К ВОПРОСУ МОДЕЛИРОВАНИЯ ПРОГРАММНЫХ СИСТЕМ

Диомидов И.А., аспирант,

ИрГУПС, г. Иркутск, Россия

Аннотация. В данной работе рассмотрен вопрос моделирования программных систем, их связь с формальными моделями. Цель настоящей работы - определить проблему моделирования программных систем и предложить пути ее решения. Для этого был рассмотрен исторический контекст, различные аспекты моделирования программного обеспечения.

Ключевые слова: программное обеспечение, системный анализ, алгоритм, система, компьютерные науки

Введение. Момент становления компьютерных наук можно определить с появлением понятия алгоритма, введённого великим философом и математиком Аль-Хорезми. Алгоритм — это строго определенная последовательность шагов, направленная на

решение конкретной задачи. Именно разбиение процесса на упорядоченные этапы — это основа любой компьютерной программы. Однако в реальном мире сложно сть процессов заключается в их глубокой взаимосвязанности, изменчивости и нелинейности. Для их анализа используется системный подход, предполагающий разбиение системы на элементы и изучение связей и взаимодействий между ними. Такой подход позволяет не только упростить анализ, но и раскрыть скрытые закономерности сложных систем. В целом, с увеличением сложности можно сказать, что постановка проблемы является специальной задачи, требующего отдельного рассмотрения [1]. Кроме этого, при работе с программными системами исследователь вынужден возвращаться к концепции алгоритма, ведь фактически именно алгоритм определяет сущность проходящих в программе процессов. Все эта вариативность порождает определенную путаницу и является источником сложности программных систем. С одной стороны, программа – это процесс, с другой стороны – система модулей.

Сущность проблемы. В инженерных областях часто наблюдается разделение на теоретические и прикладные аспекты. Теория служит инструментом для решения конкретных практических задач: например, расчет габаритов и формы крыла самолета или определения предельно допустимого веса товарного состава. Математика, как язык описания абстрактных зависимостей, лежит в основе такого разделения. Преобразуя определенные абстрактные структуры в конкретные законы, мы можем применятьих в различных отраслях и достигать удовлетворительных результатов будь то машиностроение, сельское хозяйство и так далее.

В программных системах ситуация аналогичная, но имеет определенную специфику. Так, теория множеств и теория категорий активно используются в качестве теоретической основы для моделирования информации. На их основе создана реляционная модель данных [2]. Теория алгоритмов опирается на понятия графов и конечных автоматов.

На основе данных концепцией разработано большие число различных прикладных технологий. Сегодня широко используются языки программирования, их фреймворки, интегрированные среды разработки, системы развертывания и системы машинного обучения. Многообразие этих технологий и их изменчивость привело к существенному расхождению между теорией и практикой. Сейчас любая программная система как правило является встраиваемой в определенный контекст: аппаратное устройство, операционная система, конкретный язык программирования и так далее.

При этом попытки подобрать общий теоретический аппарат или наиболее эффективную парадигму программирования, которая позволила бы рассматривать программу с точки зрения проблем и терминов предметной области, не привели к значимым результатам. Так, например, объектно-ориентированное программирование, при всех своих достоинствах, не смогла скрыть сложность программ и существенно снизить затраты на разработку. Данная проблема подробно описана в письме Фредерика Брукса «Серебряной пули - нет», где основной тезис заключался в том, что невозможно снизить сложность программ, из-за самой природы программного обеспечения. Несмотря на давность сообщения (от 1986 года), оно остается верным и по сей день [3].

Все еще теория и практика в программировании развиваются несколько раздельно. Например, подобная ситуация сложилась на уровне реляционных моделей и объектных, и известна как проблема соответствий реляционной и объектной модели (англ. object-relational impendance mismatch). Данная проблема была решена следующим образом: в настоящий момент разработаны ORM-инструменты, которые преобразуют реляционную модель в объектную.

Подобный подход наглядно иллюстрирует разницу между формальным подходом к моделированию систем и спецификой программных систем. В рамках них могут сосуществовать противоречивые модели, но за счет возможности преобразования (программа в том числе и процесс) между моделями, их вполне можно использовать совместно.

Анализ проблемы. Следует различать исходный код и экземпляр программы. Исходный код определяет алгоритм работы программной системы и является самостоятельной сущностью (системой). Скомпилированная программа, в свою очередь, представляет собой отдельный объект. Следовательно, при исследовании программы как объекта необходимо учитывать ее двойственную природу: описание процесса и его исполнение. В упрощенном представлении, исходный код можно рассматривать как концептуальную систему, а конкретную программу – как физическую реализацию этой системы. Аналогичная проблема существует в системном анализе и связана с вопросом построения общей теории систем, которая объединяла бы концептуальные и физические системы [4].

Специфика программной системы определяется многими факторами на различных уровнях: аппаратными ограничениями, ограничениями операционной системы или языка программирования. На работоспособность системы будут воздействовать версии используемых библиотек и совместимость их интерфейсов.

Зависимости между модулями можно представить в виде графов, а для зависимостей между типами данных (что в ряде языков программирования - синонимично) – использовать морфизмы из теории категорий.

Важным вопросом является вопрос соответствия модулей конкретным требованиям программной области. Во многом работа разработчика заключается в сборе и правильной верификации этих требований. Формальные модели могут применяться в том числе для представления требований в таком виде, при котором они будут понятны разработчику, например, на основе теории категорий [5].

