УДК 004.9
DOI:10.30533/GiA-2025-022

Поведенческая декомпозиция: метод проектирования адаптивных геоинформационных технологий

О.Г. Гвоздев[1] ✉

[1] Московский государственный университет геодезии и картографии, Москва, Россия
✉ gvozdev@miigaik.ru

ЦИТИРОВАНИЕ Гвоздев О.Г. Поведенческая декомпозиция: метод проектирования адаптивных геоинформационных технологий // Известия вузов «Геодезия и аэрофотосъемка». 2025. Т. 69. № 3. С. 74–93. DOI:10.30533/GiA-2025-022.

КЛЮЧЕВЫЕ СЛОВА поведенческая декомпозиция, адаптивные геоинформационные технологии, программная инженерия, программная архитектура

АННОТАЦИЯ В статье рассматривается проблема обеспечения адаптивности при разработке информационных систем в целом и геоинформационных систем и технологий в частности. Под адаптивностью в рамках данной работы понимается свойство архитектуры программного решения, способствующее (или препятствующее) удовлетворению новых требований до того, как они станут известны. Предлагается метод проектирования адаптивных геоинформационных технологий, получивший название «поведенческая декомпозиция». Область его применения — моделирование отдельных сложных процедур обработки пространственных данных. Рассмотрены существующие в отрасли подходы и предпосылки к появлению нового метода, его теоретическая база и лежащие в его основе принципы, в числе которых шаблоны проектирования «Стратегия» и «Спецификация», принцип инверсии управления, а также решения, предложенные Э. Эвансом в работе “Domain Driven Design”. Приведены приемы применения метода к практическим задачам, человеко-машиночитаемая нотация для записи решений на любом этапе проектирования. В качестве примеров рассмотрены модели двух процедур обработки пространственных данных, построенные с использованием предложенного метода: свертка двухмерного растра на регулярной сетке и метод семплирования и аугментации спутниковых изображений SEGA. Освещается опыт практического применения предложенного метода, выявленные ограничения и способы их преодоления.

1 Введение

Развитие любой области деятельности неразрывно связано с ростом потребления ее результатов, ростом ожиданий от этих результатов и, наконец, ростом требований к этим результатам. Удовлетворение внешних требований происходит за счет повышения внутренних требований к процессам достижения продуцируемых результатов: продуктов, сервисов или услуг. Так, программная инженерия прошла большой путь от доказательства концептуальной возможности решения элементарных задач до появления систем принципиально нового класса — так называемых программно-определяемых технологий[1,2] (*англ.* software-defined technologies) [1].

Первые достижения в этой области связаны с применением программно-определяемых технологий к решению задач инфраструктуры автоматизированных информационных систем. В их числе: программно-определяемые системы хранения[3] (*англ.* software-defined storage) [2, 3], программно-определяемая сетевая инфраструктура (*англ.* software-defined networking) [4, 5], программно-определяемая радиосвязь (*англ.* software-defined radio) [6, 7], концепция «инфраструктура как код»[4] (*англ.* infrastructure as a code)[5]. Активно исследуется возможность построения программно-определяемых производств (*англ.* software-defined factory, software-defined manufacturing) [8, 9]. Некоторые направления исследований и разработок, такие как системы распределенного реестра[6,7] [10, 11], смарт-контракты (*англ.* smart contracts) [12–14], в перспективе могут позволить перевести целые общественные институты в полностью автоматическую программно-определяемую форму.

В этих обстоятельствах приобретает особую значимость программируемость (способность изменять функционирование системы без изменения ее физической структуры), и, как следствие, программная инженерия становится одним из важнейших видов деятельности, а требования к ней и ее результатам оказываются как никогда высоки.

Производительность (*англ.* performance), эффективное использование вычислительных ресурсов (*англ.* resource-efficiency), надежность (*англ.* reliability), устойчивость к изменениям среды функционирования (*англ.* resilience), удобство эксплуатации (*англ.* useability), наблюдаемость (*англ.* observability), сопровождаемость (*англ.* maintainability) — в начале 2000-х годов этот список требований был близок к эталонному. В начале 2010-х годов к ним добавились горизонтальная и вертикальная масштабируемость (*англ.* horizontal and vertical scalability): потребность в программировании малых устройств с ограниченными вычислительными ресурсами и малым автономным запасом электропитания (смартфоны, планшеты, умные часы и другие формы носимых персональных компьютеров) выросла одновременно с потребностью

1 Software Defined Anything Market Overview / Market Research Future. [Электронный ресурс]. Режим доступа: https://www.marketresearchfuture.com/reports/software-defined-anything-market-24425 (дата обращения: 19.08.2024).

2 The software-defined future of industries / intive. [Электронный ресурс]. Режим доступа: https://www.intive.com/insights/the-software-defined-future-of-industries (дата обращения: 13.09.2024).

3 Software-defined storage (SDS) / TechTarget. [Электронный ресурс]. Режим доступа: https://www.techtarget.com/searchstorage/definition/software-defined-storage (дата обращения: 28.08.2024).

4 Moving from Infrastructure Automation to True DevOps. [Электронный ресурс]. Режим доступа: https://devops.com/moving-from-infrastructure-automation-to-true-devops (дата обращения: 02.09.2024).

5 Wittig A., Wittig M. Amazon Web Services in Action. New York: Manning Press, 2016. 93 p.

6 Distributed Ledger Technology: beyond block chain. [Электронный ресурс]. Режим доступа: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf (дата обращения: 07.09.2024).

7 Машиночитаемая доверенность // Федеральная налоговая служба: официальный сайт. [Электронный ресурс]. Режим доступа: https://m4d.nalog.gov.ru/emchd (дата обращения: 04.09.2024).

в разработке программных решений, обрабатывающих петабайты данных на десятках и сотнях тысяч вычислительных узлов.

К 2025 году все перечисленное стало немыслимым без облачных сервисов, позволяющих пользователю не заботиться о хранении данных или выполнении сложных ресурсоемких вычислений на собственных устройствах[8] [15], тщательно проработанного в части эстетики, эргономики, возможности интуитивного и рационального использования пользовательского интерфейса (*англ.* user interface, user experience, UI/UX)[9] [16], автоматизации с помощью моделей машинного обучения, включающих современные искусственные нейронные сети и большие языковые модели [17].

Нельзя не упомянуть и возросшие требования регуляторов по защите персональных данных и обеспечению других аспектов информационной безопасности, в частности требования к объектам критической информационной инфраструктуры, сложности работы информационных систем, функционирующих во множестве юрисдикций, и многие другие вопросы, не являющиеся по своей сути ни научными, ни инженерно-техническими, но активно влияющие на образ современных информационных систем. Даже относительно простые, узкоспециализированные, нишевые информационные системы сталкиваются со всеми перечисленными требованиями либо в явной (в виде требований заказчиков, контрагентов или регуляторов), либо в неявной (в виде пользовательских ожиданий) форме. При этом требования к скорости и рациональности разработки, в частности в виде минимизации времени между формированием идеи и выводом ее прототипа на рынок (*англ.* time to market)[10,11], также непрерывно растут.

Оптимизация процессов разработки программных решений полагается на развитие теоретической и инструментально-технологической базы и возможность задействования существующих программных решений путем их приспособления к новым задачам и обстоятельствам функционирования либо использования их компонентной базы для построения производных решений. Данные вопросы имеют как предметно-независимый аспект, например развитие языков программирования общего назначения и оптимизирующих компиляторов для них, развитие программных библиотек, практик и инструментальной базы разработки как таковой, так и предметно-зависимые аспекты.

Применительно к предметной области геоинформатики возможность задействования существующих программных решений рассмотрена в первой публикации данного цикла [18] в широком смысле — как проблема адаптивности программных решений. При этом особое внимание уделяется функциональной адаптивности и устоявшимся шаблонам ее реализации и эксплуатации, обуславливающим, в свою очередь, текущий (относительно высокий) уровень адаптивности геоинформационных систем, основы которого были заложены ввиду роли геоинформационных систем как средства интеграции и совместной обработки сведений, изначально относящихся к различным предметным областям, предполагающим различные способы получения и представления. Однако сформированные подходы, сохранившись со времен формирования геоинформатики без существенных изменений, к настоящему моменту начинают уступать прогрессу, достигнутому в смежных областях, емких в части применения информационных технологий, и, как следствие, перестают отвечать потребностям практики.

