НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2025 Том 25 № 6 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2025 Vol. 25 No 6 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

КРАТКИЕ СООБЩЕНИЯ

BRIEF PAPERS

doi: 10.17586/2226-1494-2025-25-6-1229-1233
УДК 004.2

Проектные механизмы микроархитектурного уровня для встраиваемых систем

Максим Вячеславович Кольчурин[1]✉, Алексей Евгеньевич Платунов[2]

[1] ООО «ЛМТ», Санкт-Петербург, 199034, Российская Федерация
[2] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] maxim.kolchurin@gmail.com✉, https://orcid.org/0000-0002-7061-9357
[2] aeplatunov@itmo.ru, https://orcid.org/0000-0003-3003-3949

Аннотация
Стремительно растущая потребность в проектировании специализированных вычислительных систем (особенно встраиваемых систем и систем на кристалле) сдерживается ограниченными возможностями разработчиков из-за высокой сложности задачи. Предлагается «проектный механизм» как абстракция (абстрактное понятие) с явным разделением на архитектурную логику и уровень реализации по комплексному критерию. Его применение наиболее эффективно при проектировании подсистем с высокой внутренней вариативностью и множеством технологических альтернатив. Разработанная абстракция на всех этапах маршрута проектирования позволяет эффективнее формировать и анализировать пространство проектных решений, радикально сокращая концептуальные ошибки и расширяя для повторного использования число результатов проектной деятельности.

Ключевые слова
встраиваемые системы, архитектурные решения, метрики

Ссылка для цитирования: Кольчурин М.В., Платунов А.Е. Проектные механизмы микроархитектурного уровня для встраиваемых систем // Научно-технический вестник информационных технологий, механики и оптики. 2025. Т. 25, № 6. С. 1229–1233. doi: 10.17586/2226-1494-2025-25-6-1229-1233

Design mechanisms at the microarchitectural level for embedded systems

Maxim V. Kolchurin[1]✉, Alexey E. Platunov[2]

[1] LMT Ltd., Saint Petersburg, 199034, Russian Federation
[2] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] maxim.kolchurin@gmail.com✉, https://orcid.org/0000-0002-7061-9357
[2] aeplatunov@itmo.ru, https://orcid.org/0000-0003-3003-3949

Abstract
The rapidly growing demand for the design of specialized computing systems, primarily embedded systems and systems-on-chip, constrained by the limited capabilities of developers, due to the high complexity of the task. A Design Mechanism is proposed as an abstraction (a conceptual construct) that explicitly separates architectural logic from the implementation level according to a comprehensive criterion. Its use is most effective in the design of subsystems characterized by high internal variability and numerous technological alternatives. Such an abstraction, applied at all stages of the design process, enables more efficient formation and analysis of the design solution space, radically reducing conceptual errors and expanding the number of design outcomes available for reuse.

Keywords
embedded systems, design patterns, architectural solutions, metrics

For citation: Kolchurin M.V., Platunov A.E. Design mechanisms at the microarchitectural level for embedded systems. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2025, vol. 25, no. 6, pp. 1229–1233 (in Russian). doi: 10.17586/2226-1494-2025-25-6-1229-1233

В области проектирования встраиваемых систем проблема заполнения семантического разрыва между представлениями концептуального (архитектурного) уровня и уровня реализации по-прежнему стоит очень остро [1]. Она заключается в существовании «серой зоны» между архитектурным этапом проектирования и этапом реализации, недостаточно обеспеченной методологически и инструментально. Сокращение семантического разрыва возможно за счет выделения в архитектурном проектировании микроархитектурного этапа [2], который включает в себя определение внутренней организации компонентов, выбор алгоритмов, адаптацию решений под платформу, обоснование компромиссов. Другими словами, основная задача микроархитектурного этапа проектирования — формирование, визуализация и документирование пространства детализированных проектных решений [3].

В настоящей работе рассматривается проблема представления инженерных решений на микроархитектурном уровне для встраиваемых систем и предлагается подход, направленный на повышение степени формализации и переносимости таких решений между технологическими контекстами. Ключевое отличие состоит в принудительном разделении каждого инженерного решения на архитектурную логику и множество вариантов реализации, адаптированных под конкретные платформы и ограничения, а также в способе такого разделения, основанном на явной фиксации инварианта решения и критериев корректности.