В разработке программного обеспечения применяются своеобразные теоретические подходы, паттерны проектирования: наблюдатель, абстрактная фабрика, модель-контроллер-отображение и другие. Несмотря на широкое применение, эти паттерны не обладают достаточной теоретической базой. Их формирование происходило преимущественно на основе практического опыта, а выбор конкретного паттерна для данной ситуации во многом зависит от решения разработчика. В зависимости от разработчика, выбор конкретного архитектурного подхода может варьироваться.

Функциональная парадигма программирования, позволяющая описывать программы в терминах чистых функций, монад и функций высшего порядка, наиболее близка к формальным моделям (достаточно строго и хорошо описана), и при этом удобна для разработчиков. Она получила большую популярность в последнее время за счет того, что позволяет создавать эффективные и высокопроизводительные приложения с распределенными вычислениями. Различные функциональные модули можно легко выполнять в нескольких потоках. Тем не менее, даже в этом случае функция не полностью соответствует математическому представлению. Она привязана к определенному типу, занимает конкретное место в коде и может иметь несколько точек вызова.

Сам исходный код можно рассматривать с условно двух точек зрения: программной и доменной. Программный аспект охватывает модули, обеспечивающие общую функциональность программы: взаимодействие с другими системами, использование библиотек, реализацию параллельных вычислений и пользовательского интерфейса, применение паттернов проектирования и т.д.

Доменный аспект фокусируется на решении конкретных задач: вычислениях, алгоритмах и, в частности, традиционных формальных моделях (математические функции, имитационные модели, вероятностные модели процессов). Система типов и модель данных играют здесь ключевую роль.

Доменные модули представляют наибольший интерес, т.к. они вполне могут рассматриваться как самостоятельная модель. В программе могут быть определены модули, отражающие сущности предметной области, их атрибуты и вычисляемые атрибуты. Например, уже есть работы, где с помощью нейронных сетей выполняется преобразование исходного кода в конкретные выражения: что именно там выполняется [6].

Выводы. При построении модели программы все еще требуется комбинировать формальные и эвристические методы. Исходя из проведенного исследования, автором сделаны следующие выводы:

- Формальный подход хорошо подходят для моделирования программных аспектов (исходного кода в отрыве от предметной области). Так для понимания процессов и размещения модулей и зависимостей могут применяться модели потоков данных, конечных автоматов состояний и иных абстрактных математических структур.;

- Также традиционные формальные методы могут эффективно применяться для построения требований к системе (конкретного приложения): описание предикатов, которым должны удовлетворять система;

- Предметные (доменные) аспекты программы являются самостоятельной моделью и должны рассматриваться вместе с формальными методами;

Исходный код на конкретном языке программирования может рассматриваться как самостоятельная модель. Для повышения наглядности можно использовать инструменты интеллектуального анализа кода, в том числе с целью получения представления о том, какие правила предметной области могут применяться.

Заключение. В данной работе рассмотрен вопросы и проблемы моделирования программных систем. Моделирование программных систем остаётся сложной задачей из-за расхождения между абстрактной теорией и конкретной реализацией. Для полного моделирования программных систем формальные методы должны использоваться в совместно с эвристическими. Отмечено, что конкретная реализация системы (исходный код) сама по себе может рассматриваться как модель предметной области. Таким образом, одним из перспективных направлений дальнейших исследований является изучение вопроса интеграции программных аспектов в формальные модели, а также применение новых инструментов (искусственного интеллекта) для выполнения конвертации исходного кода в конкретные спецификации предметной области.

Литература

1. Денисов, А. А. Современные проблемы системного анализа: информационные основы : учеб. пособие для студентов вузов, обучающихся по специальностям направления "Систем. анализ и управление" / А. А. Денисов ; А. А. Денисов ; М-во образования Рос. Федерации, С.-Петерб. гос. политехн. ун-т. – СПб. : Изд-во СПбГПУ, 2003. – 275 с. – ISBN 5-7422-0454-X. – EDN QJNTYD.

2. Дейт, К. Д. Введение в системы баз данных / К. Д. Дейт ; К. Дж. Дейт ; [пер. с англ. и ред. К. А. Птицына]. – 8-е изд.. – Москва [и др.] : Вильямс, 2008. – ISBN 978-5-8459-0788-2. – EDN QMSTGX.

3. D. A. Grier, “There Is Still No Silver Bullet,” Computer, vol. 54, no. 2, pp. 60–62, Feb. 2021, doi: 10.1109/MC.2020.3042682.

4. Маторин, С. И. Учет общесистемных закономерностей при концептуальном моделировании понятийных знаний / С. И. Маторин, А. Г. Жихарев, В. В. Михелев // Искусственный интеллект и принятие решений. – 2019. – № 3. – С. 12-23. – DOI 10.14357/20718594190302. – EDN WYSFCB.

5. Антонов, В. В. Метод проектирования адаптивного программного комплекса на основе методологии категорийной формальной модели открытой предметной области / В. В. Антонов // Вестник Уфимского государственного авиационного технического университета. – 2015. – Т. 19, № 1(67). – С. 258–263. – EDN TPNUQL.

6. S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, “Summarizing Source Code using a Neural Attention Model,” in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany: Association for Computational Linguistics, 2016, pp. 2073–2083. doi: 10.18653/v1/P16-1195.