---

8 Mell P., Grance T. The NIST Definition of Cloud Computing / National Institute of Standards and Technology, 2001. [Электронный ресурс]. Режим доступа: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (дата обращения: 15.12.2024).

9 The Definition of User Experience (UX) / NNgroup. [Электронный ресурс]. Режим доступа: https://www.nngroup.com/articles/definition-user-experience (дата обращения: 09.09.2024).

10 Cohen M. A., Eliasberg J., Ho T.H. New product development: The performance and time-to-market tradeoff // Management Science. 1996. Vol. 42. No. 2. P. 173–186.

11 Brown C.L., Lattin J. Investigating the Relationship Between Time in Market and Pioneering Advantage // Management Science. 1994. Vol. 40. No. 10. P. 1361–1369.

В предыдущей работе автора [18] выявлена системная проблема программной инженерии, свойственная геоинформационным технологиям, и для дальнейшей теоретической и прикладной проработки поставлен вопрос о том, какие принципы и методы должны использоваться при проектировании сложных процедур обработки пространственных данных, чтобы полученная реализация обладала высокой степенью адаптивности при сохранении возможности обеспечить производительность на уровне специализированных решений, не отягощенных механизмами адаптивности. Кроме того, показана важность решения данного вопроса как в производственных процессах, т. е. при разработке решений, ориентированных на конечных пользователей, так и в исследовательских процессах, направленных на поиск решений, предполагающих множество экспериментов.

В данной работе предлагается оригинальный метод проектирования адаптивных геоинформационных технологий, набор приемов, уточняющих его применение на практике, а также технические решения, способствующие разработке с его применением.

2 Материалы и методы

2.1 Предпосылки и теоретическая основа метода поведенческой декомпозиции

Создание программной реализации любой нетривиальной процедуры обработки пространственных данных не является задачей с однозначным решением. Так, основная сложность чаще всего заключается в определении оптимального с точки зрения заявленных требований решения из множества всех возможных. При этом перечень заявленных требований может быть как неполным, так и избыточным, может содержать противоречия, неточности или быть частично неактуальным (устаревшим).

Такие свойства, как производительность, зависят от рациональности последовательности команд, которая фактически выполняется вычислительным устройством. Однако, например, сопровождаемость и адаптивность определяются главным образом тем, насколько структура исходного кода способствует его модификации, а семантика удобна для понимания и в оптимальной степени моделирует предметную область.

В наиболее вычислительно-эффективных решениях сохраняется минимум семантических свойств предметной области. И напротив, исходный код, отражающий все детали предметной области, имеет тенденцию к низкой производительности. Данное противоречие частично решается с помощью оптимизирующих компиляторов. Особого внимания заслуживают достижения проектов Julia [19] и Triton[12], основанных, в свою очередь, на достижениях проекта LLVM [20]. Однако полное решение этой проблемы — получение оптимальной с точки зрения вычислительной эффективности реализации из оптимальной с точки зрения сопровождаемости и адаптивности реализации — на текущем уровне научно-технического развития не представляется возможным. С этим фактом в значительной степени связано множество современных исследований и разработок.

Так, новый подход к проблеме развивается в области моделирования искусственных нейронных сетей, в частности в рамках проектов PyTorch [21] и JAX[13].

12 Introducing Triton: Open-source GPU programming for neural networks / OpenAI. [Электронный ресурс]. Режим доступа: https://openai.com/index/triton/ (дата обращения: 02.09.2024).

13 Bradbury J., Frostig R., Hawkins P., et al. JAX: Autograd and XLA // Astrophysics Source Code Library. 2021. [Электронный ресурс]. Режим доступа: https://ascl.net/2111.002 (дата обращения: 15.12.2024).

Его можно рассматривать как гибридизацию императивной и декларативной парадигм программирования: императивная программа на медленном, но выразительном языке программирования Python используется как декларативная. Из нее извлекается структура базовых операций, далее являющаяся основой для генерации машинного кода, в котором операции «сплавлены» (*англ.* fused) в единое целое и адаптированы к текущим обстоятельствам эксплуатации: конкретному оборудованию (CPU или GPGPU), обрабатываемым типам данных и др. Этот процесс реализуется на базе таких компонентов, как Triton[14], Torch Dynamo[15], XLA[16,17].

Это не является компиляцией в привычном смысле: исходная программа не проходит цепочку эквивалентных преобразований, приводящую ее к результирующей форме, а используется для формирования требований к итоговой программе. Этот подход применим лишь для малого подмножества возможностей базового языка программирования и не подходит для задач, где сложность структур данных и вариативность их поведения доминируют над объемом вычислений. Качественная и полная реализация этого подхода чрезвычайно сложна и трудоемка, что ограничивает его применимость только задачами, одновременно очень востребованными и при этом требовательными к вычислительным ресурсам. Ярчайшим примером задач такого типа является моделирование искусственных нейронных сетей, для которого данные системы и реализованы. В силу этого традиционные подходы программной инженерии не утрачивают актуальности.

Выявленные в работе [18] классы адаптивности существенно отличаются по «мощности» — выразительным возможностям или «моделирующей способности». Так, к одним из самых «мощных» относится A.INJECT — инъекционная адаптивность: если процедура параметризуется опцией-перечислением (A.ENUM), то ее можно преобразовать в процедуру, параметризуемую произвольной реализацией (A.INJECT). Любая композиция (A.COMPOSE) — конвейер или граф операций (представление, являющееся стандартом де-факто при моделировании процессов обработки пространственных данных в геоинформационных системах) — также может быть выражена с помощью структуры, в которой одни исполняемые блоки параметризуются другими (A.INJECT). Обратные преобразования без потери функциональности в общем случае невозможны.

В объектно-ориентированном программировании (ООП) такое преобразование соответствует шаблону проектирования «Стратегия»[18,19] [22, 23], а в функциональном программировании реализуется на базе функций высших порядков. Дальнейшим обобщением этих принципов являются концепции инверсии управления (*англ.* Inversion of Control) и внедрения зависимостей (*англ.* Dependency Inversion)[20,21].

---

14 Introducing Triton: Open-source GPU programming for neural networks / OpenAI. [Электронный ресурс]. Режим доступа: https://openai.com/index/triton/ (дата обращения: 02.09.2024).

15 Dynamo Overview / PyTorch [Электронный ресурс]. Режим доступа: https://pytorch.org/docs/stable/torch.compiler_dynamo_overview.html (дата обращения: 12.09.2024).

16 Bradbury J., Frostig R., Hawkins P., et al. JAX: Autograd and XLA // Astrophysics Source Code Library. 2021. [Электронный ресурс]. Режим доступа: https://ascl.net/2111.002 (дата обращения: 15.12.2024).

17 OpenXLA. [Электронный ресурс]. Режим доступа: https://openxla.org (дата обращения: 13.09.2024).

18 Refactoring to patterns. Replace conditional logic with strategy / Medium. [Электронный ресурс]. Режим доступа: https://medium.com/@Yerazhas/refactoring-to-patterns-replace-conditional-logic-with-strategy-9970e057093a (дата обращения: 08.09.2024).

19 Garbinato B., Guerraoui R. Using the Strategy Design Pattern to Compose Reliable Distributed Protocols // Proceedings of the 3rd USENIX Conference on Object-Oriented Technologies and Systems (COOTS'97). Berkeley, USENIX Association, 1997. Vol. 3. P. 17–29.

20 Gamma E., Helm R., Johnson R., et al. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1994. 395 p.

21 Johnson R.E., Foote B. Designing Reusable Classes // Journal of Object-Oriented Programming. 1988. Vol. 1. No. 2. P. 22–35.

Особый интерес с точки зрения адаптивности представляет шаблон ООП «Спецификация»[22], предполагающий представление сложных правил за счет комбинирования элементарных правил с помощью логических операций «Не», «Или», «И».

