ИПМ им.М.В.Келдыша РАН

Абрау-2021 • Труды конференции

Л.В. Городняя

О неявной мультипарадигмальности параллельного программирования

Рекомендуемая форма библиографической ссылки

Городняя Л.В. О неявной мультипарадигмальности параллельного программирования // Научный сервис в сети Интернет: труды XXIII Всероссийской научной конференции (20-23 сентября 2021 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2021. — С. 104-116.
https://doi.org/10.20948/abrau-2021-6
https://keldysh.ru/abrau/2021/theses/6.pdf

Видеозапись выступления

Размещена также презентация к докладу

О неявной мультипарадигмальности параллельного программирования

Л.В. Городняя

Институт систем информатики СО РАН

Аннотация. Доклад посвящен результатам парадигмального анализа проблем, средств и методов организации параллельных вычислений и многопоточных программ для многопроцессорных комплексов и распределённых систем. Парадигмальный анализ языков и систем программирования позволяет снижать сложность решаемых задач методами декомпозиции программ на автономно развиваемые компоненты, оценивать их сходство и различия, учёт которых необходим при прогнозировании хода процессов применения, а также при планировании изучения и организации разработки программ. Показано разнообразие парадигмальных характеристик, присущих подготовке и отладке долгоживущих программ параллельных вычислений. Приведена краткая схема мультпарадигмального языка параллельного программирования учебного назначения.

Ключевые слова: парадигмальная декомпозиция, преподавание системного программирования, параллельные вычисления, языки программирования

On implicit multiparadigmality of parallel programming

LV. Gorodnyaya

A.P. Ershov Institute of Informatics Systems (IIS), SB RAS

Abstract. The report is devoted to the results of a paradigmatic analysis of problems, means and methods of organizing parallel computing and multi-threaded programs for multiprocessor complexes and distributed systems. Paradigmatic analysis of programming language and systems allows decomposing the complexity of the tasks being solved into autonomously developed components, assessing their similarities and differences, which must be taken into account when predicting the course of application processes, as well as when planning the study and organizing the development of programs. A variety of paradigmatic characteristics inherent in the preparation and

debugging of long-lived parallel computing programs are shown. A sketch of a multi-paradigm parallel programming language for educational purposes is presented.

Keywords: paradigm decomposition, teaching systems programming, parallel computing, programming languages

Рассматривая результаты парадигмального анализа проблем, средств и методов организации параллельных вычислений, а также подготовки многопоточных программ для многопроцессорных комплексов и распределённых систем, можно обратить внимание на явное разнообразие постановок задач и соответствующих им приоритетов в принятии решений на разных этапах разработки и отладки программ. Не противореча многочисленным описаниям парадигм программирования, парадигмальная методика сравнения языков программирования (ЯП), представленная в работах [2-6], опирается на приведённое в статье [11] неформальное определение термина «парадигмы программирования», констатирующее, что при сравнении парадигм следует выделять различающие их признаки, допускающие проверку. Здесь в качестве таких признаков использованы приоритеты в принятии решений на разных этапах разработки и отладки программ. В результате эта методика позволяет сводить постановки решаемых задач к комплектам автономно развиваемых подзадач, оценивать их сходство и различия, учёт которых необходим при прогнозировании сложности процессов применения программируемых решений, начиная с планирования, изучения и организации разработки программ решения таких подзадач для долгоживущих программ. Изложение начинается с парадигмальной характеристики постановок задач, вывода оценки трудоёмкости параллельного программирования на основе учёта степени изученности задач параллельных вычислений и особенностей уровня квалификации специалистов, выпускаемых образовательной системой. В заключении описаны требования к языковой поддержке параллельного программирования.

