных задач необходима специальная стадия:

- *Организация итераций* для проведения оптимизационных расчетов, которая заключается в вычислении целевых функционалов и обновлении краевых условий и других параметров для возврата к предыдущим стадиям, если требуется продолжение процесса поиска решения.

Последние две стадии связаны с использованием полученной при моделировании информации:

- *визуализация результатов расчета*, т.е. подготовка их для диалоговой обработки на заключительной стадии;

- *принятие решений* на основе математического моделирования, которое предусматривает валидацию результатов, их осмысление и выбор итоговых действий.

Перечисленные стадии и специальные условия, в которых они выполняются, определяют особые требования к процессу создания соответствующего прикладного программного обеспечения. В первую очередь здесь следует упомянуть о необходимости формирования системного подхода к обозначенной большой проблеме и комплексного решения задач, связанных с моделированием. Непрерывное развитие математических моделей и алгоритмов, предполагающее длительный жизненный цикл продукта, требует, чтобы время воплощения этих достижений в программных разработках было минимально, а постоянная эволюция компьютерных архитектур предопределяет задачу адаптации программного обеспечения к новым возможностям МВС. Успешное решение проблем современного и перспективного моделирования невозможно без скоординированного объединения усилий многочисленных групп разработчиков алгоритмов и программ, а также конечных пользователей.

Осознание потребности интеграции приводит к появлению идей и попыток их реализации в программных системах технологической поддержки моделирования. Примерами таких интеграционных проектов, в разной степени отражающих представленные только что аспекты интеграции, являются OpenFOAM [6] и DUNE (Distributed Unified Numerical Environment [7]). В этом ряду проектов следует отметить базовую систему моделирования БСМ [8], в которой заложены принципы разработки и эксплуатации программных инструментариев для всех основных типов задач математической физики: электромагнетизм, напряженно-деформированные состояния твердого тела, гидро-газодинамические течения, многофазный тепломассо-перенос и т.д. Эта система еще не может рассматриваться как программный продукт для независимого использования, но полученный опыт разработанных ее компонентов уже сегодня позволяет сделать вывод о перспективности развиваемого подхода, основой которого является выделение ядра, допускающего наращивание предоставляемых методов и адаптацию к используемому оборудованию.

3. МЕТОДИЧЕСКИЕ АСПЕКТЫ РЕАЛИЗАЦИИ РАСЧЕТНЫХ СТАДИЙ МОДЕЛИРОВАНИЯ

Перечисленные стадии постановки и проведения вычислительных экспериментов в реальных условиях конкретизируются как проектные этапы разработки, и каждый из них нуждается в поддержке, учитывающей как особенности выполняемых работ, так и связи с другими этапами.

Далее рассматриваются основные особенности этапов, характерных для проектов, реализующих моделирование. При изложении мы заостряем внимание на логической сложности функционального и алгоритмического наполнения этапов. Представленные положения сформировались в результате концептуальной проработки системы БСМ, а также анализа опыта реализации и первоначального использования компонент, которые предоставляет ядро БСМ (см. [8–13]).

3.1. Геометрическое и функциональное моделирование

Решение начально-краевой или иной задачи, лежащей в основе построения математической модели, начинается с задания исходной информации с помощью графических и текстовых средств. Этот этап проекта соответствует стадии геометрического и функционального моделирования: описываемая математическая модель содержит в общем случае представление расчетной многомерной области и ее геометрических свойств. Она может быть ограниченной или неограниченной, со сложной конфигурацией многосвязной кусочно-гладкой границы, на различных сегментах которой задаются разного вида начальные и/или краевые условия. Расчетная область может состоять из подобластей с разнотипными материальными свойствами. В них решаются “свои” системы дифференциальных и/или интегральных уравнений с соответствующими характерными для них видом членов и коэффициентов, задаваемых с помощью значений, формул, и/или функциональных соотношений. Всё это в совокупности составляет функциональные объекты модели.

Важным классом проблем являются нестационарные, в которых данные меняются со временем в процессе решения, а также нелинейные, когда, например, положение границы или значения коэффициентов зависят от искомого решения и также меняются в ходе вычислительного процесса. Кроме того, отдельного рассмотрения требуют обратные задачи, где исходные данные содержат некоторые варьируемые параметры, значения которых необходимо оптимизировать по дополнительным условиям, заключающимся в минимизации какого-то зависящего от решения целевого функционала, при задаваемых линейных или нелинейных ограничениях на искомые параметры.

Очевидно, что геометрические объекты могут быть описаны многими способами. Например, сфера однозначно задается положением своего центра и радиусом, а также подходящим уравнением в полярных или декартовых координатах, при этом первый способ является более экономным. Аналогично могут задаваться усеченный конус, цилиндр, параллелепипед и другие типовые, но более сложные фигуры. С другой стороны, поверхности первого или второго порядка определяются коэффициентами своих уравнений, а образуемые с их помощью тела можно идентифицировать путем нахождения линий пересечения и применения теоретико-множественных операций. Наконец, расчетную область можно построить на экране вручную с помощью стандартных графических средств.

В различных ситуациях целесообразность того или другого способа задания меняется, и поэтому естественно предусмотреть избыточную множественность представления геометрических (и функциональных) данных, с возможностями их взаимной конвертации. Например, большое количество геометрических форматов используется в САПРовских продуктах [14]. Надо также иметь в виду, что геометрическое моделирование должно предоставлять средства для выполнения различного типа операций над объектами: сдвиги, повороты, масштабирование, копирование и т.п. Достаточно детальный анализ данных технологических вопросов обсуждается в [10].