Современные методологии проектирования систем на кристалле, встраиваемых и киберфизических систем развивались в ответ на растущее многообразие, внутреннюю сложность и многовариантность технических решений. Существующие методологии, такие как модель-ориентированные подходы (MBSE, SysML/UML) [4, 5], формальные методы (TLA+, UPPAAL) [6, 7], отраслевые фреймворки (DDD, TOGAF) [8] — частично преодолевают семантический разрыв между замыслом и реализацией, но сосредоточены на отдельных аспектах проектирования, таких как согласованность представлений, верификация свойств, структурирование процессов.

Работы, связанные с развитием платформно-ориентированного проектирования (Platform Based Design, PBD) [9, 10], заложили важную концептуальную основу — ортогонализацию (разделение) между функцией (что система делает) и архитектурой (как система это делает). Вместе с тем PBD сосредоточено на выборе платформы и отображении функциональности на ее компоненты через симуляцию и оценку производительности, документируя результаты этого процесса, но не рассматривает (как самостоятельную) задачу извлечения и формализации архитектурного знания, направляющего эти решения, в форме, пригодной для повторного использования. Практики записи архитектурных решений (Architecture Decision Records, ADR) [11] документируют принятые решения, но фиксируют их постфактум, не раскрывая компромиссы между альтернативами.

Таким образом, существующие подходы либо описывают процесс принятия решений (PBD), либо фиксируют их результат (ADR), но не предлагают артефакта, который обеспечивал бы явную связь между сутью принятого проектного решения и свойствами его технологических воплощений.

Для заполнения этого пробела предлагается концепция, основанная на абстракции (абстрактном понятии) — «проектный механизм» (ПМ) — подход к формализации инженерных решений, направленный на структурирование архитектурного знания, который обеспечивает явное разделение логики проектного решения и конкретных технических воплощений. ПМ включает в себя:

— архитектурную логику — технологически нейтральное описание решения на микроархитектурном уровне [1, 2], фиксирующее обоснование выбора, ключевые компромиссы, метрики для сравнения альтернатив и границы применимости;
— реализации — конкретные воплощения архитектурной логики с учетом особенностей и ограничений конкретных технологических платформ.

Практическая применимость ПМ опирается на четыре критерия корректности:

1. Технологическая нейтральность (инвариантность к платформе). Архитектурная логика сохраняет семантику в разных технологических контекстах и не содержит деталей конкретных реализаций.
2. Генеративная способность. Из архитектурной логики выводится множество реализаций для различных условий, при этом логика не предписывает конкретную технологию или инструмент.
3. Измеримость свойств. Архитектурная логика содержит (или указывает на) метрики и критерии, обеспечивающие количественное сравнение альтернатив и валидацию соответствия решения заявленным свойствам.
4. Трассируемость реализации. Каждая реализация явно связана с компонентами архитектурной логики через объяснимые проектные выборы, что предотвращает искусственные сопоставления и обеспечивает воспроизводимость.

Критерии корректности трансформируют процесс выбора из качественного сравнения технологий в количественный анализ соответствия характеристик реализаций ограничениям проекта, обеспечивая переносимость архитектурного знания между проектами. ПМ наиболее применим при проектировании подсистем, где существуют: множество технологических альтернатив; нефункциональные характеристики определяются разнородными критериями; компромиссы между вариантами не очевидны, и цена ошибочного выбора высока.

Продемонстрируем применение ПМ на задаче организации подсистемы конфигурирования для промышленного контроллера со встроенной средой исполнения пользовательских программ. Такие программы требуют параметризации (сетевые настройки, пороги срабатывания, режимы работы и т. д.) без модификации внутренней программы (прошивки) контроллера. Рассматриваемая система разработана на базе микроконтроллера без блока управления памятью и механизма страничной памяти и, как следствие, без

механизмов защиты от фрагментации. Начальная реализация подсистемы конфигурирования использовала хранилище типа ключ-значение[1], которое обеспечивало устойчивость к сбоям питания и равномерный износ постоянного хранилища данных. Реализация такого механизма требовала применения динамической памяти, которая подвержена фрагментации (15–20 % по измерениям), а также создавала непредсказуемые паузы из-за работы сборщика мусора (Garbage Collector, GC), которые достигали нескольких миллисекунд при полном заполнении активного хранилища. В сценариях с высокой нагрузкой на чтение (одна запись на 5000 чтений) механизм оказался избыточным. Адаптация через специальные режимы работы, изоляция области динамической памяти, хотя и улучшали ситуацию, но не устраняли фундаментальные ограничения текущей реализации.

