Formalization of test processes of the spacecraft onboard equipment

Rodion Vyacheslavovich Vogorovskiy, *engineer*

Lyudmila Fedorovna Nozhenkova*, Ph. D., Professor, the head. Department of applied Informatics*

Institute of computational modeling SB RAS, Siberian Federal University, Krasnoyarsk

In this paper, the problem of formalization of test process of the spacecraft onboard equipment is observed. The representation principles of test processes as formal scripts are described. The cases of using scripts at all stages of test process are presented.

Keywords: spacecraft, command and measuring system, test execution support, scripts of testing.

УДК 004.43 (042.4)

О КЛАССИФИКАЦИЯХ ПАРАДИГМ ПРОГРАММИРОВАНИЯ И ПАРАЛЛЕЛЬНОМ ПРОГРАММИРОВАНИИ

Лидия Васильевна Городняя*, к.ф.-м.н., доцент, старший научный сотрудник*
Тел.: 8 383 330 6470, e-mail: lidvas@gmail.com
Новосибирский национальный исследовательский государственный университет
www.nsu.ru
Институт систем информатики им. А.П. Ершова СО РАН
www.iis.nsk.su

Работа нацелена на привлечение внимания к проблеме изучения парадигм программирования, актуальность которой связана с расширением множества языков программирования и массовым распространением распределённых информационных систем, использующих взаимодействующие параллельные процессы.

Ключевые слова: языки и системы программирования, парадигмы программирования, методы определения компьютерных языков, параллельные алгоритмы, учебное программирование.

Актуальность изучения и систематизации парадигм программирования связана с практическими задачами обоснования при выборе подходов к обеспечению производительности, надежности и эффективности сложных информационных систем, конструируемых с использованием разносортных информационных технологий и сервисов, применяемых в разных условиях и требующих учёта многообразных явлений, связанных с поддержкой параллельных процессов на базе многопроцессорных конфигураций [1].

Л.В. Городняя

Парадигмами программирования в форме языков и систем программирования представлено знание о потенциале надёжных и безопасных средств и методов создания информационных систем. Теоретически различие парадигм программирования может быть выражено на уровне операционной семантики, представляющей детали структур памяти, механизмы выполнения укрупнённых действий, принципы компиляции и верификации программ [2; 3; 4].

Суть авторской работы заключается в анализе парадигматической классификации систем программирования, допускающей лаконичную характеристику их практически значимых особенностей, определяющих трудоёмкость программирования и живучесть разрабатываемых программ с акцентом на проблемы параллельного программирования [5; 6; 7]. Для выработки практических рекомендаций такого рода требуется чёткая

формулировка различий на всех уровнях определения языков программирования, что и рассматривается в данной статье.

Разнообразие парадигм программирования

Альтернативные подходы к обработке информации, сложившиеся при создании и применении языков и систем программирования, принято называть парадигмами программирования. Независимо от области приложения парадигмы могут отличаться уровнем абстрагирования от возможностей аппаратуры, степенью изученности постановок задач, мерой организованности и рангом работоспособности программ решения задач. При определении парадигм обычно поляризуются следующие характеристики языков и систем программирования:

– программируемые решения представляются в императивно-процедурной или в декларативной форме;

– обрабатываемые элементы данных позиционируются как адресуемые блоки памяти или независимо размещаемые значения;

– программа может быть защищена от изменений в процессе её выполнения или допускать программируемые модификации по ходу получения результатов вычисления;

– программа может быть целостной или собираться из типовых компонент и шаблонов;

– представленные в программе функции могут быть частичными, типизированными, обрабатывающими значения заданного домена, или универсальными, дающими разумную реакцию на любой элемент данных;

– управление вычислениями выполняется последовательно или параллельно;

– порядок действий может быть определённым или недетерминированным;

– вычисления могут быть энергичными или ленивыми;

– области видимости имён могут быть глобальными или локализованными по иерархии конструкций с возможностью восстановления контекста;

– распределение и повторное использование памяти может быть действием в программе или выполняться автоматически системой программирования;

– инициирование памяти первоначально размещаемыми значениями может требовать программируемых действий или выполняться в системе программирования по умолчанию;

– домены значений могут быть независимыми или допускать пересечения;

– результат выполнения программы может быть рассредоточен по ряду переменных или сконцентрирован в специальном регистре;

– контроль правильности может выполняться статически, при анализе текста программы, или динамически, при выполнении кода программы.