1. Парадигмальная характеристика

Используя систематизацию парадигм программирования на основе различий в приоритетах принятия программируемых решений, можно сделать вывод, что одной из причин сложности разработки программ параллельных вычислений является их скрытая мультипарадигмальность [3]. Для разработки параллельной программы многое требуется продумывать по разному одновременно в различных парадигмах, удобных

для решения отдельных подзадач без возможности решения полного комплекса подзадач в единой обстановке, Наиболее очевидно парадигмальные различия видны при решении задач масштабирования вычислений на различные многопроцессорные комплексы, синхронизации взаимодействий над локальной и общей памятью в многопоточных программах, представлении естественного асинхронного параллелизма уровня постановок задач и достижения высокой производительности программ с учётом критерия полноты загрузки доступных многопроцессорных комплексов или распределённых систем. Отдельно имеет место образовательная проблема — ознакомление с феноменами параллелизма, дающее специалистам интуитивное понимание методов решения особо трудных задач. Создано заметное число языков и систем программирования (ЯиСП), позволяющих решать отдельные из этих задач в рамках парадигм, поддержка которых представлена в разных языках программирования. (Таблица 1).

Таблица 1.

№	*Проблема*	*Парадигма*	*ЯП и API*
1	Масштабирование	Многопроцессорное программирование	VHDL, XC, СИГМА, bash, Occam, mpC, Эль-76, Limbo, Kotlin, MPI
2	Синхронизация потоков	Синхронное (синхронизующее) программирование	APL, VAL, Sisal, Alef, E, X10, LuNA, Charm, Go, Java, Scala, Rust, Пифагор, OpenMP
3	Постановки задач	Асинхронное программирование	БАРС, Haskell, Erlang, JavaScript, Python, C#
4	Производитель-ность программ	Высокопроизводитель-ное программирование	Setl, HPF, G, Sparkel, mpC, Sanscript, D, Rest, F#
5	Ознакомление с параллелизмом	Учебное программирование	Logo, Робик, Karel, OZ, A++, СИНХРО, Lsl

Таким образом, для каждой из трудно решаемых задач параллельных вычислений уже сформирована отдельная удобная парадигма её решения и создан ряд языков программирования, поддерживающих такую парадигму. Различие между парадигмами проявляется в упорядочении важности средств и методов, используемых при решении отдельных задач, другие задачи требуют иного упорядочения. В каждый момент разработки программы обычно используется одна парадигма. Соответственно и в разных языках программирования выделяется одна ведущая парадигма. Требования к решению достаточно сложных задач параллельных вычислений связано с целым рядом трудных задач, что влечёт необходимость использования разных парадигм на разных этапах их создания и фазах их жизни. При переходе к технологии параллельного программирования важна гарантия получения практичного результатов,

что требует поддержки полного спектра парадигм, используемых на разных этапах разработки программ. Трудоёмкость использования разных парадигм при решении одной задачи обычно минимизируется созданием многоязыковых систем, допускающих по мере необходимости возможность перехода от одной парадигмы к другой без затрат на освоение разных интерфейсов, что приводит к целесообразности создания мультипарадигмального языка параллельных вычислений, поддерживающего одновременно все основные парадигмы параллелизма.

2. Парадигмальная декомпозиция

Как показывает опыт языков БАРС и Haskell, в таких случаях удобно декомпозировать определение ЯП на отдельные подъязыки, поддерживающие основные парадигмы или монады, нацеленные на конкретные модели подготовки и представления программ, так чтобы на каждом этапе разработки программы локализовать использование одной парадигмы, характеризуемой сравнительно небольшим набором средств и методов в рамках одного способа мышления. Каждая парадигма имеет своё наполнение категорий семантических систем и своё упорядочение их роли в процессе программирования [3].

Средства многопроцессорного программирования обычно опираются на данные и характеристики доступной архитектуры, включая основной многопроцессорный комплекс и взаимосвязи между его элементами. Оперирование комплексом позволяет инициировать процессы функционирования отдельных процессоров, их блокировку, возобновление и отмену процессов. По ходу процессов возможны обмены данными по определённым протоколам, результаты которых можно рассматривать как цель программы. Принятие решений начинается с определения пространства возможных многопроцессорных комплексов, что можно рассматривать как особую разновидность памяти со своей дисциплиной функционирования и взаимодействия элементов. Далее происходит выбор подходящих конфигураций и структурирования пространства итерирования процессов, предназначенных для выполнения на отдельных процессорах. Затем полученная схема управления процессами наполняется собственно действиями, выполняющими вычисления. Обычно предпочитаются процедуры без рекурсии, освобожденные от других сложностей управления вычислениями и побочных эффектов над общей памятью. Строится многопроцессорная программа, допускающая в динамике реконфигурацию многопроцессорного комплекса[1].