Из сказанного выше видно, что многообразие постановок задач и вариантов подходов к их решению, а также методов задания геометрических и функциональных объектов моделей, иными словами — уровень интеллектуальности первой стадии математического моделирования, не оставляет надежды на существование реальной технологии для нее. Вместе с тем, это же указывает на актуальность разработки технологической поддержки, суть которой заключается в своеобразной инвентаризации типов решаемых задач и методов, относящихся к этим типам, а также условий их применимости. Необходимо, чтобы разработчику конкретной модели всегда были доступны проверяемые критерии оценки выбираемых вариантов решений.

Другой аспект требуемой поддержки связан с тем, что во многом аналитическая работа первого этапа закладывает основы всей последующей проектной деятельности, включая возможность, а зачастую и необходимость проверки альтернативных решений. Как следствие, система поддержки должна строиться из расчета на осуществимость идентификации операционных маршрутов развития проекта, на обеспечение необременительных возвратов к ранее пройденным работам для их повторения в альтернативных вариантах, в том числе, к формированию математической модели для оптимизации построений, идентификации параметров и других действий. С целью преодоления риска проектного хаоса, чреватого неизбежными ошибками, и облегчения сравнения вариантов построения модели, а также для возможности использования результатов ранее выполненных работ очень важно обеспечить хранение и извлечение нужной информации о проведенных действиях. Иными словами, требуется, чтобы первый этап моделирования предусматривал построение репозитария проекта с развитой системой версионного контроля.

К сожалению, в существующих инструментальных системах поддержка операционных маршрутов развития проекта чаще всего игнорируется. В качестве одного из характерных примеров можно привести систему SALOME [15], которая была задумана как связующее программное обеспечение для CAD-CAE. По мере своего развития система трансформировалась в платформу поддержки математического моделирования, предлагающую ряд модулей

для использования при организации вычислительных экспериментов. Набор инструментов платформы достаточен для решения многих задач, однако отсутствие обязательных регламентов и средств проверки корректности их использования в конкретных условиях приводит к тому, что при построении моделей приходится обращаться к предлагаемой в SALOME документации. Однако содержащиеся в ней рекомендации не в состоянии заменить автоматизированные средства специальной поддержки надежности постановки экспериментов.

С учетом представленной специфики этапа можно сформулировать следующие положения, характеризующие требуемую для него технологическую поддержку:

- в ходе выполнения работ этапа выявляются математические свойства, от которых зависит план организации расчетов (существование и единственность решения, устойчивость и др., с одной стороны, а с другой — особенности расчетной области и ее подобластей как объектов моделирования);

- этап заканчивается определением математической модели, представляющей реальную задачу в виде математического объекта, а также расчетных методов для этого объекта;

- в результате этапа создаются *геометрическая и функциональная структуры данных* модели (ГСД и ФСД), которые принимаются как основа модели для задачи, решаемой в проекте;

- технологическая поддержка этапа связывается с общесистемными средствами работы, обеспечивающими доступ к исходной информации о задаче и возможность составления и редактирования ГСД и ФСД с поддержкой версионности для прямого использования на всех других этапах (в частности, для контроля версий);

- верификация полученных результатов — компетентная экспертиза, целью которой является подтверждение качества отражения изучаемых процессов с точки зрения условий задачи моделирования. Эта деятельность поддерживается обеспечением доступности информации о задаче и ее представлении в ГСД и ФСД.

Первая стадия математического моделирования связана с принятием ключевых для проекта стратегических творческих решений, а потому среди других расчетных работ ее непосредственная интеллектуальность постановочного этапа геометрического и функционального моделирования наиболее высока. Вместе с тем, для повышения производительности труда разработчиков на этом этапе и в дальнейшем необходим высокий уровень привнесенной интеллектуальности общей и специальной технологической инструментальной поддержки.

3.2. Дискретизация начально-краевой задачи

На основе ГСД и ФСД производится дискретизация исходной непрерывной постановки. Этот второй этап разработки заключается в построении адаптивной квазиструктурированной сетки. Последний термин означает, что генерируемая сеточная расчетная область может разбиваться на сеточные подобласти, в каждой из которых используются различные типы ячеек, или конечных элементов: тетраэдры, призмы и всякого вида многогранники, в которых некоторые грани могут быть криволинейными.

Существует огромное число публикаций по алгоритмам построения сеток, по определению критериев их качества и по программным реализациям, достаточно содержательный обзор этих вопросов имеется в [11]. Важно отметить, что современные эффективные методы решения краевых задач используют такие достаточно сложные подходы, как локальные сгущения, многосеточные технологии и декомпозиция областей, которые требуют специальных программных средств поддержки. Технологические инструменты этапа — генераторы сеток и средства разбиения области с учетом критериев качества. Результатом работы данного этапа для конкретного проекта является *сеточная структура данных* модели (ССД), отображающая в совокупности с ГСД и ФСД все свойства дискретизированной рассматриваемой задачи. Эта структура может быть статической, т.е. не меняющейся

в ходе последующих расчетов, либо динамической, когда требуемая сетка уточняется или даже переопределяется в ходе выяснения некоторых свойств модели в процессе вычислений. Простейший пример — достижение требуемой точности за счет выяснения необходимости сгущения сетки вокруг некоторых особых точек расчетной области на основе априорной и/или апостериорной информации.

Специализированная технологическая поддержка этапа дискретизации сводится к организации выбора и автоматизации метода построения сетки. Специфика поддержки этапа — необходимость обеспечивать вариантность задания сеточной области и ее подобластей с учетом их особенностей и способов задания начальных и краевых условий. В частности, для динамической ССД требуется обеспечить параметризацию этой структуры данных. Важно, чтобы разработчик мог оперативно получать информацию об особенностях, чтобы система поддержки указывала на возможное несоответствие дискретизации требованиям точности, на другие ошибки.