Такая объектная структура может отвечать на вопрос: «Соответствует ли объект А спецификации?», — может быть применена для генерации своего представления на другом языке программирования, что используется для генерации SQL-запросов в ORM[23]. Подобные спецификации могут иметь произвольное количество точек входа, а интроспекция[24] позволяет применять к ним преобразования, изначально не предусмотренные в их реализации, в частности техники метапрограммирования, аналогичные упомянутым в начале раздела.

Особо необходимо отметить работу Э. Эванса “Domain Driven Design”[25], которая посвящена предметно-ориентированному проектированию программных систем и в которой предлагается комплексный подход по воплощению модели предметной области в архитектуре информационной системы.

Результатом авторского переосмысления и обобщения перечисленных методов программной инженерии в приложении к задачам, характерным для геоинформатики (главным образом задачам разработки сложных процедур преобразования пространственных данных), является предлагаемый в данной работе метод поведенческой декомпозиции.

2.2 Концепция метода поведенческой декомпозиции

Метод поведенческой декомпозиции — это метод анализа и моделирования предметной области и проектирования программных решений, воплощающих полученную модель в их архитектуре, позволяющий обеспечить высокий уровень их функциональной адаптивности. Областью применения метода являются сложные процедуры обработки пространственных данных и данных дистанционного зондирования Земли (ДЗЗ) в составе геоинформационных систем и технологий, а также систем с элементами пространственного анализа.

Целевым состоянием применения метода является процедура, состоящая из множества небольших блоков (объектов-спецификаций), скрытое состояние которых после создания остается неизменным (*англ.* immutable), при этом каждый из них отвечает за отдельный аспект всей процедуры и функционирует в рамках одного уровня абстракций путем диспетчеризации выполнения вложенных блоков, переданных ему в качестве аргументов, и предназначается для аналогичной передачи его в другие блоки, реализуя и эксплуатируя, таким образом, инъекционную адаптивность (A.INJECT) [18].

Объекты-спецификации предназначены для функционирования в составе структуры, являющейся исполнимой спецификацией — одновременно семантической моделью и программной реализацией требуемой специализированной версии процедуры. Каждый уровень такой иерархии последовательно уточняет функционирование предыдущего.

Такая структура может быть построена явно — пользователем (при наличии пользовательского интерфейса либо DSL) или программистом (с помощью API) — либо косвенно, с помощью вспомогательных инструментальных средств. Она может иметь произвольное количество встроенных в нее путей выполнения

22 Evans E., Fowler M. Specifications. [Электронный ресурс]. Режим доступа: https://www.martinfowler.com/apsupp/spec.pdf (дата обращения: 14.09.2024).

23 Bayer M. SQLAlchemy // The Architecture of Open-Source Applications. Vol. II: Structure, Scale, and a Few More Fearless Hacks / Ed. by A. Brown, G. Wilson. Mountain View, 2011. 390 p.

24 Sobel J.M., Friedman D.P. An Introduction to Reflection-Oriented Programming // Proceedings of Reflection’96. San Francisco: Springer Verlag, 1996. P. 107–126.

25 Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley, 2003. 359 p.; Evans E. Domain-Driven Design Reference: Definitions and Pattern Summaries. Indianapolis: Dog Ear Pub Llc, 2014. 75 p.

и допускать создание произвольного количества сторонних механизмов, обеспечивающих альтернативные пути ее выполнения, не предусмотренные изначальной реализацией. Каждый путь выполнения может реализовывать отдельную задачу или ее часть, например:

- выполнение основной части целевой задачи;
- подготовку к выполнению или очистку среды выполнения после выполнения;
- загрузку необходимых данных из внешних источников;
- формирование человекочитаемого представления внутренней структуры;
- генерацию программного кода и др.

Каждое обращение к любому из путей выполнения должно быть полностью независимым и не предполагать изменения состояния любого из объектов в составе структуры-спецификации вплоть до того, что должно быть возможно выполнение нескольких параллельных процессов, основанных на одних объектах-спецификациях. Все временные, промежуточные и окончательные данные должны быть сохранены в обрабатываемых объектах или в связанном с ними хранилище, передаваемом на вход нужного пути выполнения. Разработчик реализаций обрабатываемых объектов ответственен за создание абстракций, обеспечивающих эффективный доступ к внешним данным.

Процесс применения метода заключается в итеративном взаимном уточнении и приближении семантической модели предметной области и ее структуры-спецификации. Такой процесс позволяет найти «естественные» для данной предметной области границы вариативности, характеризующиеся, как правило, минимальным количеством данных, пересекающих такие границы.

Эмпирическим критерием корректности применения метода является близость работы со структурой-спецификацией к работе со специализированным для данной задачи DSL: полный контроль над структурой процесса при отсутствии (минимально необходимом количестве) элементов, не несущих предметного смысла. Другим признаком корректности является возможность интерпретировать отдельную версию структуры-спецификации как технического задания на разработку специализированного программного решения: достаточно конкретного, чтобы быть исполнимым, но достаточно абстрактного, чтобы не содержать избыточности и вариативных деталей реализации.

Такой подход к проектированию позволяет повысить адаптивность получаемого решения за счет кратного увеличения количества вариантов реализации каждой потенциальной модификации, среди которых путем итеративного укрупнения области вмешательства можно выбрать оптимальный, обеспечивающий необходимый баланс между затратами на разработку и качеством полученного результата.

2.3 Подходы и приемы применения метода поведенческой декомпозиции

В предыдущем разделе описана концептуальная схема построения архитектуры программных решений по методу поведенческой декомпозиции: определены ключевые сущности процесса и требования к ним. Однако получение оптимальной декомпозиции предметной области в объекты-спецификации, учитывающей потенциальные потребности в расширении и уточнении разработанного решения, является сложной исследовательской задачей с элементами инженерно-технического творчества. В процессе решения данной задачи происходит множество итераций, на которых уточняется как модель предметной области, так и ее воплощение в программной реализации.

В этой связи автором разрабатывается ряд взаимодополняющих подходов и приемов, в некотором смысле аналогичный ТРИЗ [24], для метода поведенческой декомпозиции. В данной работе приведены основные из них.

Восходящий подход к проектированию на основе анализа предметной области предполагает итеративное наращивание модели предметной области,

начиная с наиболее очевидных, компактных и простых в понимании и реализации блоков, а также блоков, являющихся тривиальными адаптерами для взаимодействия с внешними системами, с постепенным переходом к блокам, комбинирующим базовые и управляющим ими.

Нисходящий подход предполагает противоположное: итеративную детализацию модели предметной области от наиболее общих концепций к их детализации.

На практике часто имеет смысл одновременное или попеременное применение восходящего и нисходящего подходов.

Следующие приемы применимы на любом этапе проектирования (порядок условный):

- выделить часть, для которой очевидно существование множества взаимоисключающих версий (вариантов реализации);
- выделить часть, изменение которой наиболее вероятно, даже если альтернативные версии неочевидны;
- выделить части, взаимодействующие с внешними системами;
- выделить части, взаимодействующие со специфическими внутренними механизмами программного решения;
- выделить части, отвечающие за преобразование представлений данных;
- выделить наиболее простую или сложную часть;
- выделить наиболее изолированную (просто выделяемую) часть;
- выделить часть, минимально обменивающуюся данными со своим окружением;
- выделить части, общие для нескольких дочерних;
- выделить части, отражающие наиболее характерные проблемы предметной области;
- разделить части в соответствии с делением, принятым в предметной области;
- разделить специализированную часть на обобщенную и ее специализацию;
- разделить части, в которых решения принимаются, и части, в которых они выполняются;
- выделить часть, передающую данные между частями, принимающими решения, и частями, реализующими их;
- разделить части по характеру используемых в них данных или их представлений;
- разделить части, имеющие различный жизненный цикл;
- выделить части или пути выполнения, ответственные за инициализацию или финализацию процесса;
- выделить части, результаты которых могут быть кешированы;
- выделить части, результаты которых могут быть использованы несколькими потребителями в составе структуры-спецификации;
- разработать блок, кеширующий результаты вложенного в него блока;
- разработать объект контекста, обеспечивающий обмен данными между частями структуры-спецификации;
- поменять местами части на соседних уровнях иерархии;
- объединить части, изменение которых по отдельности невозможно;
- объединить модели нескольких родственных процедур в одну.

