Технические науки

УДК 004.4

Молодяков Сергей Александрович
доктор технических наук, доцент,
профессор Санкт-Петербургского политехнического
университета Петра Великого
Molodyakov Sergey
doctor of engineering sciences, professor,
Peter the Great Saint-Petersburg Polytechnic University

Петров Александр Владимирович
старший преподаватель Санкт-Петербургского
политехнического университета Петра Великого
Petrov Alexander
senior lecturer, Peter the Great Saint-Petersburg
Polytechnic University

Молодяков Александр Сергеевич
магистр Санкт-Петербургского политехнического
университета Петра Великого
Molodyakov Alexander
Master, Peter the Great Saint-Petersburg
Polytechnic University

МЕТОДОЛОГИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ И МЕТОДЫ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ИНФОРМАЦИИ
METHODOLOGY OF SOFTWARE ENGINEERING AND METHODS OF PARALLEL PROCESSING

Аннотация: Представлены основные элементы и понятия системы знаний в программной инженерии. В рамках систематизации выделены методология, области знаний и инструменты. Определена связь между методами написания параллельных программ и методами параллельного выполнения команд. Области знаний программной инженерии можно соотносить с изучаемыми в университетах дисциплинами.

Ключевые слова: программная инженерия, методология, инструменты программной инженерии, методы параллельных вычислений.

Summary: Basic elements and concepts of system of knowledge of software engineering are presented. Within systematization bound the methodology, area of knowledge and tools are allocated. Communication between methods of writing of parallel programs and methods of parallel execution of commands is defined. Area of knowledge of program engineering can be correlated to the disciplines studied at universities.

Key words: software engineering, methodology, software engineering tools, methods of parallel processing.

Введение. Каждый год при встрече с очередными абитуриентами, желающими работать в области программирования, встает вопрос что такое программная инженерия (ПИ) и где ее лучше изучать. Преподаватели кроме множества вопросов научной, преподавательской, административной работ [1] должны отвечать и на эти вопросы. Ниже представлена во многом известная информация, которая получена из публикаций [2-6], а также из опыта работ авторов по разработке и применению вычислительных систем для радиоастрономии и обработке видеоинформации [7-16]. Для доступности информация частично представлена на сайте кафедры информационных и управляющих систем СПбПУ (http://ics.ftk.spbstu.ru/).

Программная инженерия – приложение систематического, дисциплинированного, измеримого подхода к разработке, функционированию и сопровождению программного обеспечения, а также исследованию этих подходов. Термин – software engineering (программная инженерия) - впервые был озвучен в октябре 1968 года на конференции подкомитета НАТО по науке и технике (г. Гармиш, Германия) [4].

Одним из основных понятий ПИ является понятие жизненного цикла программного продукта и программного процесса. **Жизненный цикл** ПО – период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации. Этот цикл - процесс построения и

развития ПО. Жизненный цикл разбивается на отдельные процессы. Процесс – совокупность действий и задач, имеющих целью достижение значимого результата.

Основными процессами (иногда называют этапами или фазами) жизненного цикла являются:

- Разработка спецификации требований (результат – описания требований к программе, которые обязательны для выполнения – описание того, что программа должна делать);
- Разработка проекта программы (результат – описание того, как программа будет работать);
- Кодирование (результат – текст программы и файлы конфигурации);
- Тестирование программы (результат - контроль соответствия программы требованиям);
- Документирование (результат – документация к программе);
- Сопровождение (внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям).

Модель жизненного цикла ПО – структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач на протяжении жизненного цикла. Модель жизненного цикла зависит от специфики, масштаба и сложности проекта и специфики условий, в которых система создается и функционирует. Модель задается в виде практических этапов, необходимых для создания ПО. В модели мы говорим, что и как мы будем делать: какие процессы и в какой последовательности будем выполнять. Правильный выбор модели во многом определяет успех проекта.

Международные стандарты определяют практически все процессы жизненного цикла сложных программных систем. Процесс стандартизации и сертификации составляет основу промышленного производства программных продуктов.

Наиболее известными стандартами ПИ являются:

ISO/IEC 12207 - Information Technology - Software Life Cycle Processes - Процессы жизненного цикла программных средств. Стандарт содержит определения основных понятий ПИ (в частности программного продукта и жизненного цикла программного продукта). ·

SEI CMM - Capability Maturity Model (for Software) - модель зрелости процессов разработки программного обеспечения. Стандарт отвечает на вопрос: «Какими признаками должна обладать профессиональная организация по разработке ПО?». ·

PMBOK - Project Management Body of Knowledge - Свод знаний по управлению проектами.

SWBOK - Software Engineering Body of Knowledge - Свод знаний по ПИ - содержит описания состава знаний по разделам (областям знаний) ПИ. ·

ACM/IEEE CC2001 - Computing Curricula 2001 – Академический образовательный стандарт в области компьютерных наук. Выделены 4 основных раздела компьютерных наук: Computer science, Computer engineering, Software engineering и Information systems, по каждому из которых описаны области знаний соответствующего раздела, состав и планы рекомендуемых курсов.