Выбор между так противопоставленными характеристиками в тексте программы выражается синтаксически, нередко с помощью спецификаторов языка программирования. Определение операционной семантики языка программирования обычно использует понятие абстрактной машины, которая в случае многопроцессорных конфигураций естественно становится конструкцией из абстрактных процессоров – абстрактным комплексом, что позволяет выделять схемный уровень и упрощать включение в схему разработки программ механизмов верификации (подобие модели или соответствие аксиомам), а на их основе подключать проверку программ на правдоподобие, логический вывод свойств, выполнение индуктивных и дедуктивных построений [6].

Практичность парадигм программирования

Границы между областями практичного проявления разных парадигм программирования можно выразить типичными схемами постановок задач на программирование.

Стандартное императивно-процедурное программирование: «Определён алгоритм решения актуальной задачи. Необходимо получить программу реализации алгоритма с практичными пространственно-временными характеристиками на доступном оборудовании».

Функциональное программирование: «Известна предметная область. Следует выбрать символьное представление данных для этой области и отладить систему универсальных функций, пригодных для создания прототипов решения актуальных задач из этой области».

Логическое программирование: «Дана коллекция фактов и отношений, характеризующая актуальную задачу. Надо привести эту коллекцию к форме, демонстрирующей ответы на практичные запросы относительно данной задачи».

Объектно-ориентированное программирование: «Доступна иерархия классов объектов с работоспособными методами решения ряда задач некоторой сферы приложения информационных технологий. Требуется без лишних трудозатрат уточнить эту иерархию, чтобы приспособить её к решению новых востребованных задач этой области, её расширения или ей подобной».

Многие задачи включают такие формулировки в качестве подзадач, что приводит при создании языков программирования и разработке систем программирования к поддержке разных парадигм одновременно. Так, например, при целенаправленной разработке монопарадигмального языка Haskell, позиционируемого как чисто функциональный язык, авторы пришли к концепции монад, позволяющей привлекать механизмы других парадигм. Потребность в поддержке парадигм, отсутствующих в реализуемом языке программирования, часто обеспечивается в системе программирования посредством библиотечных процедур, ассемблерных вставок, макрогенераторов или организацией выхода на уровень операционной системы.

В практике признают основными парадигмы императивно-процедурного, функционального, логического и объектно-ориентированного программирований, поддерживающие механизмы снижения трудоёмкости полного жизненного цикла программ, с тенденцией продвижения к параллельному программированию. В настоящее время по существу различают более двух десятков парадигм. Многие языки программирования относят к пяти-восьми парадигмам. Часть упоминаемых в разных источниках парадигм можно характеризовать как стили или методики, отражающие поиск путей снижения трудоёмкости программирования и повышения надёжности программ на базе доступных систем программирования. Например, аспектно-ориентированное программирование поддерживается как макрорасширение объектно-ориентированного программирования. Структурное программирование фактически сводится к ряду ограничений на стиль представления императивно-процедурных программ. Метапрограммирование представляет собой технику компиляции программ в комплекте с типовыми элементами данных. Встречаются прецеденты рассмотрения недетерминизма как основной реализации параллелизма.