Управление процессом дискретизации связывается со следующими возможностями, предоставляемыми пользователям:

Общесистемная технологическая поддержка предусматривает возможность возврата к предыдущему состоянию ССД для коррекции выполненных действий и средства возобновления этапа после выполнения последующих этапов для сравнения вариантов конструируемой модели с частичным использованием результатов прежних дискретизаций. Требование поддержки подобных возвратов является исключительно важным не только для дискретизации, но и вообще для всех стадий моделирования при любой постановке вычислительных экспериментов и при какой бы то ни было организации проектной деятельности. Это одно из ключевых требований, реализация которого в системе поддержки обеспечивает автоматизацию сопоставления вариантов развития проектов, проверку критериев качества и других важных для вычислительного эксперимента свойств. Оперирование, предполагающее возвраты к ранее выполненным работам, естественно рассматривать в более общем плане. Дело в том, что развитая система поддержки моделирования должна поддерживать разработку многоверсионных приложений, в частности, для адаптации их под конкретные условия использования. Как следствие, информационное обеспечение корректных возвратов следует считать одним из средств подсистемы поддержки версионности. Эта подсистема в состоянии создать условия для существенного повышения производительности труда как разработчиков, так и пользователей.

Из приведенных сведений видно, что в дискретизации привносимая интеллектуальность занимает более существенное место, чем непосредственная, связанная с учетом особенностей, требующих адаптацию стандартных (или новых) алгоритмов построения сеток. При моделировании с использованием развитой системы поддержки этот этап оказывается более технологичным по сравнению с геометрическим и функциональным моделированием, поскольку он опирается на решения, принятые на предыдущем этапе, а применение инструментов привносит в проект интеллектуальность, предопределяющую заранее сформированные регламенты выполнения работ.

3.3. Аппроксимация исходных уравнений

Стадия аппроксимации, представленная в проекте соответствующим ей этапом построения вычислительной модели, завершает переход от исходной непрерывной задачи к ее дискретной постановке, итогом чего является сеточная система линейных или нелинейных алгебраических уравнений (СЛАУ или СНАУ), имеющих в большинстве случаев разреженные матрицы очень высоких порядков (до 10^8–10^{10}). Такие матрицы обычно представляются с помощью сжатых форматов, когда в целях экономии памяти хранятся только ненулевые элементы и необходимые индексные указатели. Подходов к аппроксимации существует достаточно много, в том числе методы конечных разностей, конечных объемов, конечных элементов и разрывные методы Галеркина различных порядков, а также коллокаций и спектральные методы (связанные с разложениями в ряды Фурье). Абстрактная *алгебраическая структура данных* модели (АСД), которая рассматривается как концептуальная основа

конкретной АСД, реализуемой в качестве цели аппроксимации, от этого не зависит. При всем значительном теоретическом различии ее алгоритмов у всех АСД имеется замечательное общее свойство: естественным образом распараллеливаемые поэлементные методы, основанные на вычислении локальных матриц и сборке глобальной матрицы задачи (см. [12, 16, 17]). Важно отметить, что применение алгоритмов высокой точности теоретически дает большое преимущество в эффективности, но приводит к громоздким многостраничным формулам, и их программная реализация представляет непростую техническую проблему.

По сравнению с предыдущими этапами свобода выбора при построении АСД ограничивается установленными ранее проектными решениями, в частности — свойствами уже построенной сетки. Интеллектуальность аппроксимации чаще всего привнесенная, связанная с методиками, разработанными в рамках соответствующих теоретических исследований. Основная творческая проблема этапа — выбор способа аппроксимации исходной задачи, обеспечивающего достаточно высокую точность и производительность расчетов. Надо сказать, что последний вопрос отнюдь не простой и имеет комплексный характер, поскольку схемы повышенного порядка приводят к более сложным и более плотным матричным структурам, что “удорожает” алгебраические методы решения.

Технологическая поддержка этапа заключается в проверке критериев контроля качества: порядок погрешности, возможная точность решения и скорость сходимости к нему, устойчивость к возмущениям исходных данных и к машинным округлениям и т.п. Не все полезные критерии могут быть определены путем анализа АСД. Некоторые из них могут потребовать проведения вычислительных экспериментов. Таким критерием является требование максимально полного использования ресурсов МВС на последующей стадии решения алгебраических уравнений. Вообще говоря, это проблемное требование в том смысле, что далеко не всегда для теоретически допустимой расчетной схемы можно доказать, что построенная АСД обеспечивает оптимально организованные параллельные вычисления на имеющихся ресурсах. Выбирая схему, приходится принимать ту или иную стратегию загрузки оборудования и сравнивать варианты на основе последующих расчетов, т.е. организовывать возврат к определению требуемой АСД, по предварительным результатам последующих этапов моделирования. Таким образом, проверка качества аппроксимации требует поддержки не только возвратов к предыдущим этапам, но и возобновления аппроксимации после выполнения последующих.

Стоит отметить взаимосвязанность аппроксимации и дискретизации. В ряде проектов построение ССД и АСД выполняется одновременно. По существу, в таких случаях реализуется следующая схема: делается априорный выбор алгоритма и расчетной схемы, которые итеративно уточняются. Однако это означает лишь то, выполнение двух этапов осуществляется совместно, т.е. завершение дискретизации не фиксируется как контрольная точка проекта.

Среди инструментов этапа аппроксимации особое место должны занимать средства автоматизированного определения особенностей генерируемой матрицы. За счет использования таких специализированных средств поддержки можно существенно повысить уровень корректности и производительности последующих расчетов и, в частности, избавить проект от более сложной проблемы собственных значений. Они также способствуют росту производительности труда разработчиков, избавляя их от рутинных процедур.