В настоящее время сообществом SWBOK разрабатывается расширенная версия знаний по ПИ, которая включает 15 областей: Software Requirements — требования к ПО; Software Design — проектирование ПО; Software Construction — конструирование ПО; Software Testing — тестирование ПО; Software Maintenance — сопровождение ПО; Software Configuration Management — управление конфигурацией; Software Engineering Management — управление IT проектом; Software Engineering Process — процесс ПИ; Software Engineering Models and Methods — модели и методы разработки; Software Engineering Professional Practice — описание критериев профессионализма и компетентности; Software Quality — качество ПО; Software Engineering Economics — экономические аспекты разработки ПО; Computing Foundations — основы вычислительных технологий, применимых в

разработке ПО; Mathematical Foundations — базовые математические концепции и понятия, применимые в разработке ПО; Engineering Foundations — основы инженерной деятельности.

Метод программной инженерии — это структурный подход к созданию ПО, который способствует производству высококачественного продукта эффективным с точки зрения экономики способом. В этом определении есть две основные составляющие: (а) создание высококачественного продукта и (б) экономически эффективным способом. Иначе, метод – это то, что обеспечивает создание качественного программного продукта при заданных ограниченных ресурсах времени, бюджета, оборудования, людей.

Методология ПИ и стандарты регламентируют современные процессы управления проектами сложных систем и программных средств. Они обеспечивают организацию, освоение и применение апробированных, высококачественных процессов проектирования, программирования, верификации, тестирования и сопровождения программных средств и их компонентов. Тем самым эти проекты и процессы позволяют получать стабильные, предсказуемые результаты и программные продукты требуемого качества.

Программные инструменты предназначены для обеспечения поддержки процессов жизненного цикла программного обеспечения. Инструменты позволяют автоматизировать определенные повторяющиеся действия, уменьшая загрузку инженеров рутинными операциями и помогая им сконцентрироваться на творческих, нестандартных аспектах реализации выполняемых процессов.

CASE (Computer-Aided Software Engineering) – набор инструментов и методов ПИ для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Выделяют 9 групп инструментов ПИ (Software Engineering Tools) [2]. Так к инструментам конструирования (Software Construction Tools) относят:

редакторы (program editors), компиляторы и генераторы кода (compilers and code generators), интерпретаторы (interpreters), отладчики (debuggers).

В целом методология, методы и инструменты ПИ не связаны с аппаратной платформой, на которой выполняются программы. Однако появившиеся аппаратные возможности параллельного выполнения команд и задач должны быть учтены в таких инструментах как компиляторы, отладчики и других инструментах проектирования и тестирования ПО. Рассмотрим современные методы параллельного выполнения программ (параллелизации) и их связь с ПИ.

Параллелизация. Методы параллельного выполнения команд основаны на современных возможностях вычислительной техники. Большинство современных процессоров и соответственно суперкомпьютеров поддерживают методы векторного (а не скалярного) выполнения команд (векторизации) и выполнения программ (потоков команд) на нескольких ядрах [7, 8]. На рис. 1 представлена схема связи методологии ПИ и методологии построения вычислительной техники (ВТ). Эта связь осуществляется через внутренние элементы: с одной стороны - методы и инструменты разработки параллельных алгоритмов и программ; и с другой стороны - методы параллелизации команд.

Рис.1. Схема связи методологии ПИ и методологии построения ВТ (разработка авторов).

Векторизация или векторная обработка массивов данных связана с возможностью выполнения команды над несколькими операндами

одновременно (в классификации Флинна – SIMD single instruction, multiple data). При использовании векторизации можно в несколько раз повысить скорость обработки. Необходимость применения векторизации видно из процесса развития процессоров Intel. Команды класса SIMD над регистрами MMX (64 разряда) появились в первых процессорах Pentium. Затем SIMD-команды нашли свое развитие в технологиях SSE (128 разрядов) и AVX (256 разрядов). В настоящее время появляются процессоры с SIMD-регистрами разрядностью 512, 1024. Количество SIMD-команд превосходит количество команд вычислительного ядра процессора. SIMD-технология поддержана как на низком, так и на высоком уровне программирования. На низком уровне применяются ассемблерные вставки и векторные intrinsic функции (библиотеки типа xmmintrin.h). На высоком уровне для векторизации используют или специальные директивы, например, из библиотеки OpenMP 4.0, или автовекторизация при выборе необходимого уровня оптимизации компилятора.

Параллельная обработка на вычислительных ядрах. Известны три парадигмы использования вычислительных ядер. Первая парадигма определяется архитектурой применения общей памяти несколькими процессорными ядрами. Примером является архитектура процессоров i7, которые включают, например, 4 ядра, расположенных на общей шине с памятью. Программирование поддержано с использованием библиотеки OpenMP. Она позволяет организовать множество параллельных потоков команд, выполняемых на вычислительных ядрах компьютера. Функции и директивы библиотеки предоставляют большие возможности по контролю над поведением параллельных программ. Вторая парадигма связана с архитектурой систем с распределенной памятью. Примером являются различные архитектуры суперкомпьютеров. Программирование осуществляется с использованием библиотеки MPI (Message Passing Interface). MPI представляет собой программный интерфейс (API) для передачи информации, который позволяет обмениваться сообщениями между

процессами, выполняющими одну задачу. Третья парадигма связана с использованием гетерогенных вычислений, например, с применением языка OpenCL (Open Computing Language).