1. [1]https://mooc.tsu.ru/mooc-openedu/mpi/ Курс «Параллельное программирование с использованием OpenMP и MPI»

При синхронном (синхронизующем) многопоточном программировании выделяются достаточно чёткие схемы управления вычислениями и разделяются регулярные участки и типичные модели программы, удобные для распараллеливания. Обычно выделяются фрагменты, свободные от побочных эффектов в памяти, и допускается неимперативное управление временем исполнения потоков программы с учётом иерархии схемы управления программой и некоторых временных отношений. Принятие решений начинается с выбора стандартных схем управления, используется понятие «пространство итерирования», которое можно структурировать в зависимости от распределения данных и методов их хранения, что может влиять на эффективность и дисциплину обработки многоуровневой памяти. Управление итерированием может использовать произвольные предикаты. Схема управления над пространством итерирования потоков наполняется сравнительно простыми для отладки фрагментами, возможно отлаженными заранее или программируемыми автономно. Для вывода результатов программы выделяется специальные средства формирования результатов вычислений, полученных на равноправных потоках многопоточной программы. Таким образом получается многопоточная программа с динамически изменяемым пространством потоков над локальной памятью с возможностью эпизодической синхронизации их отдельных фрагментов[2].

Асинхронное программирование нацелено на предельное выражение независимых элементов программы, обусловленных природой решаемой задачи, что может быть базой для максимального распараллеливания при условии представления специальных схем организации вычислений с учётом специфики доступного оборудования. Начинается принятие решений с выбора схем управления действиями и представления условий их срабатывания. Действия могут использовать ту или иную дисциплину обработки памяти. Поддерживается неявное и программируемое разнообразие дисциплин доступа к памяти, включая иерархию неоднородный памяти, и схем вычислений — фрагментов, наполняющих общую схему программы процедурами или библиотечными модулями. Получается схема программируемых синхросетей, асинхронно управляющая выполнением действий в зависимости от условий их готовности к выполнению.

Для высокопроизводительного программирования необходим переход от отдельного прогона программы к учёту перспектив её многократного применения и улучшения. Появляется возможность использовать недогруженные мощности многопроцессорных комплексов выполнением фрагментов программы в расчёте на предстоящие в будущем прогоны при данных, возможно востребованных в предстоящих

[2] https://habr.com/ru/post/121925/ Курс «Основы MPI»

прецедентах её отладки и применения. Примерно так организуют программы, предназначенные для сверх быстрого реагирования на опасные события, например, при прогнозировании цунами. Принятие решений начинается со схем и моделей вычислений, возможно над общей памятью, но с приоритетом локальной памяти. Управление вычислениями учитывает особенности многократного выполнения программы при её отладке и применении, включая возможность наследования результатов между сеансами и сравнения измеримых характеристик производительности версий программы. Получается ряд улучшаемых версий программы, выбор одной из которых может учитывать особенности текущих условий применения, включая конфигурацию многопроцессорного комплекса и требования к измеримым характеристикам производительности программ.

Учебное программирование призвано компенсировать недостатки общепринятых образовательных ориентиров на безошибочность, единственность, последовательность, императивность и однозначность решений при решении любых задач. Такие ориентиры затрудняют обучение методам программирования вообще, и параллельных вычислений ещё более. Учебное программирование может давать старт ко всем необходимым парадигмам параллельных вычислений на базе опыта применения и конструирования игровых программ типа визуализации роботов. Это достаточная причина для мультипарадигмальности учебных языков программирования, обычно поддерживающих некоторые средства представления параллельных вычислений [6, 7].

