РОЛЬ UML В СОВРЕМЕННОМ ПРОЕКТИРОВАНИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Д.А. Бощенко, *студент*
Научный руководитель: *Т.П. Машихина, канд. пед. наук, доцент*
Волгоградский государственный университет
(Россия, г. Волгоград)

DOI:10.24412/2500-1000-2026-4-2-13-17

Аннотация. *В статье исследуется значение унифицированного языка визуального моделирования UML при создании современных информационных систем. Рассматриваются основные разновидности диаграммного аппарата UML, специфика их практического использования на последовательных этапах разработки, а также сильные и слабые стороны данной нотации. Приводится сопоставительный анализ UML с альтернативными языками описания архитектуры – IDEF и BPMN. Отдельное внимание уделяется вопросам встраивания UML в адаптивные («гибкие») методологии ведения проектов и той роли, которую этот язык играет в документировании сложных программных комплексов. Намечаются векторы эволюции UML с учётом актуальных проблем, стоящих перед проектировщиками информационных систем.*

Ключевые слова: *UML; унифицированный язык моделирования; архитектура информационных систем; диаграммы классов; визуальное проектирование; гибкие методологии разработки; CASE-инструменты; нотации описания бизнес-процессов.*

В эпоху цифровой трансформации и повсеместного внедрения сложных распределённых вычислительных систем вопросы качественного архитектурного проектирования выходят на первый план. Разработка крупномасштабных приложений, включающих десятки и сотни взаимодействующих компонентов, невозможна без создания наглядных абстрактных моделей, которые служат единым языком для всех стейкхолдеров: от заказчика и бизнес-аналитика до инженеров по тестированию и технических писателей. Унифицированный язык визуального моделирования UML (Unified Modeling Language) на протяжении более двух десятилетий сохраняет статус мирового стандарта в области объектно-ориентированного анализа и синтеза архитектурных решений. Он даёт разработчикам и архитекторам выразительный инструментарий для визуализации будущего программного продукта, специфицирования его компонентного состава и документирования всех значимых проектных решений.

По своей сути UML представляет собой графическую нотацию, абстрагированную от конкретных языков программирования, что позволяет описывать информационные системы в трёх ключевых разрезах: функциональном (что система делает), статическом (из каких элементов она состоит и как они связаны) и динамическом (как элементы взаимодействуют во времени). Фундаментом языка выступает метамодель, зафиксированная в спецификациях консорциума Object Management Group (OMG). Центральная роль UML в современном проектировании информационных систем заключается в формировании универсального средства коммуникации между различными профессиональными ролями. Аналитики обращаются к диаграммам прецедентов (use case diagrams) для формализации требований заказчика. Архитекторы строят диаграммы классов и компонентов, чтобы зафиксировать статическую структуру. Разработчики используют диаграммы последовательностей и состояний, чтобы детализировать алгоритмическое поведение объектов. Такое функциональное разделение позволяет эффективно управлять сложностью, фокусируя внимание команды на конкретном аспекте системы в рамках отдельной проекции.

Актуальная версия спецификации UML 2.x насчитывает 14 типов диаграмм, которые традиционно подразделяются на структурные и поведенческие. Структурные диаграммы (классов, компонентов, развёртывания, объектов, пакетов, композитной структуры) фиксируют статическую организацию системы –

иерархии классов, интерфейсы взаимодействия, топологию узлов, варианты компоновки артефактов. Поведенческие диаграммы (деятельности, конечных автоматов, вариантов использования, последовательности, коммуникации, синхронизации, обзора взаимодействий) отображают динамику: потоки управления, обмен сообщениями между объектами во временно́й шкале и переходы из одного состояния в другое под действием событий. Практика показывает, что наибольшей востребованностью пользуются диаграммы классов – они позволяют не только проектировать доменную модель предметной области, но и напрямую генерировать программный код на таких языках, как Java, C#, Python или C++. Не менее значимы диаграммы последовательности, без которых практически невозможно корректно спроектировать распределённые транзакции и протоколы взаимодействия в микросервисных архитектурах.

Ещё одной важной гранью роли UML выступает его вклад в управление требованиями и формализацию архитектурной документации. Диаграмма вариантов использования играет роль связующего звена между неформальными пожеланиями заказчика и строгими техническими спецификациями. Каждый прецедент (вариант использования) может быть детализирован с помощью текстового сценария, а затем развёрнут в последовательность диаграмм активности или взаимодействия. В русле концепции Architecture Description Standard (ADS) диаграммы UML часто комбинируются с таблицами решений и текстовыми аннотациями, формируя ядро архитектурного документа. В отсутствие подобного визуального остова сопровождение и развитие крупной системы превращается в крайне трудоёмкий процесс: инженерная команда быстро теряет понимание скрытых зависимостей между модулями, что неизбежно ведёт к накоплению технического долга и ошибкам при внесении изменений.