Ниже представлена возможная реализация алгоритма с применением функции `omp parallel for` библиотеки OpenMP и функций `cvQueryFrame` (захват картинки с камеры или с видеофайла), `CV_GAUSSIAN` (сглаживание по гауссиане в области 3x3 вокруг каждого пикселя изображения), `cvWriteFrame` (запись кадра в видеофайл) библиотеки OpenCV. Программа захватывает кадр, фильтрует по гауссиане и записывает в выходной видеофайл. Захват (чтение) и вывод кадров происходит последовательно, для этой цели применена директива `omp critical`. Обработка полученных кадров происходит параллельно тем количеством вычислительных ядер, которые имеются в вычислительной системе.

```
#pragma omp parallel for
     for (int i = 0; i < count; i++) {
          IplImage *inputFrame3;
#pragma omp critical
          {inputFrame3 = cvQueryFrame(input3);}
          cvSmooth(inputFrame3, inputFrame3, CV_GAUSSIAN, 5, 3);
#pragma omp critical
          {cvWriteFrame(parallelGaussWriter, inputFrame3);}
     }
```

Инструменты ПИ всегда поддерживают достижения вычислительной техники, в том числе и в области параллелизации вычислений. Так вычислительный пакет Intel Parallel Studio, включающий широкий набор инструментов разработки и тестирования ПО, поддерживает описанные ранее методы параллелизации.

Таким образом, знания специалиста по ПИ должны охватывать широкий круг вопросов, которые касаются не только методов и средств разработки ПО, но и методов и возможностей ВТ. Кем бы не работал специалист в области программной инженерии, являясь разработчиком прикладного и системного ПО, организатором и руководителем (менеджером проектов) промышленной разработки программных систем, он должен быть подготовлен к постоянному наращиванию своих знаний, к привлечению новых методов и подходов.

Список литературы

1. Молодяков С.А. Преподаватель в вузе: из опыта повседневной жизни / Высшее образование в России.- 2016.- № 3.- С. 91-98.
2. Орлик С., Булуя Ю. Введение в программную инженерию и управление жизненным циклом. http://software-testing.ru/library/around-testing/engineering/267-swebok.
3. Программная инженерия http://iibs.vvsu.ru/ispi/nap/pi/.
4. Naur P., Brian R. (7–11 October 1968). Software Engineering: Report of a conference sponsored by the NATO Science Committee(PDF). Garmisch, Germany: Scientific Affairs Division, NATO. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.
5. ISO/IEC TR 19759:2015 Software Engineering -- Guide to the software engineering body of knowledge (SWEBOK) http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=67604.
6. Липаев В.В. Программная инженерия в жизненном цикле программных средств http://citforum.ru/SE/lipaev/.
7. Молодяков С.А. Системное проектирование оптоэлектронных процессоров обработки сигналов. СПб.: Изд-во Политехн. ун-та, 2011.- 226 с.
8. Молодяков С.А. Проектирование специализированных цифровых видеокамер. / СПб.: Изд-во Политехн. ун-та, 2016 .— 286 с.
9. Молодяков С.А. Управление информационными характеристиками фотоприемника на приборе с зарядовой связью в устройстве ввода изображения в ЭВМ / Приборы и техника эксперимента.- 1987.- № 3.- С. 71-75.
10. Лавров А.П., Молодяков С.А., Саенко И.И. Акустооптоэлектронные устройства в радиоастрономических приемных комплексах / Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление.- 2010.- Т.4.- № 103.- С.233-242.

11. Лавров А.П., Молодяков С.А., Саенко И.И. Акустооптические процессоры в радиоастрономических приемниках / Антенны.- 2009.- № 7.- С.45-55.
12. Гречнев В.В., Есепкина Н.А., Занданов В.Г. и др. Исследование макета акустооптоэлектронного приемника на Сибирском солнечном радиотелескоп / Письма в журнал технической физики.- 1988.- Т.14.- № 7. - С581-585.
13. Есепкина Н.А., Гаврилов Г.А., Лавров А.П. и др. Оптоэлектронный процессор на основе матричного ФПЗС с волоконной шайбой / Письма в Журнал технической физики. 1992.- Т.18.- № 3.- С.32-37.
14. Есепкина Н.А., Молодяков С.А., Саенко И.И. Организация синхронного накопления на матричном ПЗС фотоприемнике в модуляционном спектрометре / Письма в Журнал технической физики.- 1986.- Т. 12.- № 2.- С. 118-123.
15. Есепкина Н.А., Лавров А.П., Молодяков С.А. Акустооптический компенсатор дисперсии для сжатия импульсов радиоизлучения пульсаров / Известия высших учебных заведений России. Радиоэлектроника. 1998.- № 2.- С.21-29.
16. Лавров А.П. Молодяков С.А. Оптоэлектронный процессор для регистрации радиоизлучения пульсаров / Приборы и техника эксперимента.- 2015.- № 1.- С.136–145.