Долгоживущие ЯП, как и новые ЯП нашего века, обычно мультипарадигмальны. Есть основания для заключения, что успешная практика параллельного программирования требует мультипарадигмальной поддержки полного спектра парадигм параллельных вычислений, допускающей их развитие, пополнение и применение по мере необходимости с возможностью перехода к очередной парадигме без изменения общеязковой и системной обстановки.

3. Трудоёмкость

Зависимость трудоёмкости и качества программирования от квалификационного уровня программистов исследуется с первых шагов формирования программирования как профессии. По свидетельству Дж.Вейнберга, автора уникальной монографии «Психология программирования» [10], фирма ИБМ в начале 1970-х провела исследования разброса трудоёмкости программирования в зависимости от опыта, возраста и способностей. Оказалось, что на простых задачах разброс составляет 1 к 28, причём почти независимо от опыта и возраста. На сложных задачах заметна положительная зависимость от способностей, опыта и возраста с несколько меньшим разбросом, примерно 1 к 10.

Причём, скорость выполнения заданий редко способствует качеству решений, чаще наоборот, скороспелые решения при программировании приводят к много большим трудозатратам при отладке и эксплуатации программ. Не меньший разброс был замечен и в оценке качества программируемых решений, не выявлено зависимости производительности программ от трудоёмкости программирования. Эти вывод дополняют исследования В.Л. Авербуха зависимости производительности труда в сфере высокопроизводительных вычислений от возрастного ценза. Вывод оказался в пользу старшей возрастной категории — специалистов предпенсионного и пенсионного возраста, способных не только использовать свой богатый опыт, но и критически воспринимать модные новинки, находить практичные композиции классики и модерна [1].

Следует отметить, что жаргон современного практического программирования использует понятие «язык программирования» как «входной язык – расширенное подмножество ЯП типовой системы программирования (СП), функционирующей на базе определённой конфигурации оборудования». Разница заключается в том, что СП обычно сопровождает реализацию ЯП расширяемым комплектом библиотечных модулей. В результате происходит сглаживание видимых на практике различий между ЯиСП. Кроме того, прямые измерения трудоёмкости программирования и производительности программ почти не отражают зависимости результата от принятых программистом решений и выбора конструкций ЯП. Хотя программируемые решения представляются в терминах ЯП, их влияние растворяется в весьма сложном комплексе, наследующем производительность СП и оборудования с большим доминированием характеристик элементной базы и аппаратуры. Таким образом, существует проблема создания методики, позволяющей выявлять такие зависимости совмещением прямых измерений с результатами экспертных оценок особенностей ЯиСП, возможно отличающихся от оценок ЯП [8].

Есть основания при прогнозировании трудоёмкости параллельного программирования учитывать не только степень изученности решаемых задач, но ещё и уровень квалификации и способностей разработчиков программы решения задачи, умеющих преодолевать понятийную сложность реализуемых и используемых программируемых и программных средств, особенности функционирования которых могут выходить за пределы привычных представлений. Для параллельных вычислений степень изученности часто наследует результат ранее созданной последовательной программы решения задачи, обладающей математически точной постановкой. Возникает соблазн, тормозящий осознание или создание более эффективного параллельного алгоритма. Кроме того, уровень квалификации специалистов опирается на опыт императивно-процедурного программирования, препятствующего

восприятию более сложных зависимостей в параллельных процессах. Понятийная сложность решаемых задач выходит за пределы обычного образования и сами понятия обретают более широкое толкование, отчасти противоречащее привычному интуитивному пониманию. Представление результатов оценки понятийной сложности, позволяющее структурировать пространство таких параметров, рассмотрено в статье [2].

4. Степень изученности

По степени изученности существенно различаются следующие категории постановок задач, влияющие на выбор методов решения задач и трудоёмкость их программирования:

- новые;
- исследовательские;
- практичные;
- точные.