Переход к параллелизму

Следует особо отметить трудоёмкость параллельного программирования, приводящую к тенденции включать его в любую парадигму на правах обобщения отдельных конструкций и тем самым откладывать на будущее необходимость разработки методов верифицирующей компиляции и оптимизации программных компонентов, средств масштабируемой генерации кода и автоматизируемых трансформаций программ с удостоверением сохранения свойств при их комплексации из ранее отлаженных компонентов [3; 4; 5; 8]. Парадигма параллельного программирования занимает нишу, связанную с реализацией программ выполнения вычислений на многопроцессорных системах для организации высокопроизводительных вычислений [1]. Эта ниша обременена резким повышением сложности отладки программ, вызванной комбинаторикой выполнения фрагментов асинхронных процессов. Если первый барьер к успеху в параллельном про-

граммировании обусловлен последовательным стилем мышления, то второй барьер зависит от концептуально не преодолённой трудоёмкости отладки программ. Хотя основная трудоёмкость жизненного цикла программ связана именно с тестированием и отладкой программ, это направление за полвека профессионального программирования прогрессирует преимущественно за счёт «силовых» характеристик оборудования, концептуально не выходя за пределы возможностей экстракодов отладки программ в доязыковую эпоху. Целесообразность методов верификации программ, уже достигших практичных характеристик, ещё не получила доверия наших практиков, возможно из-за квалификационного ценза на создание формальных спецификаций, проблематика которых препятствует серьёзному отношению к вопросам отладки параллельных программ, что представлено в форме появления теорий для обработки информации, подверженной искажениям, и методики сравнительной отладки программ, в которой задействовано понятие эталонной программы и схемы распределённого отладчика [3].

Как правило, языки и системы параллельного программирования включают в себя средства, характерные для разных парадигм. Это определяет возможность трансформационного подхода к накоплению правильности программных решений при разработке и модернизации параллельных программ на разных языках в рамках общей системы программирования. Развитие языков и систем программирования в настоящее время ориентировано на технику использования общих библиотечных модулей, обеспечивающих эффективную организацию процессов, или подъязыков, допускающих многопоточное программирование. Это не исключает практику ручного распараллеливания ранее отлаженных обычных программ, приведения их к виду, удобному для применения производственных систем поддержки параллельных вычислений. Значительная часть таких работ носит рутинный характер и заключается в систематической реорганизации структур данных, изменении статуса переменных и включении в программу аннотаций, сообщающих компилятору об информационно-логических взаимосвязях. Существенным ограничением результата ручного распараллеливания является не только опасность повторной отладки алгоритма, но и его избыточная зависимость от характеристик целевой архитектуры.

Операционная семантика параллелизма

Разнообразие моделей параллельных вычислений и расширение спектра доступных архитектур следует рассматривать как вызов разработчикам языков и систем программирования, претендующим на решение проблем создания методов компиляции многопоточных программ для многопроцессорных конфигураций. Язык должен допускать представление любых моделей параллелизма, проявляемого на уровне решаемой задачи или реализуемого с помощью реальной архитектуры, причем такое представление требует лаконичных разноуровневых форм и конструктивных построений, гарантирующих сохранение свойств программ при их реорганизации. Не менее важна расширяемость языка по мере развития средств и методов параллельных вычислений, темп которого превышает скорость осознания специалистами их возможностей.

Переход к параллельным алгоритмам влечёт пересмотр содержания многих понятий и введение новых терминов, отражающих разного рода явления и эффекты, не имевшие особого значения для обычных последовательных алгоритмов. Алгоритмы становятся параллельными, программы – многопоточными, исполнение кода – многопроцессорным, действие может начать выполняться до завершения предыдущего, средства управления кроме логических значений используют сигналы и сообщения, обработка структур данных может требовать изменения порядка их обработки и многое другое. Возникающее в результате расширение пространства понятий меняет подходы к реализации решений, использующих параллелизм, и в некоторой мере сказывается собственно на постановке задач и планировании жизненного цикла программ решения задач, ориентированных на использование параллелизма.

Принципиальная разница между последовательным и параллельным программированием может быть выражена различием операционной семантики языков высокого и сверхвысокого уровня. Семантика языка высокого уровня может быть сведена к детерминированному автомату. Семантика языка сверх высокого уровня требует тиражируемого автомата, определяющего вариации допустимых процессорных конфигураций.