2.4 Производительность программных решений, построенных по методу поведенческой декомпозиции

Метод поведенческой декомпозиции не специфицирует конкретный набор технологий или конструкций языка программирования, на базе которых должна быть реализована разработанная модель. Как правило, разработчик или исследователь заинтересован в получении наиболее простой и гибкой реализации в рамках доступных ему инструментальных средств. Например, в рамках ООП такая реализация будет полагаться на динамическую компоновку объектов во время выполнения и динамическую диспетчеризацию обращений к их методам.

Для подходов, фактически основанных на различных версиях динамической диспетчеризации, особенно в динамических языках программирования, таких как Python, свойственно наиболее острое проявление проблем с производительностью.

Согласно опыту автора, в основной массе практически значимых случаев потери производительности не существенны, частично компенсируются за счет повышения скорости разработки либо могут быть компенсированы за счет простых приемов, таких как кеширование или локальная оптимизация отдельных блоков без изменения общей структуры. Однако, если производительность оказывается недостаточной, метод поведенческой декомпозиции оставляет множество возможностей для получения высокопроизводительных реализаций. В экспериментах автора особенно зарекомендовали себя три из них.

Подход первый — обобщенное программирование
Данный подход актуален для языков программирования с поддержкой обобщенного программирования и развитыми оптимизирующими компиляторами, таких как C++ и Rust. Автоматическая мономорфизация шаблонов типов, методов и функций позволяет избавиться от потерь производительности, связанных с динамической диспетчеризацией, но и дает возможность компилятору совместно оптимизировать части кода, относящиеся к различным компонентам. Очевидным недостатком этого подхода является невозможность конфигурирования системы во время выполнения.

Подход второй — метапрограммирование
Этот подход включает в себя все множество вариантов генерации высокопроизводительной реализации на основе структуры-спецификации в режимах JIT (*англ.* Just-in-time) или AOT (*англ.* Ahead-of-time). В современных реалиях особый интерес представляет генерация специализированной реализации для последующего выполнения на GPGPU, в том числе с помощью специализированных средств, таких как уже упомянутые Triton и XLA[26]. Не теряет актуальности генерация кода на C, C++, Rust или LLVM IR для последующей обработки оптимизирующим компилятором.

Подход третий — автоматическое распараллеливание
Данный подход предполагает генерацию графа элементарных операций на основе структуры-спецификации, которая далее передается в систему, обеспечивающую их параллельное выполнение. Примером такой системы может быть библиотека dask для Python [25].

2.5 Специализированная нотация метода поведенческой декомпозиции

Поведенческая декомпозиция может использоваться для проектирования и моделирования программных решений задолго до их фактической реализации. Эффективность этого процесса может быть существенно увеличена за счет инструментального сопровождения, упрощающего и унифицирующего процесс проектирования.

В качестве такого инструментального сопровождения автором разрабатывается специализированная человеко-машиночитаемая нотация, основанная на характерном для языков C, C++, Java, Rust, JavaScript обозначении блоков с помощью фигурных скобок и окончании инструкций с помощью символа «;». Она содержит три основных инструкции.

26 Bradbury J., Frostig R., Hawkins P., et al. JAX: Autograd and XLA // Astrophysics Source Code Library. 2021. [Электронный ресурс]. Режим доступа: https://ascl.net/2111.002 (дата обращения: 15.12.2024).

1. *Реализация* — `impl NAME [TYPE] [DETAILS]` — вводит некоторый тип объектов-спецификаций.
2. *Слот* — `slot [optional] [multiple] NAME [TYPE] [DETAILS]` — обозначает вакантное место, которое может быть занято одним из объектов-спецификаций. Соответствует точке применения инъекционной адаптивности (A.INJECT).
3. *Параметр* — `param NAME [TYPE] [DETAILS]` — обозначает возможность настройки объекта-спецификации за счет передачи ему параметра. Соответствует точке применения параметрической адаптивности (A.PARAM).

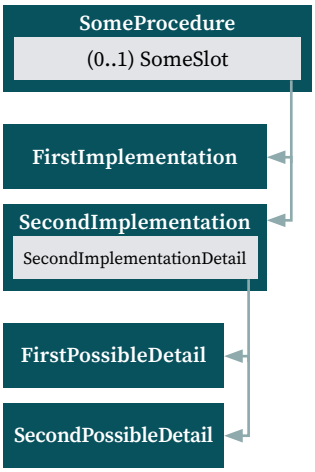
В этих определениях `NAME` и `TYPE` — произвольные идентификаторы, а `DETAILS` — блок вложенных определений, заключенный в фигурные скобки; `optional` (слот может оставаться пустым) и `multiple` (слот может содержать более одной реализации) в различных комбинациях позволяют определить допустимое количество реализаций для слота.

Нотация также предполагает возможность определения методов (точек входа), комментарии, простую систему типов, основанную на типажах, и поддержку пользовательских расширений. Эти возможности не будут детально рассмотрены в предлагаемой работе в связи с ограничениями ее объема.

Данная нотация подразумевает работу в режиме постепенного итеративного уточнения, что предполагает постепенное наращивание объема указанных деталей, а также работоспособность при любом уровне детализации. Она является основой для построения инструментальных средств генерации программного кода, документации и другой автоматизации. Модель условной процедуры, записанная с применением предложенной нотации, визуализированная с помощью автоматизированного инструмента на базе GraphViz [26], представлена на рис. 1.

Рис. 1
Модель условной процедуры, записанная с применением предложенной нотации, визуализированная с помощью автоматизированного инструмента на базе GraphViz

Fig. 1
A model of a conditional procedure written using the proposed notation, visualized using an automated tool based on GraphViz



Примечание. Оформление переработано редакцией.

3 Результаты

Для демонстрации работы метода предлагается рассмотреть два характерных примера.

Первый пример
Свертка двухмерного растра на регулярной сетке. Данная операция является известной и хорошо изученной, при этом в задачах геоинформатики, в частности обработки данных ДЗЗ или геополей, имеет множество аспектов, не характерных для других областей применения.

Второй пример
Метод семплирования и аугментации спутниковых изображений SEGA, применяемый для преодоления дефицита обучающих данных, а также площадного и количественного дисбаланса классов в обучающих и валидационных выборках, используемых при обучении искусственных нейронный сетей. Являясь авторской разработкой, метод изначально создавался с применением метода поведенческой декомпозиции и эволюционировал параллельно с ним. Применение поведенческой декомпозиции позволило объединить множество ситуативных решений в единый универсальный аппарат, активно используемый во множестве производственных и исследовательских задач.

Ввиду ограниченного объема публикации исходные описания с использованием предложенной человеко-машиночитаемой нотации, а также прочие материалы представлены по ссылке[27].

27 Гвоздев О.Г. Публикации по поведенческой декомпозиции. [Электронный ресурс]. Режим доступа: http://pub.almhq.org/publications/2024-behavioral-decomposition (дата обращения: 10.10.2024).

3.1 Пример 1. Свертка двухмерного растра на регулярной сетке

Операция свертки двухмерного растра на регулярной сетке в большинстве программного обеспечения общего назначения, осуществляющего обработку растровой графики, представлена рядом специализированных фильтров, таких как:

- размытие по Гауссу;
- определение границ;
- нерезкое маскирование (*англ.* unsharp masking);
- медианный фильтр.

В отдельных случаях присутствует возможность явного задания коэффициентов ядра свертки, на чем вариативность, как правило, заканчивается.

В задачах геоинформатики, в частности обработки данных ДЗЗ или геополей, проявляется множество особенностей данной предметной области, приводящих к необходимости построения значительно более сложных процедур свертки.