Для новых постановок задач характерно отсутствие доступного прецедента практичного решения задачи, новизна используемых средств или недостаток опыта исполнителей. Задачи параллельных вычислений поставлены ещё в докомпьютерную эпоху и постановки многих таких задач обладают математической точностью. Тем не менее часть задач параллельного программирования их решений приходится рассматривать как новые из-за стремительного обновления ИТ, элементной базы и нерешённых образовательных проблем программирования в целом. Любая проблема, не получившая хорошего решения, остаётся в статусе новой задачи независимо от времени её постановки. Исследовательские постановки задач параллельного программирования в настоящее время следуют тенденции функционального подхода и изучения схем структурирования пространства итерирования процессов, позволяющего оптимизировать обработку памяти. Функциональный подход можно рассматривать как методику сведения решений сложных задач к композициям из планарных проекций, достаточно удобных для понимания, анализа и обработки. Практичные постановки математических задач параллельных вычислений, нацеленные на актуальность и удобство применения, преимущественно используют мощность доступных ИТ для воспроизведения математических моделей, ранее ограниченных низкой эффективностью оборудования, а теперь получивших перспективу стать новой информационной революцией. Точные постановки большинства задач параллельного программирования сложились на базе последовательных алгоритмов и включают в себя испытание возможностей используемых средств, связанных с мерой организованности ранее созданной императивной программы. Некоторые сложности связаны с

использованием однопроцессорных конфигураций как исходной модели параллельных многопоточных программ. Не исключено, что предельным случаем удобнее рассматривать двухпроцессорные конфигурации. Появление собственно параллельных алгоритмов встречается не так уж часто, хотя в середине 1990-ых годов проводились конференции такого направления.

5. Образование и квалификация

Обучение параллельным вычислениям входит в образовательные программы многих университетов, что достаточно для понимания их сложности и постановки задач их исследования. Проблемой является переход к практике реализации высокопроизводительных программ, удовлетворяющих особо сложным, трудно удостоверяемым критериям надёжности и безопасности. Разница в содержательной сложности трудовых функций, согласно профессиональным стандартам «Программист» и «Системный программист», сосредоточена на требованиях, нацеленных на понимание смысла решаемых задач, развитие области приложения решений, определение границы применимости полученных результатов и достижение системности при обобщении созданных инструментальных средств, важнейших для решения задач современных ИТ [9]. Характерной чертой системного подхода как ведущего метода программирования является переход к классам задач при содержательном анализе постановок задач. Границы класса устанавливаются выбором процесса решения задач. Переход к экспериментам на суперкомпьютерах показал, что именно системные решения могут дать весомый вклад в производительность параллельных вычислений, причём такой вклад может превышать теоретические прогнозы. Это можно рассматривать как обоснование необходимости более фундаментального подхода к программированию, особенно к системному программированию и его математическим основам, дающим путь к созданию высокопроизводительных программ [4].

При создании, формировании и исследовании математических моделей как фундаментального базиса для решения особо трудных проблем эффективности, надёжности и безопасности программного обеспечения важную роль играет развитие моделей, связанных со временем и ресурсами, слабо представленных в курсах по классической математике.

Квалификация системного программиста включает в себя формирование способностей самостоятельно изобретать решения новых задач и навыки ответственно повышать качество готовых решений, наряду с глубоким владением неклассической и фундаментальной математикой и ознакомлением с современной физикой, психологией и лингвистикой, что

слишком неудобно образовательной системе, нацеленной на формальный контроль усвоения устойчивых стереотипов, представленных в учебно-методической литературе, противоречит учебно-методическим традициям.

Экстенсивное развитие ИТ заметно опережает возможности человека оперативно осваивать новые возможности аппаратуры и системных средств ИТ, выходящие за пределы пользовательского уровня, поддерживаемого поставщиками инструментария и программных продуктов.

Миссия системного программирования – создание инструментария, позволяющего повышать качество информационных систем, включая поиск новых решений по обеспечению надежности и безопасности информационных технологий [5].