Вместе с тем у языка UML имеются серьёзные ограничения, которые необходимо учитывать при его практическом применении. Наиболее часто критикуемым недостатком считается избыточность – подавляющее большинство проектов задействует не более четырёх-пяти типов диаграмм (обычно это диаграммы классов, вариантов использования, последовательности и деятельности), тогда как остальные возможности языка остаются невостребованными. Эта особенность породила восприятие UML как «тяжеловесного» инструмента, требующего неоправданно высоких затрат на поддержание моделей в актуальном состоянии. К тому же классический UML слабо приспособлен для описания систем, функционирующих в сильновероятностной среде (например, на основе событийного программирования) или систем с интенсивными потоками данных – здесь более уместны специализированные нотации, такие как потоковые диаграммы данных (DFD) либо языки, ориентированные на моделирование потоков работ. Дополнительным сдерживающим фактором выступает отсутствие в стандартной спецификации формальных средств верификации моделей; корректность UML-модели целиком зависит от опыта и квалификации архитектора, что вносит субъективный фактор.

Если провести сравнительный анализ UML с другими распространёнными нотациями моделирования, можно увидеть, что единого «лучшего» инструмента не существует. Методологии функционального моделирования IDEF0 и информационного моделирования IDEF1X прекрасно подходят для описания бизнес-процессов верхнего уровня и потоков данных в организации, однако они не поддерживают объектно-ориентированный подход и принципы инкапсуляции, наследования и полиморфизма. Нотация BPMN (Business Process Model and Notation) заметно превосходит UML в части наглядного отображения бизнес-процессов для заказчиков – она интуитивно понятна даже неподготовленному пользователю. Однако BPMN уступает UML при проектировании непосредственно программной архитектуры, особенно когда речь идёт о сложных иерархиях классов или распределённых взаимодействиях. На практике всё чаще применяется гибридный стиль: на этапе бизнес-анализа и предпроектного обследования используют BPMN, а на этапе технического проектирования и кодирования – UML с акцентом на диаграммы классов и компонентов. Ни одна из существующих альтернатив не предлагает столь же полного, стандартизированного и расширяемого набора инструмен-

тов для сквозного объектно-ориентированного проектирования.

С приходом и широким распространением гибких (Agile) методологий, таких как Scrum, Kanban и XP, отношение профессионального сообщества к UML претерпело значительные изменения. Классический «водопадный» (каскадный) подход требовал создания исчерпывающей и детализированной модели «до написания первой строки кода» – так называемый подход «UML как чертёж» (UML as blueprint). В Agile-среде такой стиль признан избыточно тяжёлым. Сегодня доминирует подход «UML как набросок» (UML as sketch), при котором диаграммы создаются выборочно – исключительно для проработки наиболее сложных архитектурных решений, критических сценариев или спорных моментов, требующих коллективного обсуждения. Диаграммы быстро рисуются на доске или в лёгком инструменте, обсуждаются на ежедневных стендапах или сессиях планирования, а затем при необходимости уточняются или заменяются кодом. Это позволяет существенно снизить затраты на поддержание моделей в актуальном состоянии и органично вписать визуальное моделирование в итеративный процесс разработки. Роль UML в таком контексте трансформируется от тотального документирования к живому коммуникативному инструменту и средству быстрого прототипирования альтернативных проектных решений.

Современные интегрированные среды разработки и CASE-средства предоставляют широкие возможности для автоматизации работы с UML. Платформы Enterprise Architect, Visual Paradigm, MagicDraw, а также легковесные инструменты вроде PlantUML и Mermaid позволяют не только рисовать диаграммы, но и выполнять обратное проектирование (reverse engineering) по готовому коду, генерировать скелетные реализации классов, проверять согласованность моделей (например, соответствие диаграммы классов диаграммам последовательности) и автоматически экспортировать техническую документацию в форматах HTML, PDF или Markdown. Отдельного внимания заслуживает архитектурный подход Model-Driven Architecture (MDA), где модель UML на независимом от платформы уровне (Platform Independent Model, PIM) с помощью формальных трансформаций преобразуется в модель на уровне конкретной платформы (Platform Specific Model, PSM), а из неё – в исполняемый код. Это направление существенно ускоряет разработку типовых корпоративных систем, особенно в таких доменах, как банковские информационные системы, системы управления ресурсами предприятия (ERP) и документооборота. В распределённых и облачных системах диаграммы развёртывания UML позволяют наглядно описать размещение компонентов по физическим или виртуальным узлам, задать параметры сетевых соединений и требования к ресурсам – что является критически важным для команд, практикующих DevOps-культуру.

Российская инженерная практика проектирования информационных систем также активно использует UML, особенно в государственных информационных системах, оборонных проектах и крупных корпоративных разработках, где требуется строгая документация в соответствии с национальными стандартами (ГОСТ Р). Ряд отечественных нормативных документов, включая ГОСТ Р ИСО/МЭК 19501-2005, гармонизирован с международными спецификациями UML. В банковской сфере, телекоммуникационной отрасли и при создании ERP-систем диаграммы UML являются неотъемлемой частью технического проекта. При этом российские команды разработчиков часто комбинируют UML с классическими ER-диаграммами для проектирования реляционных баз данных, а также дополняют модели текстовыми сценариями в структурированной форме. Тем не менее, наблюдается устойчивый дефицит квалифицированных архитекторов, владеющих UML на уровне, достаточном для проектирования гетерогенных распределённых систем с тысячами классов и сложными протоколами взаимодействия.