В доминирующих на рынке ГИС данные процедуры реализованы в виде монолитных блоков с набором предопределенных параметров[28,29,30], что покрывает основные сценарии использования, но далеко не все из них. Особый практический интерес представляют следующие:

1. Необходимость сохранения типа данных, общего диапазона и зарезервированных значений. Так, в спутниковых изображениях отдельные значения могут иметь зарезервированную семантику и должны обрабатываться особым способом. При этом общий диапазон значений может не соответствовать типу данных. Например, в представлении с помощью 16-битных беззнаковых целых значение 0 может означать отсутствие данных, значения [1; 1000] — интенсивность, а остальные значения считаться недопустимыми.
2. Помимо вышеупомянутого, может потребоваться дополнительное маскирование (исключение из вычислений) используемых значений.
3. Возможно множество подходов к обработке краевых артефактов как путем исключения краевых значений, так и с помощью добавления синтетических данных в них — паддинга (*англ.* padding).
4. В качестве ядра свертки может выступать не только функция, выраженная в виде набора коэффициентов, но и агрегирующая функция (min, max, sum), вычисление медианы или в общем случае квантилей и т. п.
5. Свертка может осуществляться с произвольным шагом скользящего окна (*англ.* striding) и произвольным шагом между элементами скользящего окна (*англ.* dilatation).

Все эти и другие особенности могут встречаться в любых комбинациях.

Применение метода и нотации поведенческой декомпозиции позволило учесть все перечисленные аспекты и получить следующую модель процедуры свертки (рис. 2).

28 Convolution, ArcGIS Pro / Esri. [Электронный ресурс]. Режим доступа: https://pro.arcgis.com/en/pro-app/latest/arcpy/spatial-analyst/convolution.htm (дата обращения: 05.10.2024).

29 R.resamp.filter, GRASS GIS. [Электронный ресурс]. Режим доступа: https://grass.osgeo.org/grass78/manuals/r.resamp.filter.html (дата обращения: 05.10.2024).

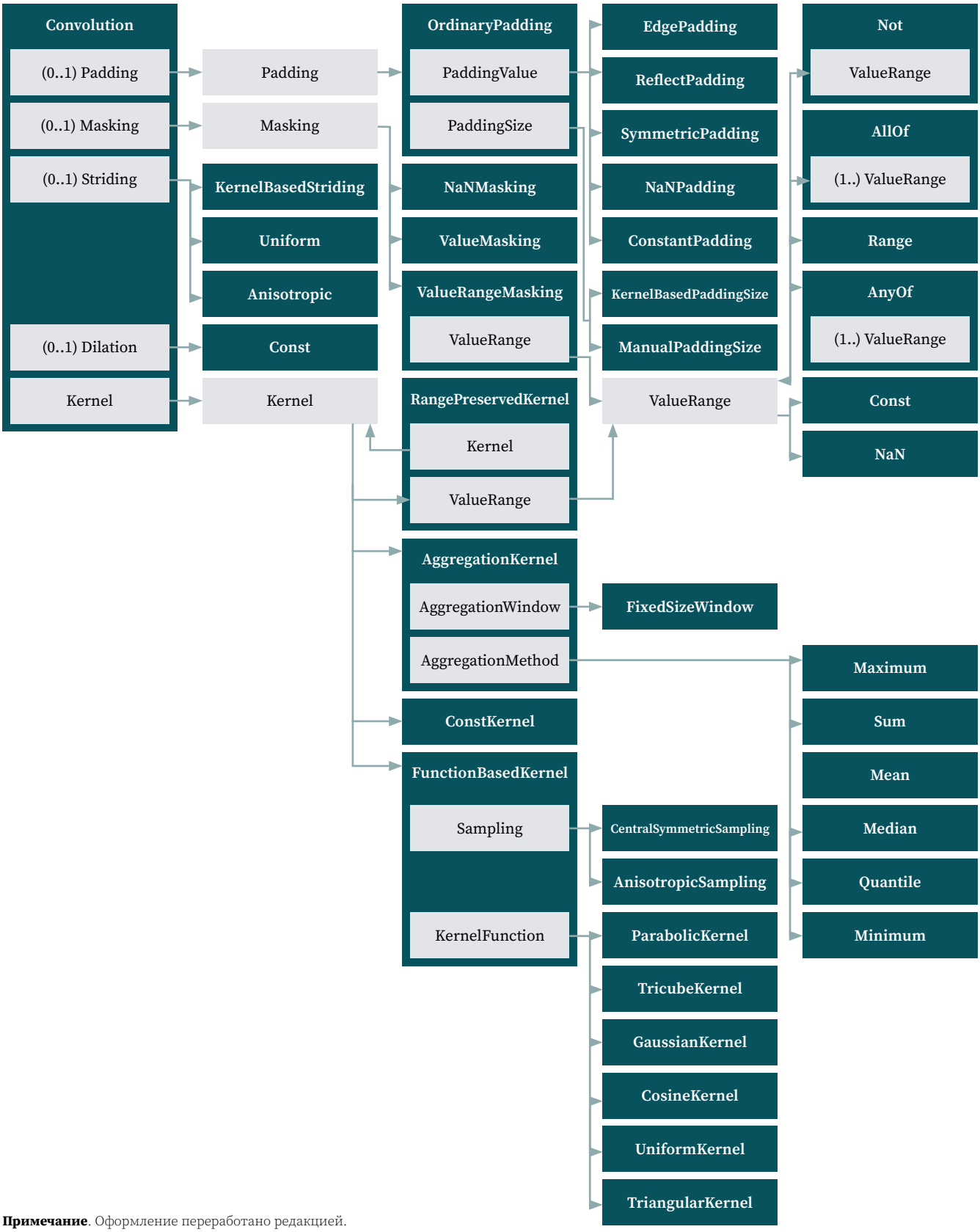
30 Convolution and Morphology Filters / L3Harris Technologies. [Электронный ресурс]. Режим доступа: https://www.l3harrisgeospatial.com/docs/convolutionmorphologyfilters.html (дата обращения: 05.10.2024).

Рис. 2

Модель процедуры свертки двухмерного растра на регулярной сетке, полученная с помощью метода и нотации поведенческой декомпозиции и визуализированная с помощью автоматизированного инструмента на базе GraphViz

Fig. 2

A model of the procedure of convolution of a two-dimensional image on a regular grid, obtained using the Behavioral Decomposition method and notation and visualized using an automated tool based on GraphViz



Примечание. Оформление переработано редакцией.