3.4. Решение алгебраических задач

Методически этап, связанный с решением систем линейных или нелинейных уравнений, можно рассматривать как определение неизвестных переменных АСД. Как правило, разработчикам проекта приходится иметь дело со сверхбольшими СЛАУ, решение которых является наиболее ресурсоемким в процессе моделирования, поскольку здесь объем вычислений растет нелинейно с увеличением числа степеней свободы. Именно с этим этапом связывается фиксация принимаемой стратегии параллелизма вычислений с целью максимально эффективного использования ресурсов МВС.

Данное направление является одним из бурно развиваемых разделов вычислительной математики. Наиболее высокопроизводительные подходы базируются на параллельных методах декомпозиции областей, а также на предобусловленных итерационных процессах в подпространствах Крылова (см. литературу в [13]). Для иллюстрации спектра алгоритмов, применяемых при решении СЛАУ, укажем на необходимость решения задач с разными типами матриц: вещественными и комплексными, квадратными и прямоугольными, эрмитовыми и неэрмитовыми, положительно определенными и знаконеопределенными. В этих методах приходится бороться не только с вычислительной сложностью, но и с огромными объемами пересылаемых данных, в силу чего различные проблемы вычислительной алгебры относят сейчас к направлению Data Intensive Computing.

Широкий арсенал подходов, потенциально применимых для решения систем уравнений, представляющий собой привнесенную в проект интеллектуальность, резко усложняется в связи с тем, что разработчики моделей имеют дело с реальными вычислительными ресурсами. Они вынуждены решать вопрос о применимости того или иного метода в условиях доступных МВС, учитывать перспективы их развития, заботиться о том, чтобы разрабатываемые системы моделирования были бы адаптируемы к различным конфигурациям оборудования с максимально полным и продуктивным использованием ресурсов. В связи с постоянным ростом сложности вычислительного оборудования и появлением новых архитектур адаптивность является “бесконечной” задачей: ее приходится решать постоянно на всем историческом пути развития моделирования. Впрочем, это общая проблема для вычислительного программирования в целом.

Для разработчиков конкретного проекта главной задачей этапа является выбор алгоритма и построение программ, реализующих зафиксированную на предыдущих этапах схему расчета с использованием библиотек системы поддержки моделирования и с учетом архитектур, принимаемых для вычислительного эксперимента. Этой задаче подчинена задача определения требований к вычислительным ресурсам, необходимых для проведения данного эксперимента. Сегодня разработчики имеют возможность применять при решении СЛАУ очень много различных библиотек, и каждая из них имеет свои преимущества и недостатки, с точки зрения функциональности или других особенностей конкретных условий. При решении о том, какие из них будут использоваться в конкретных условиях, чрезвычайно полезными являются эксперименты с многообразными типами СЛАУ, коллекции которых можно найти в Интернете.

Неоднозначность указанных задач свидетельствует о том, что их технологизация должна быть связана с поддержкой творческих процессов, избавленных от рутины и сопровождающихся контролем от принятия ошибочных решений. Поскольку основным требованием к этапу является организация максимально эффективных вычислений, необходимо, чтобы при использовании поддержки не возникали ситуации, когда разработчик не в состоянии отдать предпочтение той или иной стратегии локального выбора варианта из числа возможных. При проектировании такой поддержки нужно учитывать возможность настройки алгоритмов алгебраических библиотек на особенности задачи и оборудования, вопросы точности расчетов и выбора предобуславливателей для ускорения вычислений, другие аспекты реальных проектов.

Представленные соображения обосновывают утверждение о высокой интеллектуальности систем технологической поддержки для этапа решения алгебраических уравнений. При этом удельный вес привнесенной интеллектуальности — методы, алгоритмы и др. — выше той, которая появляется в проекте в связи с его спецификой, требующей творческого подхода к выбору метода решения. К такой специфике относятся особенности расчетной схемы и матрицы системы, а также ресурсы МВС, предоставляемые для модельного вычислительного эксперимента.

3.5. Постобработка

Изучение достаточно сложных явлений или процессов с помощью моделирования представляет собой длительную процедуру с активным

взаимодействием человека и компьютера. Это подразумевает необходимость поддержки, по крайней мере, двух типов работ, которые характеризуют содержание этапов разработки и применения моделей: предварительная обработка результатов расчета и последующее их предъявление для использования. Первый из них — это *постобработка*, или постпроцессинг, готовит полученную на предыдущих этапах информацию для второго, называемого *визуализацией*. Без постобработки и визуализации невозможен оперативный анализ и управление вычислительным процессом.

Целью этапа постобработки является приведение результатов моделирования после аппроксимации и решения уравнений, т.е. ССД и АСД с определенными в ходе расчетов данными, к виду, требуемому для анализа и принятия решений.

Конкретизация постобработки может быть различной, но обычно нужная структура для разработчиков модели известна. Так, в системах автоматизированного управления технологическим процессом должны быть выработаны управляющие сигналы, активизирующие действия моделируемого процесса, который нуждается в управлении. При высоком уровне автоматизации реальный пользователь модели может отсутствовать, т.е. не участвовать в принятии решения. Противоположная ситуация с моделированием для исследовательских и аналитических работ. Здесь нужны различные форматы визуального представления многомерных полей: изолинии, изоповерхности, изображение градиентных характеристик, графиков и др., с анимацией поведения или статические. Число полезных видов представлений может быть значительно, и, что весьма существенно, конкретные форматы определяются за пределами задачи разработки модели. Как следствие, целесообразно рассматривать постобработку как получение буферного представления, которое, во-первых, допускает необременительную для пользователя и достаточно простую для определения проверку критериев качества моделирования, а во-вторых, предлагает максимально полный комплект средств визуального предъявления информации о модели. При решении обратных задач, кроме того, необходимо обеспечить возможность автоматической организации расчетных итераций (см. следующий раздел).