Парадигмальная характеристика языков и систем программирования

Рассмотрение технических особенностей языков и систем программирования через парадигмальную характеристику может дать удобные формы их определения относительно ряда простых концептуальных языков, включая языки низкого уровня, накопившие механизмы обработки данных, унаследованные методами реализации языков высокого уровня. Сложность и противоречивость классификации стремительно расширяющегося множества языков и систем программирования при сравнительно небольшом наборе различимых парадигм программирования приводит к понятию «определитель» поддержанных в языков и систем программирования парадигм [5], содержащий следующие процедуры:

– разложение языка на фрагменты по уровням с целью выделения базовых средств языка и его реализационного ядра – семантический базис;

– декомпозиция семантического базиса языка на основные семантические системы с минимизацией их сложности и, возможно, их описание относительно концептуальных языков, поддерживающих известные парадигмы;

– определение абстрактной машины языка и интерпретатора, формально достаточного для построения расширений, эквивалентных исходному языку – нормализованное определение;

– сравнение полученного определения с описаниями известных парадигм и концептуальных языков;

– обоснование выводов относительно парадигм исследуемого языка;

– определение уровня языка и его ниши в жизненном цикле программ и деятельности программистов (цели и задачи), а также, базовых языков, использованных при его создании и реализации, в качестве основы для рекомендаций по выбору и применению языка программирования и его систем программирования.

Рассматривая задачу формализации языков параллельного программирования как путь к решению проблемы адаптации программ к различным особенностям используемых многопроцессорных комплексов и многоядерных процессоров, можно сделать вывод, что решение этой проблемы требует разработки новых методов компиляции программ с акцентом на тестирование, верификацию и отладку, а также, развития средств и методов ясного описания семантики языков параллельного программирования, включая представление программируемых преобразований текста кода программы с удостоверением их корректности.

Образовательное значение парадигм программирования

Пришло время изучать информатику и программирование, начиная с мира параллелизма. Трудно не замечать, что выполнение «одинаковых» действий обладает разной длительностью, что время реагирования информационной системы может зависеть от текущей ситуации, что собственно выполнимость действий зависит от не всегда заранее известных условий, что системы можно и нужно настраивать, что ряд систем могут работать одновременно и влиять на работу друг друга. Существуют кэши, протоколы, верификаторы, резервное копирование, транзакции, «гонки» данных, «смертельное объятие» и многое другое. Известно об успехах любительской астрономии, перспективах GRID-технологий и «облачных» вычислений. Активизация таких знаний вместе с интуитивным опытом обычной деятельности образует основу для быстрого изучения

средств и методов параллельного программирования, начало которому дает предлагаемый экспериментальный курс [6].

Особый круг образовательных проблем связан с навыками учёта особенностей многоуровневой памяти в многопроцессорных системах. Обычное последовательное программирование такие проблемы может не замечать, полагаясь на решения компилятора, располагающего статической информацией о типах используемых данных и способного при необходимости выполнить оптимизирующие преобразования программы. К сожалению, использование языков параллельного программирования в качестве языка представления программы не гарантирует её приспособленность к удачному распараллеливанию.

При оценке образовательного значения парадигм выделяют в качестве принципиальных функциональное, параллельное и императивно-процедурное программирования, а логическое и объектно-ориентированное рассматривают как дополнение, изучаемое в виде расширения ранее изученных парадигм [9].

Заключение

В данной работе новыми являются следующие положения и результаты:

– определены поляризуемые, противопоставленные характеристики языков и систем программирования;

– сформулированы типичные схемы постановок задач на программирование согласно выбранной парадигме;

– приведены результаты анализа средств и методов параллельного программирования;

– предложена методика парадигмальной характеристики языков и систем программирования.