3.2 Пример 2. Метод семплирования и аугментации спутниковых изображений SEGA

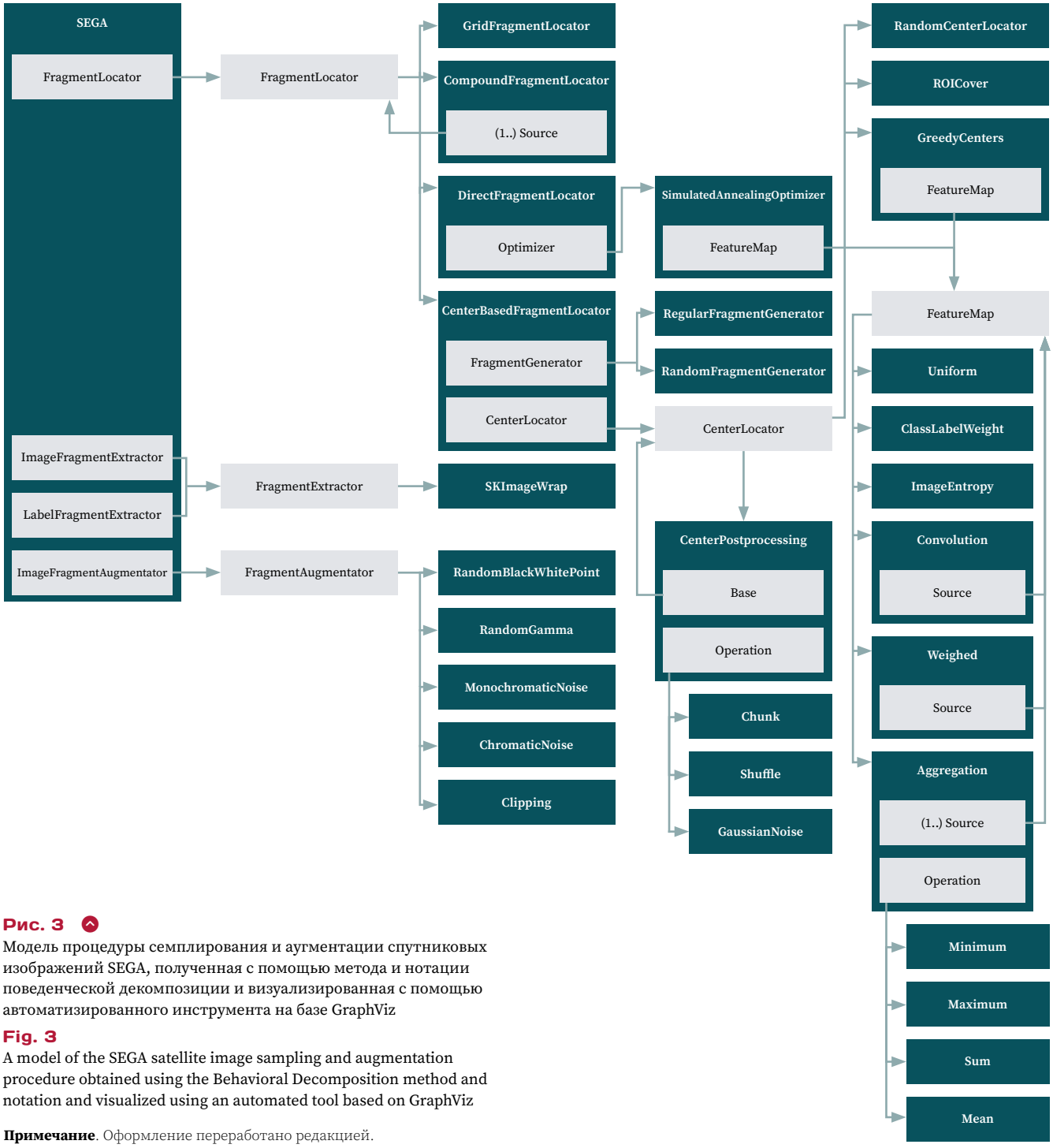


Рис. 3
Модель процедуры семплирования и аугментации спутниковых изображений SEGA, полученная с помощью метода и нотации поведенческой декомпозиции и визуализированная с помощью автоматизированного инструмента на базе GraphViz

Fig. 3
A model of the SEGA satellite image sampling and augmentation procedure obtained using the Behavioral Decomposition method and notation and visualized using an automated tool based on GraphViz

Примечание. Оформление переработано редакцией.

Метод семплирования и аугментации спутниковых изображений SEGA (рис. 3) используется для преодоления дефицита обучающих данных, а также площадного и количественного дисбаланса классов в обучающих и валидационных выборках, применяемых при обучении искусственных нейронных сетей. Его полное описание и экспериментальное исследование приведено в [27].

Суть метода состоит в определении множества четырехугольников, покрывающих изображение, их извлечении одновременно с геометрической аугментацией

и последующим выполнением цветовой аугментации. Множество четырехугольников может быть определено непосредственно, с помощью оптимизационного алгоритма, обеспечивающего покрытие изображения, пропорционально его карте значимости, определяемой для каждой области применения индивидуально, либо приближенным методом, менее точным, но более оптимальным с вычислительной точки зрения способом, заключающимся в случайной генерации четырехугольников вокруг набора центров, определяемых по следующим алгоритмам:

- регулярное покрытие;
- покрытие областей интереса (размеченных объектов) с заданным шагом (ROICover);
- покрытие карты значимости (GreedyCenters [28]).

Применение метода и нотации поведенческой декомпозиции позволило учесть все перечисленные аспекты.

4 Обсуждение

Поведенческая декомпозиция — специализированный метод проектирования и моделирования сложных адаптивных процедур обработки больших объемов относительно однородных данных. Обширный источник таких задач — предметная область геоинформатики, обработки пространственных данных и данных ДЗЗ. Рассматриваемый метод является теоретическим обобщением приемов программной инженерии, выработанных в процессе выполнения множества прикладных и исследовательских проектов.

Практическое применение данного метода многократно подтверждало высокую адаптивность получаемой архитектуры программных решений и высокую практическую полезность этой адаптивности. Но, что более важно, применение предложенного метода систематически позволяло выявлять новые подходы к решению исследуемых проблем.

Технические возможности, доступные специалистам в области программной инженерии на сегодняшний день, невероятно широки. В этой связи рационализация разработки программных систем часто достигается не благодаря созданию новых возможностей и технологий, а благодаря поиску и формализации подмножеств возможностей, использование которых сокращает область поиска решений, особенно на ранних этапах проектирования систем, при этом не внося существенных ограничений для достижимого результата и не оказывая негативного влияния на производительность труда. Предлагаемый метод можно рассматривать в этом ключе как формализацию подмножеств возможностей, позволяющую систематически получать программные решения, обладающие высокой степенью адаптивности.

С другой стороны, о предлагаемом методе можно говорить как о расширении концепции Domain-Driven Development (DDD)[31] на частный случай сложных процедур обработки данных.

Практика также показала ряд сложностей и ограничений, связанных с применением поведенческой декомпозиции. Получаемая структура-спецификация не позволяет обеспечить однородность шаблонов доступа к внешним данным или ресурсам (входным, выходным и промежуточным). Это требует от разработчика создания гибких абстракций, обеспечивающих оптимальный доступ к ним.

Метод требует точного понимания высокоуровневой структуры предметной области, в частности наличия моделей сущностей этой области, проектирование

31 Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston: Addison-Wesley, 2003. 359 p.; Evans E. Domain-Driven Design Reference: Definitions and Pattern Summaries. Indianapolis: Dog Ear Pub Llc, 2014. 75 p.

которых выходит за рамки данного метода. Его применение для слабо изученных предметных областей может оказаться малоэффективным и (или) привести к неприменимым на практике результатам.

В большинстве случаев накладные расходы на выполнение процедуры, построенной по предложенному методу, будут соизмеримы с расходами на динамическую диспетчеризацию методов конкретного языка программирования. Однако в отдельных случаях это может привести к существенной деградации производительности.

Если производительность и адаптивность важны в равной степени, возможно совмещение предложенного метода с методами метапрограммирования, что предполагает генерацию низкоуровневой специализированной реализации с последующей компиляцией и оптимизацией либо задействование других преобразований, повышающих производительность. Несмотря на принципиальную реализуемость этих подходов, они существенно усложняют построение и сопровождение системы.

Количественная оценка любого метода программной инженерии — сложная задача, требующая разработки формальных методов и множества наблюдений. Исследование этого аспекта начаты, однако строгие определяющие результаты пока не получены.

5 Выводы

Все больше аспектов жизни начинают зависеть от программного обеспечения и программно-определяемых технологий. И тем важнее становится скорость адаптации программных решений к быстро меняющимся реалиям и новым задачам.

В данной работе предложен метод поведенческой декомпозиции, являющейся попыткой подойти к решению проблемы создания адаптивных программных решений, архитектура которых учитывает возможности их приспособления к новым задачам и обстоятельствам функционирования до того, как они станут известны.

Можно утверждать, что предлагаемое исследование является вкладом в следующий виток развития программной инженерии, на который она вышла благодаря применению ее достижений к ее собственным практикам — программно-определяемому программному обеспечению (*англ.* software-defined software), самосборке программного обеспечения [29] или программированию высшего порядка (по аналогии с функциями высших порядков), когда одни программы не просто помогают разрабатывать другие (как компиляторы, IDE и др.), а определяют создание других. Развитие этих подходов в совокупности с последними достижениями в области искусственного интеллекта может позволить расширить достижимые границы сложности задач, решаемых с помощью программных систем.

БЛАГОДАРНОСТИ

Автор будет благодарен за любые сведения относительно опыта применения результатов данной работы, а также выражает готовность оказать содействие всем заинтересованным в практическом применении предложенного метода.

Результаты получены в рамках государственного задания Министерства науки и высшего образования Российской Федерации (№ FSFE-2022-0002).