Средства предъявления рассчитываемой информации можно охарактеризовать как возможность построения различных *конкретных структур данных результатов моделирования* (КСДР). Роль постобработки в этом построении — создание *абстрактной структуры данных результатов* (АСДР) в качестве базы реализации КСРД, нужных для разработчиков проекта или для пользователей. С учетом множества вариантов КСДР целесообразно стандартизовать формат АСДР в технологической поддержке постобработки. Важным требованием к стандарту является определение абстрактного управления оперированием результатами моделирования, для которого на этапе визуализации разрабатывается пользовательское интерфейсное представление. Это позволяет организовать технологическую поддержку в соответствии с хорошо себя зарекомендовавшим шаблоном (схемой) проектирования MVC (Model, View, Controller — модель, вид, управление) [18]: постобработка отвечает за формирование модели и управления, а визуализация — за вид предъявляемых пользователю средств. Следствием такого решения является повышение уровня привнесенной интеллектуальности разработки конкретных проектов.

Важно отметить технологическую связь постобработки с этапом дискретизации. Она возникает из-за требования согласования сеточных представлений модельных данных проекта. Итоговый результат постобработки АСДР является основой показа расчетной информации для пользователей. Но эта же структура может быть использована при визуализации данных для разработчиков проекта, оперативно изменяющих ССД для модернизации модели. Поэтому средства визуальной работы с АСДР целесообразно строить так, чтобы допускалась возможность использования той же инструментальной базы, что при дискретизации и аппроксимации. Иными словами, визуальное представление ССД и АСД в проекте следует рассматривать как варианты КСДР, приспособленные для оперирования разработчиков.

3.6. Организация итераций для оптимизационных расчетов

Самостоятельный важный блок процесса моделирования — решение обратных задач на основе оптимизационных принципов, заключающихся в условной минимизации заранее описываемого целевого функционала. Этот подход предполагает решение последовательности прямых задач с параметризованными данными. Для поиска нужных значений параметров в последние десятилетия разработано большое количество эффективных алгоритмов (методы внутренних точек, последовательного квадратичного программирования, доверительных интервалов и др.), которые сейчас активно внедряются в практику.

Организация итеративного расчетного процесса является ключевым условием при решении задач с обратными постановками. Но потребность возврата к предыдущим этапам, как правило, возникает и при решении прямых задач. Дело в том, что полученное решение по разным причинам может оказаться неудовлетворительным, и для получения полезных результатов нужна коррекция ранее выполненных работ, которая организуется как итеративный возврат к ранее пройденным этапам. Оптимизация алгоритмов решения прямых задач — это фактически обратная задача, связанная с поиском необходимых параметров использованного метода. Как было показано при обсуждении аппроксимации, такая итеративность организуется, например, при формировании АСД и расчетной схемы, максимально полно использующих ресурсы МВС (см. 3.3). Отличие двух видов итеративности в том, что, во-первых, они имеют различные критерии прекращения итераций, а во-вторых, — для обратных задач и проверка критерия, и возврат к предыдущим этапам автоматизируются, тогда как для прямых задач решение о возврате нуждается в специальном анализе, который указывает на необходимость коррекции ранее выполненных работ.

Технологическая поддержка итеративности обоих видов сводится к обычным и общим для систем программирования средствам управления версионности. Эти средства позволяют без труда вернуться к работе в состоянии, сложившемся ранее, а также обеспечивают сравнение и использование результатов версий программ и данных. Как уже было сказано, возможность такого режима работ должна быть заложена в проекте изначально. В этих условиях интеллектуальность поддержки итеративности оказывается общесистемной и привнесенной в проект. Специализированная часть такой поддержки требуется только для обратных задач в связи с автоматизацией возвратов к ранним этапам после вычисления целевого функционала.

3.7. Визуализация результатов расчета

Концепция разделения подготовки рассчитываемой информации к внешнему использованию на абстрактную и конкретную части позволяет значительно облегчить реализацию предъявления результатов вычислительного эксперимента пользователю. На нее можно посмотреть и с такой стороны: постобработка занимается извлечением необходимой информации (data mining), а визуализация — показом ее для анализа или для управления вычислительным процессом (data viewing). Показ данных и задание управляющих воздействий связаны, во-первых, с чисто графическими и интерфейсными средствами, а во-вторых — с передачей данных между МВС и рабочим местом пользователя результатов моделирования. Это две разные задачи. Решение первой из них — цель этапа визуализации. Что касается второй задачи, то ее следует рассматривать в более широком плане, поскольку связь МВС и рабочими местами нужно предусматривать не только для пользователей, но и для разработчиков модели. Более того, для пользователей достаточно лишь оперирование с КСДР, тогда как разработчикам необходим доступ ко всем модельным структурам данных, а также к сопутствующей информации в течение всего развития проекта. Они имеют дело с проектной документацией, с тестами и результатами тестирования, с версиями базовых структур и др. Многое из этого необязательно, а в большинстве случаев нежелательно или даже невозможно хранить на рабочих местах.

Показ данных, представленных в АСДР и опосредованных через КСДР, для пользователей зависит от различных потребностей. Будет ли для этой цели востребована КСДР разработчиков,

пусть даже урезанная до необходимого уровня, вопрос особый. Помимо факторов, непосредственно связанных с содержанием моделирования, потребности пользователей зависят от особенностей деятельности, в которой используется получаемая из модели информация. И именно это определяет требования к тому, какие средства нужны для оперирования модельной информацией и в каких интерфейсных формах они должны задаваться. Для максимально точного удовлетворения пользовательских потребностей в средствах оперирования модельной информацией целесообразно предлагать языки, учитывающие специфику области интереса пользователей — DSL (Domain Specific Languages, см. [19]). Такие языки определяются общей абстрактной моделью вычислений и специфичными для каждого круга пользователей конкретными представлениями этой модели. Применительно к вычислительному моделированию в качестве базы абстрактной модели выбирается АСДР, а основой конкретных представлений становятся соответствующие КСДР. В практике вычислительного программирования такой подход использован, например, в системе OpenFOAM [6]. Конструирование требуемых семейств DSL в проекте вычислительного моделирования характеризуется как специализированная интеллектуальность, а среда поддержки разработки их систем программирования, — как привнесенная интеллектуальность проекта.