Заметим, что наиболее успешные, долго живущие языки и системы программирования являются мультипарадигмальными. Объединение разных парадигм в рамках одного языка программирования или их поддержка при реализации систем программирования повышает их сферу применения и способствует производительности труда программистов, смягчая избыточное разнообразие теорий и моделей, сложившихся на разных фазах и этапах жизненного цикла программ. Многие авторы новых технологий программирования, таких как экстремальное программирование и функционально ориентированное проектирование, выражают претензии к исторически сложившимся рекомендациям по стилю и технике программирования, якобы сделавшим практику программирования беспомощной [10; 11]. Здесь желательно помнить о высоком темпе развития информационных технологий, осознание возможностей которых не успевает созревать. Мультипарадигмалльность долго живущих языков и систем программирования и тенденция создания новых мультипарадигмальных языков программирования говорит о созревании единой парадигмы, объединяющей выверенные в практике механизмы программирования [8]. Не исключено, что развитие методов компиляции таких языков приведёт к добротным средствам и методам подготовки параллельных программ [2; 4].

Литература

1. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – СПб.: БХВ-Петербург, 2002. 608 с.

2. *Степанов Г.Г.* Пути обеспечения переносимости программ и опыт использования системы СИГМА // Трансляция и преобразование программ. – Новосибирск: ВЦ СО АН СССР, 1984. 9 с.

3. *Бурдонов И.Б., Косачев А.С., Кулямин В.В.* Теория соответствия для систем с блокировками и разрушениями / – М.: Физматлит, 2008. 412 с.

4. *Knoop J.* Compiler Construction / 20th International Conference, CC 2011Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011 Saarbrьcken,

Germany, March 26 –April 3, 2011 // Lecture Notes in Computer Science. Springer. V2011. V. 6601. 330 p.

5. *Городняя Л.В.* О проблеме автоматизации параллельного программирования // В сборнике Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров». URL: http://agora.guru.ru/abrau2014.

6. *Городняя Л.В.* Парадигма программирования : курс лекций / Л.В. Городняя ; Новосиб. гос. Ун-т. – Новосибирск: РИЦ НГУ, 2015. 206 с.

7. *Городняя Л.В.* Парадигмы параллельного программирования в университетских образовательных программах и специализации // Всероссийская научная конференция «Научный сервис в сети Интернет: решение больших задач». – Новороссийск-Москва, 2008. с. 180-184.

8. *Хорстман К.* Scala для нетерпеливых. – ДМК пресс, 2013. 408 с.

9. Pet*er Van Roy*. The principal programming paradigms (2008). – Avalable at: https://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf.

10. Бек К. Экстремальное программирование – Питер, 2003. 224 с.

11. *Палмер С.Р., Фелсинг. Дж.М.* Практическое руководство по функционально ориентированной разработке ПО. – М.: Вильямс, 2002. 299 с.

Classification of programming and parallel programming paradigms

***Lidia Vasiljevna Gorodnyaya**, Ph.D., Docent, Senior Researcher*

The article concerns the actual problem of study and development of methods of analysis, comparison and formal definition of the programming paradigms. Importance of this topic comes from the steep increase in the number of new-generation programming languages oriented at application and development of modern information technologies.

Keywords – programming languages and systems; programming paradigms; computer languages definition methods; parallel programming; educational programming.

УДК 004.4'22:004.891

ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО СОЗДАНИЯ ПРОДУКЦИОННЫХ ЭКСПЕРТНЫХ СИСТЕМ НА ОСНОВЕ MDA

***Максим Андреевич Грищенко**, программист*
Тел.: 8 395 245 31 57, e-mail: makcmg@icc.ru
Институт динамики систем и теории управления им. А.М. Матросова СО РАН
http://www.icc.irk.ru
***Ольга Анатольевна Николайчук**, д-р техн. наук, с.н.с.*
Тел.: 8 395 245 31 57, e-mail: nikoly@icc.ru
Институт динамики систем и теории управления им. А.М. Матросова СО РАН
http://www.icc.irk.ru
***Александр Иннокентьевич Павлов**, канд. техн. наук, с.н.с.*
Тел:. 8 395 245 30 19, e-mail: asd@icc.ru
Институт динамики систем и теории управления им. А.М. Матросова СО РАН
http://www.icc.irk.ru
***Александр Юрьевич Юрин**, канд. техн. наук, зав.лаб.*
Тел.: 8 395 245 30 19, e-mail: iskander@icc.ru
Институт динамики систем и теории управления им. А.М. Матросова СО РАН
http://www.icc.irk.ru