БИБЛИОГРАФИЯ

1. Roozbeh A., Soares J., Maguire G.Q., et al. Software-Defined “Hardware” Infrastructures: A Survey on Enabling Technologies and Open Research Directions // IEEE Communications Surveys Tutorials. 2018. Vol. 20. No. 3. P. 2454–2485. DOI:10.1109/COMST.2018.2834731.
2. Weil S.A., Brandt S.A., Miller E.L., et al. A scalable, high-performance distributed file system // Proceedings of the 7th symposium on Operating systems design and implementation. Berkeley: USENIX Association, 2006. P. 307–320.

3. Goel A., Gupta S.C. Software defined storage technology // 2015 Asia-Pacific Software Engineering Conference. New Delhi, 2015. P. 6–7. DOI:10.1109/APSEC.2015.62.
4. Masoudi R., Ghaffari A. Software defined networks: A survey // Journal of Network and Computer Applications. 2016. Vol. 67. P. 1–25. DOI:10.1016/j.jnca.2016.03.016.
5. Яковлев В.В., Беркинбаева Ж.М. Основные отличия между традиционными и программно-конфигурируемыми сетями // Интеллектуальные технологии на транспорте. 2018. № 3. С. 5–11.
6. Dillinger M., Madani K., Alonistioti N. Software Defined Radio: Architectures, Systems and Functions. New York: Wiley & Sons, 2003. 454 p.
7. Сивоконь В.П., Лапшов Д.В. Технология software defined radio в задачах контроля радиошумов // Вестник Камчатского государственного технического университета. 2021. № 58. С. 17–28. DOI:10.17217/2079-0333-2021-58-17-28.
8. Lasi H., Fettke P., Kemper H.G., et al. Industry 4.0 // Business & Information Systems Engineering. 2014. Vol. 6. No. 4. P. 239–242. DOI:10.1007/s12599-014-0334-4.
9. Nayak N.G., Dürr F., Rothermel K. Software-defined environment for reconfigurable manufacturing systems // 5th International Conference on the Internet of Things (IOT). Seoul, 2015. P. 122–129. DOI:10.1109/IOT.2015.7356556.
10. Sadeghi M., Mahmoudi A., Deng X. Adopting distributed ledger technology for the sustainable construction industry: Evaluating the barriers using ordinal priority approach // Environmental Science and Pollution Research. 2021. Vol. 29. P. 10495–10520. DOI:10.1007/s11356-021-16376-y.
11. Дюдикова Е.И., Куницына Н.Н. Распределенные реестры в цифровой экономике: база данных, технология или протокол? // Инновации. 2019. № 9. С. 98–106. DOI:10.26310/2071-3010.2019.251.9.015.
12. Taherdoost H. Smart Contracts in Blockchain Technology: A Critical Review // Information. 2023. Vol. 14. No. 117. P. 1–19. DOI:10.3390/info14020117.
13. Митрофанова И.А. Законодательное регулирование «умных» контрактов: проблемы и перспективы развития // Правовая парадигма. 2018. Т. 17. № 3. С. 22–29. DOI:10.15688/lc.jvolsu.2018.4.3.
14. Гребенник О.Г., Иваницкий А.В. Умные контракты и перспектива их использования // Теория и практика современной науки. 2018. Т. 1. № 31. С. 640–642.
15. Qian L., Luo Z., Du Y., et al. Cloud computing: An overview // IEEE International Conference on Cloud Computing. Beijing, 2009. P. 626–631. DOI:10.1007/978-3-642-10665-1_63.
16. Allam A., Razak A., Mohamed H. User experience: challenges and opportunities // Journal of Information Systems Research and Innovation. 2013. Vol. 3. P. 28–36.
17. Naveed H., Khan A.U., Qiu S., et al. A comprehensive overview of large language models // arXiv. 2024. arXiv:2307.06435. [Электронный ресурс]. Режим доступа: https://arxiv.org/pdf/2307.06435 (дата обращения: 15.12.2024).
18. Гвоздев О.Г. Современные геоинформационные технологии: адаптивность, адаптируемость, расширяемость, функциональная масштабируемость // Известия вузов «Геодезия и аэрофотосъемка». 2022. Т. 66. № 5. С. 28–46. DOI:10.30533/0536-101X-2022-66-5-28-46.
19. Bezanson J., Edelman A., Karpinski S., et al. Julia: A fresh approach to numerical computing // SIAM Review. 2017. Vol. 59. No. 1. P. 65–98. DOI:10.1137/141000671.
20. Lattner C., Adve V.S. LLVM: A compilation framework for lifelong program analysis & transformation // Proceedings of the International symposium on Code generation and optimization: feedback-directed and runtime optimization. San Jose, 2004. P. 75–86. DOI:10.1109/CGO.2004.1281665.
21. Paszke A., Gross S., Chintala S., et al. Automatic differentiation in PyTorch // Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, 2017. P. 1–4.
22. Christopoulou A., Giakoumakis E.A., Zafeiris V.E., et al. Automated refactoring to the Strategy design pattern // Information and Software Technology. 2012. Vol. 54. P. 1202–1214. DOI:10.1016/j.infsof.2012.05.004.
23. Bala R., Kaswan K.K. Strategy Design Pattern // International Journal of Science and Research (IJSR). 2014. Vol. 3. No. 8. P. 385–387.

24. Ilevbare I.M., Probert D., Phaal R. A review of TRIZ, and its benefits and challenges in practice // Technovation. 2013. Vol. 33. P. 30–37. DOI:10.1016/j.technovation.2012.11.003.
25. Rocklin M. Dask: Parallel computation with blocked algorithms and task scheduling // Proceedings of the 14th Python in Science Conference. Austin, 2015. P. 130–136. DOI:10.25080/Majora-7b98e3ed-013.
26. Ellson J., Gansner E. R., Koutsofios E., et al. Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools // Jünger M., Mutzel P. (eds). Graph Drawing Software. Mathematics and Visualization. Berlin; Heidelberg: Springer, 2004. P. 127–148. DOI:10.1007/978-3-642-18638-7_6.
27. Гвоздев О.Г., Матерухин А.В., Майоров А.А. Экспериментальное исследование метода семплирования и аугментации спутниковых изображений SEGA // Известия вузов «Геодезия и аэрофотосъемка». 2022. Т. 66. № 5. С. 47–59. DOI:10.30533/0536-101X-2022-66-5-47-59.
28. Gvozdev O. GreedyCenters: Satellite imagery adaptive sampling method for artificial neural networks training // E3S Web of Conferences. 2021. Vol. 310. P. 02001. DOI:10.1051/e3sconf/202131002001.
29. Гурьянов В.И. Самосборка программного обеспечения как паттерн проектирования // Программные продукты и системы. 2008. № 1. С. 62–64.

АВТОР **Гвоздев Олег Геннадьевич**
ФГБОУ ВО «Московский государственный университет геодезии и картографии» (МИИГАиК), Москва, Россия
кафедра информационно-измерительных систем,
факультет геоинформатики и информационной безопасности
канд. техн. наук
0000-0002-1917-3206

Поступила 09.01.2025. Принята к публикации 23.06.2025. Опубликована 30.06.2025.

UDC 004.9
DOI:10.30533/GiA-2025-022

Behavior decomposition: adaptive geoinformational technologies design method

Oleg G. Gvozdev[1] ✉

[1] Moscow State University of Geodesy and Cartography, Moscow, Russia
✉ gvozdev@miigaik.ru

CITATION Gvozdev OG. Behavior decomposition: adaptive geoinformational technologies design method. *Izvestia vuzov. Geodesy and Aerophotosurveying*. 2025;69(3): 74–93. DOI:10.30533/GiA-2025-022.

KEYWORDS behavior decomposition, adaptive geoinformational technologies, software engineering, software architecture

ABSTRACT The paper considers the problem of development adaptive informational and geoinformational systems. In the scope of this paper, adaptivity is defined as a property of the architecture of a software solution that supports or prevents the satisfaction of new requirements before they become known. Author proposes adaptive geoinformational technologies design method, called “Behavioral Decomposition”, focused on the development of individual high complexity procedures for spatial data processing. The theoretical basis and prior work for the emergence of the method, the underlying principles and the techniques for its application to practical tasks, human-machine-readable notation for describing solutions at any stage of design are given. As examples, models of two spatial data processing procedures, designed with the proposed method, are given. The development experience and the identified limitations of the method and ways to overcome them are given.