6. Заключение

Таким образом, возможен подход, позволяющий учитывать особенности решаемых задач и парадигм программирования, необходимых для решения проблем параллельных вычислений, влияющих на выбор методов их решения, в зависимости от приоритетов в выборе языковых средств и реализационных конструкций. Можно наметить линию, позволяющую сравнивать языки, выделяя сопоставимые примеры программ, и анализировать результаты прямых измерений производительности программ, выделяя особенности базовых средств и реализационных решений в системах программирования, нацеленных на улучшения создаваемых программных продуктов. Программирование давно уже стало массовой профессией, требующей объективных метрик для оценки качества программируемых решений. Парадигмальные ошибки, обнаруживаемые при эксплуатации привычных систем программирования на современном многопроцессорном оборудовании, показывают, что часть из них были просто незаметны до появления сетей, мобильных устройств и суперкомпьютеров.

Многие вопросы пока не получили практичного ответа. Появление новых парадигм можно связать с проявлением круга новых задач, решение которых вызывает трудности. Не ясно, насколько целесообразно взаимодействие парадигм. Остаются в стороне образовательные проблемы овладения новыми парадигмами. Кроме того, особенности парадигм лишь отчасти выражаются на уровне представления программы, часть являются требованиями к прагматике системно-реализационной поддержки в ЯиСП. Граница между семантикой программируемых решений и прагматикой их системной поддержки у каждой парадигмы своя.

В ИСИ СО РАН традиционно ведутся работы по созданию языков программирования учебного назначения, включая ознакомление с феноменами параллелизма. В настоящее время разрабатывается

мультипарадигмальный язык МетаСупер, включающий в себя подъязыки для поддержки основных парадигм параллельного программирования [6].

АМК — подъязык низкого уровня для представления многопроцессорных программ, допускающих использование общей памяти (наследует bash, SECD, pi-код, JVM, СИГМА).

СинПар — подъязык высокого уровня для представления синхронизуемых многопоточных программ с приоритетом использования локальной памяти без побочных эффектов (наследует APL, Sisal, MPI).

Асинхр — подъязык сверх высокого уровня для представления естественного параллелизма, допускающего различные схемы управления и фрагменты при определении и спецификации программируемых процессов (наследует Setl, БАРС, OpenMP, Haskell, F#).

Синхро — подъязык низкого уровня для представления и конструирования учебных тренажёров при ознакомлении с феноменами параллелизма и обучении системному программированию (наследует Logo, Робик, HFP).

Трансформ — подъязык высокого уровня, обогащённый средствами метапрограммирования, для представления, отладки, оптимизации и преобразования программ, а также измерения характеристик производительности улучшаемых программ (наследует БНФ, Рефал, YACC, LEX, mpC, Clang-LLVM).

Литература

1. Авербух В.Л. Визуализация программного обеспечения / Екатеринбург, ИММ УрО РАН. -1995. - 168 с.
2. Городняя Л.В. О представлении результатов анализа языков и систем программирования. Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17-22 сентября 2018 г., г. Новороссийск). — М.: ИПМ им. М. В. Келдыша, 2018 - с. 262-277. URL: https://doi.org/10.20948/abrau-2019-03
3. Городняя Л.В. Систематизации парадигм программирования по приоритетам принятия решений /Электронные библиотеки. Том 23 № 4 (2020) Часть 2 - с. 666-696 DOI: https://doi.org/10.26907/1562-5419-2020-23-4-666-696
4. Городняя Л.В. Подход к оценке трудоёмкости программирования /Нучный сервис в сети Интернет, сентябрь 2020 г., Новороссийск (Абрау-Москва) https://keldysh.ru/abrau/2020/temp/3.pdf
5. Городняя Л.В. Перспективно стратегические парадигмы программирования Академика Андрея Петровича Ершов. 5-я международная конференция «Развитие вычислительной техники в России, странах бывшего СССР и СЭВ (SORUCOM 2020)» — 6–8 октября 2020 г., Москва. - с. 83-97