3.8. Принятие решений

Построение модели всегда организуется, чтобы обеспечить пользователя — исследователя, инженера или другого специалиста в своей профессиональной сфере информацией для принятия решений. Требуемая информация образуется в модельных расчетах и используется в деятельности, связанной с прикладной областью. Принятие решений — деятельность, выполняемая не разработчиками модели, а потому ее следует считать постпроектной деятельностью, которая, строго говоря, этапом проекта не является. Тем не менее, некоторые из разработчиков проекта принимают в ней участие. Здесь их роль — не создание пользовательского продукта, а поддержка и сопровождение готовой модели, функция обучения пользователей с различным уровнем квалификации и исследовательской ориентации, а также обеспечение развития базовой системы поддержки моделирования. Эта сопутствующая деятельность может потребовать создание специальных версий программ, реализующих новые модели и алгоритмы с учетом когнитивных и онтологических аспектов складывающегося вычислительно-информационного окружения.

Рассмотрение аспектов постпроектной деятельности и ее интеллектуальности выходит за рамки настоящей работы. Однако ее очевидная связь с проектной задачей моделирования заслуживает особого внимания. Она не ограничивается потреблением рассчитываемых данных для выбора варианта решения из числа возможных и, прежде всего, заключается в совместной работе прикладных специалистов-пользователей с разработчиками модели. Главные задачи такой работы — валидация моделирования, т.е. проверка соответствия полученных результатов реалиям прикладной области, и анализ информации, извлекаемой из рассчитанных данных. Неадекватность результатов указывает на необходимость коррекции модели, соотношений ее параметров, требований к расчетам и пр. Последующий анализ определяет границы применимости полученных результатов для выбора решений. Если при построении модели прикладные специалисты выступают в качестве экспертов-консультантов для разработчиков, то в постпроектной деятельности экспертами становятся разработчики, консультирующие специалистов в том, как пользоваться текущими результатами моделирования, и оценивающими осуществимость реализации пожеланий развития модели и, а также его ресурсную потребность.

Совместная деятельность разработчиков и пользователей модели обуславливает целесообразность использования общих инструментов в ходе выработки решений на основе математического моделирования, которые представлены в комплекте базовых средств среды моделирования (применительно к БСМ такие инструменты обсуждаются в [8–12]). Общность технологической поддержки способствует повышению производительности труда и сокращению затрат

как при анализе, так и при выработке решений за счет привнесенной интеллектуализации соответствующих процессов.

4. ОБЗОР АРХИТЕКТУРНЫХ ТРЕБОВАНИЙ К БАЗОВОЙ СИСТЕМЕ МОДЕЛИРОВАНИЯ

Программная реализация рассмотренных выше методических стадий и их представление в требуемой технологической поддержке проектных этапов составляет *ядро* базовой системы моделирования БСМ, непосредственно отвечающей за функциональность конструируемых моделей и производительность труда разработчиков. Подчеркнем, что ядро БСМ состоит из практически *автономных блоков* — самодостаточных подсистем, взаимодействующих только через указанные ранее структуры данных, относящихся к процессу конструирования модели: ГСД и ФСД, ССД и АСД, а также к тем структурам, которые поддерживают эффективное использование результатов моделирования: АСДР и адаптированных для пользователей КСДР. Каждый блок и ядро в целом представляют собой интегрированную инструментальную среду, удовлетворяющую технологическим требованиям расширяемости состава моделей и алгоритмов, адаптируемости к архитектуре МВС, а также возможности включения и использования внешних программных продуктов. Блоки, в свою очередь, имеют библиотечную модульную структуру со стандартами внутренних интерфейсов, обеспечивающих сборку и контроль корректного выполнения соответствующих алгоритмических компонентов.

Последние функции заключаются фактически в операциях управления работой системы моделирования и составляют системное наполнение БСМ, назначением которого является общая технологическая поддержка версионности всего комплекса инструментов и разрабатываемых приложений, а также сборок необходимых конфигураций приложений для решения конкретных задач. Это архитектурное решение, называемое далее *версионной сборочностью*, работает по аналогии с детским конструктором LEGO, предлагающий комплект блоков, соединение которых позволяет строить разнообразные устойчивые конструкции.

Версионная сборочность должна служить преодолению извечного противоречия между универсальностью и эффективностью. Она естественным образом подразумевает избыточность функционального наполнения, когда для решения одной подзадачи в библиотеке имеется несколько алгоритмов, автоматически выбираемых по заложенным в конфигураторе признакам. Аналогичная вариантность позволяет поддерживать также многоязыковость модельных реализаций, а также адаптивность к различным операционным системам. Эти безусловно положительные качества БСМ могут создавать определенные трудности для разработчиков, если они не в полной мере владеют информацией о предлагаемых компонентах. Решение проблемы в том, чтобы среда показывала полный комплект средств, доступных для корректного оперирования, вместе с условиями допустимости их использования. Было бы ошибкой предоставлять инструменты, не указывая деятельность, в которой их целесообразно применять. А следствие этой ошибки — снижение уровня робастности приложений[2], создаваемых в рамках БСМ.