ACKNOWLEDGEMENTS The author would be grateful for any information on the experience of applying the results of this work, and also expresses his willingness to assist anyone interested in the implementation of the proposed method.
The results were obtained within the framework of the state assignment of the Ministry of Science and Higher Education of the Russian Federation (No. FSFE-2022-0002).

REFERENCES

1. Roozbeh A, Soares J, Maguire GQ, et al. Software-Defined “Hardware” Infrastructures: A Survey on Enabling Technologies and Open Research Directions. *IEEE Communications Surveys Tutorials.* 2018;20(3): 2454–2485. DOI:10.1109/COMST.2018.2834731.

2. Weil SA, Brandt SA, Miller EL, et al. A scalable, high-performance distributed file system. ***Proceedings of the 7th symposium on Operating systems design and implementation.*** Berkeley: USENIX Association; 2006: 307–320.
3. Goel A, Gupta SC. Software defined storage technology. ***2015 Asia-Pacific Software Engineering Conference.*** New Delhi; 2015: 6–7. DOI:10.1109/APSEC.2015.62.
4. Masoudi R, Ghaffari A. Software defined networks: A survey. ***Journal of Network and Computer Applications.*** 2016;67: 1–25. DOI:10.1016/j.jnca.2016.03.016.
5. Jakovlev VV, Berkinbaeva ZhM. Osnovnye otlichija mezhdu tradicionnymi i programmno-konfiguriruemymi setjami [Key differences between traditional and software-configurable networks]. ***Intellectual technologies on transport***. 2018;3: 5–11. (In Russian).
6. Dillinger M, Madani K, Alonistioti N. Software Defined Radio: Architectures, Systems and Functions. New York: Wiley & Sons; 2003. 454 p.
7. Sivokon' VP, Lapshov DV. Tehnologija software defined radio v zadachah kontrolja radioshumov [Software defined radio technology in radio noise control tasks]. ***Bulletin of Kamchatka State Technical University.*** 2021;58: 17–28. (In Russian). DOI:10.17217/2079-0333-2021-58-17-28.
8. Lasi H, Fettke P, Kemper HG, et al. Industry 4.0. ***Business & Information Systems Engineering.*** 2014;6(4): 239–242. DOI:10.1007/s12599-014-0334-4.
9. Nayak NG, Dürr F, Rothermel K. Software-defined environment for reconfigurable manufacturing systems. ***5th International Conference on the Internet of Things (IOT).*** Seoul; 2015: 122–129. DOI:10.1109/IOT.2015.7356556.
10. Sadeghi M, Mahmoudi A, Deng X. Adopting distributed ledger technology for the sustainable construction industry: Evaluating the barriers using ordinal priority approach. ***Environmental Science and Pollution Research.*** 2021;29: 10495–10520. DOI:10.1007/s11356-021-16376-y.
11. Djudikova EI, Kunicyna NN. Raspredelennye reestry v cifrovoj jekonomike: baza dannyh, tehnologija ili protokol? [Distributed registries in the digital economy: database, technology or protocol?]. ***Innovations.*** 2019;9: 98–106. (In Russian). DOI:10.26310/2071-3010.2019.251.9.015.
12. Taherdoost H. Smart Contracts in Blockchain Technology: A Critical Review. ***Information.*** 2023; 14(117): 1–19. DOI:10.3390/info14020117.
13. Mitrofanova IA. Zakonodatel'noe regulirovanie "umnyh" kontraktov: problemy i perspektivy razvitija [Legislative Regulation of Smart Contracts: Problems and Development Prospects]. ***Legal Concept.*** 2018;17(3): 22–29. (In Russian). DOI:10.15688/lc.jvolsu.2018.4.3.
14. Grebennik OG, Ivanickij AV. Umnye kontrakty i perspektiva ih ispol'zovanija [Smart contracts and prospects for their use]. ***Teorija i praktika sovremennoj nauki.*** 2018;1(31): 640–642. (In Russian).
15. Qian L, Luo Z, Du Y, et al. Cloud computing: An overview. ***IEEE International Conference on Cloud Computing.*** Beijing; 2009: 626–631. DOI:10.1007/978-3-642-10665-1_63.
16. Allam A, Razak A, Mohamed H. User experience: challenges and opportunities. ***Journal of Information Systems Research and Innovation.*** 2013;3: 28–36.
17. Naveed H, Khan AU, Qiu S, et al. A comprehensive overview of large language models. ***arXiv.*** 2024. arXiv:2307.06435. Available from: https://arxiv.org/pdf/2307.06435 (Accessed 15 December 2024).
18. Gvozdev OG. Sovremennye geoinformacionnye tehnologii: adaptivnost', adaptiruemost', rasshirjaemost', funkcional'naja masshtabiruemost' [Modern geoinformation technologies: adaptability, adaptability, extensibility, functional scalability]. ***Izvestiya vuzov. Geodesy and Aerophotosurveying.*** 2022;66(5): 28–46. (In Russian). DOI:10.30533/0536-101X-2022-66-5-28-46.
19. Bezanson J, Edelman A, Karpinski S, et al. Julia: A fresh approach to numerical computing. ***SIAM Review.*** 2017;59(1): 65–98. DOI:10.1137/141000671.
20. Lattner C, Adve VS. LLVM: A compilation framework for lifelong program analysis & transformation. ***Proceedings of the International symposium on Code generation and optimization: feedback-directed and runtime optimization***. San Jose; 2004: 75–86. DOI:10.1109/CGO.2004.1281665.
21. Paszke A, Gross S, Chintala S, et al. Automatic differentiation in PyTorch. ***Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017).*** Long Beach; 2017: 1–4.

22. Christopoulou A, Giakoumakis EA, Zafeiris VE, et al. Automated refactoring to the Strategy design pattern. ***Information and Software Technology.*** 2012;54: 1202–1214. DOI:10.1016/j.infsof.2012.05.004.
23. Bala R, Kaswan KK. Strategy Design Pattern. ***International Journal of Science and Research (IJSR).*** 2014;3(8): 385–387.
24. Ilevbare IM, Probert D, Phaal R. A review of TRIZ, and its benefits and challenges in practice. ***Technovation.*** 2013;33: 30–37. DOI:10.1016/j.technovation.2012.11.003.
25. Rocklin M. Dask: Parallel computation with blocked algorithms and task scheduling. ***Proceedings of the 14th Python in Science Conference. Austin***; 2015: 130–136. DOI:10.25080/Majora-7b98e3ed-013.
26. Ellson J, Gansner ER, Koutsofios E, et al. Graphviz and Dynagraph – Static and Dynamic Graph Drawing Tools. ***Jünger M., Mutzel P. (eds). Graph Drawing Software. Mathematics and Visualization.*** Berlin; Heidelberg: Springer; 2004: 127–148. DOI:10.1007/978-3-642-18638-7_6.
27. Gvozdev OG, Materuhin AV, Majorov AA. Jeksperimental'noe issledovanie metoda semplirovanija I augmentacii sputnikovyh izobrazhenij SEGA [Experimental study of SEGA satellite image sampling and augmentation method]. ***Izvestiya vuzov. Geodesy and Aerophotosurveying.*** 2022;66(5): 47–59. (In Russian). DOI:10.30533/0536-101X-2022-66-5-47-59.
28. Gvozdev OG. GreedyCenters: Satellite imagery adaptive sampling method for artificial neural networks training. ***E3S Web of Conferences.*** 2021;310: 02001. DOI:10.1051/e3sconf/202131002001.
29. Gur'janov VI. Samosborka programmnogo obespechenija kak pattern proektirovanija [Software self-assembly as a design pattern]. ***Software & Systems.*** 2008;1: 62–64. (In Russian).

AUTHOR **Oleg G. Gvozdev**
Moscow State University of Geodesy and Cartography, Moscow, Russia
Department of Information and Measurement Systems,
Faculty of Geoinformatics and Information Security
PhD in Engineering
0000-0002-1917-3206

Submitted: January 09, 2025. Accepted: June 23, 2025. Published: June 30, 2025.