Взгляд в будущее позволяет выделить несколько ключевых векторов эволюции UML. Во-первых, назрела необходимость расширения языка для моделирования микросервисных архитектур, контейнеризации (Docker, Kubernetes), serverless-вычислений и mesh-сетей. Разрабатываются специализированные профили UML для предметно-ориентированных областей: для систем реального времени – профиль MARTE (Modeling and Analysis of

Real-Time and Embedded Systems), для киберфизических систем – профили на основе UML MARTE и SysML. Во-вторых, активно идёт интеграция UML с методами машинного обучения: предлагается использовать UML для описания пайплайнов обработки данных, архитектур нейронных сетей и сценариев их обучения, что может стать стандартом для MLOps-платформ. В-третьих, исследуются возможности автоматического анализа UML-моделей на предмет архитектурных уязвимостей, конфликтов прав доступа и нарушений шаблонов безопасности – это превращает UML в элемент безопасной разработки (DevSecOps), позволяя выявлять риски на самых ранних этапах проектирования. В отдалённой перспективе не исключена трансформация UML в более легковесный, но формально верифицируемый язык, возможно, с встроенной поддержкой контрактного программирования и формальных спецификаций.

Заключение

В рамках выполненного исследования была всесторонне проанализирована роль унифицированного языка визуального моделирования UML в современной практике проектирования информационных систем. Полученные результаты убедительно свидетельствуют, что, несмотря на смену парадигм разработки (переход от каскадных моделей к гибким методологиям и DevOps-культуре), UML сохраняет свою актуальность как ключевой инструмент визуализации архитектуры, средство коммуникации между междисциплинарными командами и основа для документирования сложных объектно-ориентированных систем. Основными преимуществами UML являются богатый и хорошо стандартизированный набор диаграмм для статического и динамического моделирования, широкая поддержка со стороны индустриальных CASE-средств и возможность интеграции с архитектурным подходом Model-Driven Architecture. В то же время язык обладает объективными ограничениями: избыточность, высокий порог входа для начинающих архитекторов и недостаточная формальность, что стимулирует сообщество к адаптации UML в рамках концепции «UML как набросок». Сравнительный анализ с альтернативными нотациями (BPMN, IDEF) показывает, что UML занимает собственную нишу технического проектирования: он уступает BPMN в наглядности для бизнес-заказчика, но значительно превосходит его в выразительной силе для разработчиков и архитекторов. Дальнейшее развитие UML связывается с созданием профилей для микросервисных и киберфизических систем, интеграцией с конвейерами машинного обучения и включением механизмов формальной верификации в стандарт. Таким образом, UML остаётся неотъемлемой и востребованной частью профессионального инструментария современного архитектора информационных систем, адаптируясь к новым условиям и сохраняя свою ценность для индустрии программного обеспечения.

Библиографический список

1. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Якобсон. – 2-е изд. – М.: ДМК Пресс, 2021. – 496 с.

2. Ларман К. Применение UML и шаблонов проектирования. – 3-е изд. – М.: Вильямс, 2019. – 736 с.

3. Фаулер М. UML. Основы. – 3-е изд. – СПб.: Символ-Плюс, 2020. – 192 с.

4. Константайн, Л. UML: от проектирования до реализации / Л. Константайн, Л. Локвуд. – М.: ЛОРИ, 2018. – 340 с.

5. Емельянова, Н.З. Проектирование информационных систем / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. – М.: Форум: ИНФРА-М, 2022. – 448 с.

6. Вендров А.М. Проектирование программного обеспечения. – М.: Финансы и статистика, 2020. – 560 с.

7. Одинцов И.О. Профессиональное программирование. Системный подход. – 2-е изд. – СПб.: БХВ-Петербург, 2021. – 624 с.

THE ROLE OF UML IN MODERN INFORMATION SYSTEMS DESIGN

D.A. Boshchenko, *Student*
Supervisor: *T.P. Mashikhina, Candidate of Pedagogical Sciences, Associate Professor*
Volgograd State University
(Russia, Volgograd)

Abstract. *This article explores the role of the Unified Modeling Language (UML) in contemporary information systems design. It analyzes the main types of UML diagrams, their application across different stages of the development lifecycle, as well as the strengths and weaknesses of the language. A comparative analysis of UML with alternative modeling approaches (IDEF, BPMN) is provided. Attention is given to the integration of UML into agile development methodologies and its place in documenting the architecture of complex systems. Future trends in the evolution of UML are outlined in the context of current challenges facing system designers.*

Keywords: *UML; Unified Modeling Language; information systems design; class diagrams; software architecture; visual modeling; agile methodologies; CASE tools; business process notations.*