Для реализации версионной сборочности необходим специальный регламент, которому должны подчиняться компоненты среды технологической поддержки моделирования. Подобные регламенты были разработаны и оказались востребованными для различных областей системного программирования. Наиболее известный из них — технологический стандарт COM (Component Object Model, см. [20]), поддерживающий разработку взаимодействующих компонентов во многих продуктах компании Microsoft, в частности, в операционных системах семейства Windows. Разработку распределенных вычислений [21] поддерживает стандарт CORBA (Common Object Request Broker, см. [22]). Для вычислительного программирования предлагается стандарт CCA (Common Component Architecture, см. [23]), который включают поддержку массивного распараллеливания. К сожалению, эти и другие предложения не

[2]Повышение робастности — особая проблема разработки программных систем, которая заслуживает специального обсуждения. Некоторые подходы к ее решению можно найти в публикации [5].

в полной мере удовлетворяют требованиям поддержки эффективной реализации высокопроизводительных вычислений, а потому непосредственное их использование в базовой среде поддержки моделирования неадекватно. Подходы к преодолению этого недостатка предлагают разработчики инструментальной системы SPARSKIT [24], сведения о некоторых из них можно найти в работах [25, 26].

Будучи большой корпоративной разработкой, среда моделирования БСМ, очевидно, проектируется и создается в концепции открытых систем (Open Source), что в более широком плане соответствует идеологии открытых инноваций [27], в отличие от широко известных коммерческих продуктов ANSYS, NASTRAN и многих других (соответствующая информация легко доступна в Интернете). Разумеется, такая политика не противоречит созданию на основе общедоступных инструментариев закрытых пакетов для целевого использования.

Поскольку БСМ главным образом ориентирована на высокопроизводительное решение сверхбольших междисциплинарных задач, ее эксплуатация, естественно, целесообразна в рамках центров коллективного пользования (Data Center) на основе уже получивших признание облачных вычислений (Cloud Computing), с расширением спектра компьютерных услуг (Software as a Service, SaaS). И сама БСМ становится многопользовательской системой, что накладывает свою специфику на ее архитектуру. В частности, в соответствии с парадигмой IESP, ее разработка должна осуществляться при отсутствии формальных программных ограничений на число степеней задачи и на количество используемых процессоров и вычислительных узлов или ядер.

5. ЗАКЛЮЧЕНИЕ

Переходя теперь к главному вопросу настоящей статьи — интеллектуализации и автоматизации построения алгоритмов — хочется привести образный лозунг из [15]: переход от палео-информатики к нео-информатике, отражающий тенденции развития прикладного программного обеспечения вообще и, в частности, вычислительного моделирования, связанные с развитием требований к индустрии постановки сценариев и проведения вычислительных экспериментов. Приложения, которые можно отнести к нео-информатике, неизбежно связаны с многочисленными преобразованиями данных и разнообразием средств оперирования ими. Поэтому для оперативной реализации технологической поддержки их разработки фактически нужна система построения проблемно-ориентированных языков DSL, отражающая указанную множественность. Мы уже говорили о DSL как о методе предъявления средств для пользовательского оперирования модельной информацией. Но, как легко видеть, многообразие структур данных и их преобразований, с которыми приходится иметь дело, характерно для задач конструирования моделей. И очень заманчиво воспользоваться идеей DSL для поддержки деятельности разработчиков базовой системы моделирования. В связи с этим концепция реализации модели неизбежно меняется: требуется не просто программа, преобразующая входные данные в выходные, а так называемая мограмма (модельная программа) — термин, введенный А. Клеппе в [15] для обозначения программной системы, которая допускает многократные и многовариантные эксперименты с единой по своей сути моделью, использующие общее информационное обеспечение и необходимый комплект инструментов оперирования. В соответствии с введенными выше структурами данных, конструируемыми при разработке моделей, мограмма — это система структур ГСД, ФСД, ССД, АСД, а также АСДР и адаптированные для пользователей КСДР.

Серьезной проблемой перехода к нео-информатике является необходимость обеспечить максимально возможную приемлемость ранее разработанного программного обеспечения. Если для системного программирования она во многом решается за счет использования объектно-ориентированного проектирования и других современных решений, то для вычислительного программирования и для математического моделирования с их повышенными требованиями к эффективности расчетов, из-за чего приходится искать специальные подходы, поддерживающие многократное ис-

пользование существующих программ, в том числе написанных на Фортране во времена палео-информатики. Эти цели ставит перед собой проект поддержки так называемых научных вычислений SILD (Scientific Interface Definition Language), предлагая для этого многоязыковую систему Babel и соответствующий инструментарий, обеспечивающий возможность адаптации старого программного обеспечения к современным МВС [28]. К сожалению, несмотря на серьезные достижения данного направления (см., например, [29]) технологичного прогресса в решении проблемы переиспользования, удовлетворяющего повышенным требованиям к эффективности, пока достичь не удалось.

Проблема разработки проблемно ориентированного прикладного программного обеспечения и примеры ее решения известны много десятилетий (см., например, [30]), и в настоящее время широкое распространение имеют соответствующие многофункциональные системы MAPLE, MATLAB и другие. К сожалению, встраивание таких "монстров" в прикладные пакеты программ представляет нетривиальную, а чаще всего и просто неразрешимую техническую задачу. Поэтому в мировой практике получили распространение разработки небольших специализированных подсистем автоматической реализации специальных приложений, например, для аналитических выкладок на ЭВМ. В качестве иллюстрации, имеющей непосредственное отношение к математическому моделированию, укажем на реализованное в ППП HELMHOLTZ — 3D [31] построение векторных конечно-элементных базисных функций Неделека от 1-го до 4-го порядков включительно, на основе которых конструируются модули вычисления локальных матриц и постобработки электромагнитных полей. Достаточную серьезность и эффективность этого подхода подтверждает следующий пример: для громоздких формул, занимающих до 10 страниц текста, автоматические аналитические преобразования дают итоговый код на C++ объемом около 9 мегабайт. Важно подчеркнуть, что наличие такой подсистемы при дальнейшем развитии конечно-элементных инструментариев на порядки увеличивает производительность труда прикладного программиста.