6. Городняя Л.В. Учебный язык параллельного программирования СИНХРО – с. 92-97. Языки программирования и компиляторы — 2017 труды конференции / Южный федеральный университет ;под ред. Д. В. Дуброва . — Ростов-на-Дону: Издательство Южного федерального университета, 2017. — 282 с. ISBN 978-5-9275-2349-8 http://plc.sfedu.ru/files/PLC-2017-proceedings.pdf
7. Шилов Н.В., Городняя Л.В., Марчук А.Г. Параллельное программирование среди других парадигм программирования. Научно-практический журнал «Прикладная информатика», ISSN 1993-8314, М., №1 (31) 2011, - с.120-129. http://agora.guru.ru/abrau2011/pdf/193.pdf
8. Городняя Л.В., Демидов С.Е., Кириченко М.Д., Ткаченко Д.Д. Проект информационного стенда "ПРИЗМА" для представления измеримых характеристик языков и систем программирования. XXV Байкальская Всероссийская конференция международным участием — июль 2020 г., Иркутск. Информационные и математические технологии в науке и управлении. Издательство: Институт систем энергетики им. Л.А. Мелентьева СО РАН (Иркутск) ISSN: 2413-0133
9. Стандарты ФГОС — //fgosvo.ru/docs/101/69/2/6/
10. Weinberg G.M. The Psychology of Computer Programming. – New York: Van Norstand Reinhold Comp., 1971. 279 с.
11. Peter Wegner. 1990. Concepts and paradigms of object-oriented programming. SIGPLAN OOPS Mess. 1, 1 (August 1990), p. 7-87. https://pdfs.semanticscholar. DOI: http://dx.doi.org/10.1145/ .

References

1. Averbukh V.L. Software visualization. / Yekaterinburg IMM UrO RAS - 1995. — 168 p.
2. Gorodnyaya L. V. On the presentation of the results of the languages and programming systems analysis. Scientific service on the Internet: proceedings of the XX All-Russian scientific conference (17-22 September 2018, Novorossiysk). — Moscow: IPM im. M. V. Keldysh, 2018.
3. Gorodnyaya L.V. Systematization of programming paradigms by decision-making priorities / Russian Digital Libraries Journal. Volume 23 No. 4 (2020). Part 2 p. 666-696 DOI: https://doi.org/10.26907/1562-5419-2020-23-4-666-696
4. Gorodnyaya L.V. An approach to assessing the complexity of programming / Scientific service on the Internet, September 2020, Novorossiysk (Abrau-Moscow) https://keldysh.ru/abrau/2020/temp/3.pdf

5. Gorodnyaya L.V. Prospectively strategic programming paradigms of Academician Andrey Petrovich Ershov. 5th International Conference "Development of Computing Technology in Russia, the Countries of the Former USSR and CMEA (SORUCOM 2020)" — October 6-8, 2020, Moscow - p. 83-97
6. Gorodnyaya L.V. Educational language for parallel programming SYNCHRO — p. 92-97. Programming languages and compilers — 2017 conference proceedings / Southern Federal University; ed. D. V. Dubrova. — Rostov-on-Don: Southern Federal University Publishing House, 2017 .— 282 p. ISBN 978-5-9275-2349-8 http://plc.sfedu.ru/files/PLC-2017-proceedings.pdf
7. Shilov N.V., Gorodnyaya L.V., Marchuk A.G. Parallel programming among other programming paradigms. Scientific and practical journal "Applied Informatics", ISSN 1993-8314, M., No. 1 (31) 2011, - p. 120-129. http://agora.guru.ru/abrau2011/pdf/193.pdf
8. Gorodnyaya L.V., Demidov S.E., Kirichenko M.D., Tkachenko D.D. Project of the information stand "PRISMA" for the presentation of measurable characteristics of languages and programming systems. XXV Baikal All-Russian Conference with International Participation — July 2020, Irkutsk. Information and Mathematical Technologies in Science and Management. Publisher: Institute of Energy Systems. L.A. Melent'ev SB RAS (Irkutsk) ISSN: 2413-0133
9. FGOS standards — //fgosvo.ru/docs/101/69/2/6/
10. Weinberg G.M. The Psychology of Computer Programming. – New York: Van Norstand Reinhold Comp., 1971. - 279 p.
11. Peter Wegner. 1990. Concepts and paradigms of object-oriented programming. SIGPLAN OOPS Mess. 1, 1 (August 1990), p. 7-87. https://pdfs.semanticscholar. DOI: http://dx.doi.org/10.1145/ .