Среди рассмотренных выше технологических вопросов имеется широкое поле деятельности для системных программистов. В числе открытых проблем можно назвать комплекс алгоритмов аналитической геометрии, реализацию декомпозиции сеточной области на сбалансированные подобласти с заданными размерами взаимопересечений, многочисленные приемы конструирования и оптимизации методов вычислительной алгебры и т.д., специальные инструменты для работы с графовыми структурами.

В заключение подчеркнем, что имеющаяся мировая тенденция перехода от простейших алгоритмов к наиболее эффективным ведет к значительному увеличению их логической сложности, и предотвратить торможение их внедрения можно только активными интеллектуальными инновациями, которые обеспечат реальную возможность использования в работе программистов интегрированных окружений, предлагающих наряду с инструментарием базовые модули-компоненты как строительные блоки. По своей сути это означает превращение ремесленнического подхода к моделированию в развитую комплексную расширяемую адаптивную технологически поддержанную деятельность.

СПИСОК ЛИТЕРАТУРЫ

1. IESP: www.exascale.org/iesp
2. *Knuth D.E.* Top Down Syntax Analysis. Acta Informatica. 1971. V. 1. № 2. P. 79–110.
3. *Кнут Д.* О переводе (трансляции) языков слева направо. В сб. "Языки и автоматы". М.: Мир, 1975. С. 9–42.
4. *Levine J., Mason T., Brown D.* lex&yacc 2nd Edition. O'RELLY®, October 1992.
5. *Skopin I.N.* An Approach to the Construction of Robust Systems of Interacting Processes. Chapter 9 in: Parallel Programming: Practical Aspects, Models and Current Limitations. Editor: M.S. Tarkov 2014.
6. OpenFOAM — The Open Source Computational Fluid Dynamics (CFD) Toolbox. www.openfoam.com
7. DUNE Numerics. www.dune-project.org.
8. *Ильин В.П.* Стратегии и тактики "заоблачного" математического моделирования. Труды Международной конференции ПАВТ'2014. Челябинск, изд. ЮУрГУ 2014. С. 99–107.

9. *Ильин В.П., Скопин И.Н.* Технологии вычислительного программирования // Программирование. 2011. № 4. С. 53–72.
10. *Голубева Л.А., Ильин В.П., Козырев А.Н.* О программных технологиях в геометрических аспектах математического моделирования //Вестник НГУ, серия "Информационные технологии". 2012. Т. 10. № 2. С. 25–33.
11. *Ильин В.П.* DELAUNAY: технологическая среда генерации сеток // СибЖИМ. 2013. Т. 16. № 2(54). С. 83–97.
12. *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. ЮУрГУ 2014. С. 42–50.
13. *Бутюгин Д.С., Гурьева Я.Л., Ильин В.П., Перевозкин Д.В., Петухов А.В., Скопин И.Н.* Функциональность и технологии алгебраических решателей в библиотеке Krylov. – Вестник ЮУрГУ. Серия "Вычислительная математика и информатика". 2013. Т. 2. № 3. С. 92–105.
14. Малюх В.Н. Введение в современные САПР. М., изд. ДМК, 2010.
15. SALOME — официальный сайт. http://www.salome-platform.org/.
16. *Ильин В.П.* Методы конечных разностей и конечных объемов для эллиптических уравнений. Новосибирск, изд. ИВМиМГ СО РАН, 2001, 318 с.
17. *Ильин В.П.* Методы и технологии конечных элементов. Новосибирск: изд. ИВМиМГ СО РАН, 2007, 370 с.
18. *Рогачев С.* Обобщённый Model-View-Controller. 2007. http://rsdn.ru/article/patterns/generic-mvc.xml.
19. *Kleppe A.* Software Language Engineering: Creating Domain–Specific Language Using Metamodels // N.Y. Addison–Wesley. 2008. 207 p.
20. *Оберг Р. Дж.* Технология COM+. Основы и программирование. М.: "Вильямс", 2000. С. 480.
21. *Тель. Ж.* Введение в распределенные алгоритмы. М.: МЦНМО, 2009. 616 с.
22. Official site CORBA (Common Object Request Broker). http://www.corba.org/.
23. CCA: The Common Component Architecture Forum. www.cca-forum.org/.
24. SPARSKIT. A basic tool-kit for sparse matrix computations (Version 2). http://www-users.cs.umn.edu/ saad/software/ /SPARSKIT/index.html.
25. *Malony A., Shende S., and others.* Performance Technology for Parallel and Distributed Component Software. Concurrency Computat.: Pract. Exper. 2003. http://people.cs.uchicago.edu/ntrebon/docs/ /gridperf02.pdf.
26. *Alexeev Yu., Allan B.A., and others.* Component-based software for high-performance scientific computing. http://iopscience.iop.org/1742-6596/16/1/073/pdf/ /1742-6596_16_1_073.pdf.
27. *Чесбро Г.* Открытые инновации. М., изд. "Поколение". 2007. 333 с.
28. Babel — High-Performance Language Interoperability (home page). https://computation.llnl.gov/casc/components/ /index.html#page=home.
29. *Prantl A., Imam Sh., Sarkar V.* Interfacing Chapel with traditional HPC programming languages // In PGAS'10 Fourth Conference on Partitioned Global Address Space Programming Model. New York, NY, USA, October 12–15, 2010. ACM New York, NY, USA © 2010.
30. *Ершов А.П., Ильин В.П.* Пакеты программ — технология решения прикладных задач // Препринт ВЦ СО АН СССР, № 121, Новосибирск, 1978.
31. *Butyugin D.S., Il'in V.P.* Solution of problems of harmonic electromagnetic field simulation in regularized and mixed formulations // Russian Journal of Numerical Analysis and Mathematical Modelling. 2014. V. 29. № 1. С. 1–12.