Поиск альтернатив сталкивался с проблемами. Текстовые форматы (Tom's Obvious Minimal Language (TOML), JavaScript Object Notation (JSON)) требуют синтаксического анализа при исполнении, нагружая динамическую память. Генерация структур данных из схем данных (Concise Binary Object Representation (CBOR) с использованием Concise Data Definition Language (CDDL)) ограничивает гибкость эволюции и требует постоянных выделений памяти при исполнении. Анализ неудач выявил ключевое противоречие — формат, удобный для написания человеком, оптимален для одной фазы (генерация конфигурации), но не для других (валидация и исполнение). Эти требования не конкурируют во времени, а относятся к разным стадиям процесса конфигурирования устройства. Если требования не пересекаются, они могут быть удовлетворены через разные артефакты. Каждая стадия получает оптимальный для нее формат вместо единого компромиссного решения. Конфигурация существует на трех стадиях (генерация, транспорт и исполнение), и для каждой из них используется представление, оптимизированное под ее задачи. Неизменным должна оставаться семантическая составляющая представления и учитываться ограничения по времени доступа. Это было зафиксировано в виде архитектурной логики для поиска вариантов реализаций такой подсистемы. В результате рассмотрено несколько вариантов, основанных на выявленной архитектурной логике. На этапе подготовки применены форматы, удобные для чтения человеком (JSON и TOML), различия между ними несущественны для настоящей работы, поэтому дальнейший анализ сосредоточен на стадии исполнения. Требования проекта определяют метрики для сравнения реализаций, так, потребление оперативной памяти (ОЗУ) не должно превышать 32 КБ, (выделенный объем памяти для подсистемы), время доступа стремится к $O(1)$ (асимптотическая сложность), необходима статическая валидация (на этапе компиляции) и поддержка прямого использования данных (таблица).

Выбор осуществлялся последовательным исключением вариантов по критическим ограничениям, TOML и CBOR исключены из-за постоянного высокого потребления ОЗУ на грани лимита, Key-Value хранилище — из-за фрагментации, пауз GC и отсутствия проверки типов данных. Структуры Си исключены из-за необходимости ручной кодогенерации, снижающей гибкость, сложности переноса на другие языки. FlatBuffers является наиболее подходящим под критерии, он обеспечивает прямое использование данных (zero-copy) непосредственно из флэш-памяти в режиме eXecute In Place, константные задержки доступа к данным ($O(1)$) и compile-time валидацию схемы данных. В результате применения FlatBuffers отсутствует этап интерпретации, прямой доступ без поиска, отсутствие пауз GC.

Применимость ПМ не ограничивается подсистемами конфигурирования. Например, авторы работы имеют опыт применения ПМ при проектировании коммуникационных подсистем промышленной автоматики. При выборе протоколов для сетей реального времени ПМ помог структурировать сравнение альтернатив (802.15.4e TSCH, CAN, EtherCAT, Modbus) по единым критериям (гарантии задержки, джиттер, число узлов) вместо качественного сравнения разнородных спецификаций. В распределенных системах сбора данных ПМ фиксирует компромисс между предсказуемостью доставки и энергоэффективностью, сравнивая реализации протоколов (MQTT, AMQP, DDS, LWM2M) по приоритетным для проекта метрикам.

ПМ отличается от абстракций, используемых в методологиях PBD или ADR. Он предназначен для реше-

[1] [Электронный ресурс]. Режим доступа: https://github.com/Infineon/kv-store (дата обращения: 22.09.2025).

Таблица. Сравнение вариантов реализаций подсистемы конфигурирования

Table. Comparison of configuration subsystem implementation options

Реализация	Объем ОЗУ, КБ	Сложность доступа к данным	Проверка типов данных	Прямой доступ к данным (zero-copy)	Соответствие ограничениям
Интерпретация TOML	до 80	$O(n)$	Во время исполнения	Нет	Нет
CBOR с CDDL	до 32	$O(1)$	Во время исполнения	Нет	Нет
Key-Value хранилище	около 20*	$O(n)$	Нет	Нет	Нет
Структуры языка Си	10	$O(1)$	На этапе компиляции	Да	Возможно**
FlatBuffers	0	$O(1)$	На этапе компиляции	Да	Да

Примечание: * — с фрагментацией, по замерам 15–20 % и паузами на сборку мусора; ** — требует ручной кодогенерации, снижает гибкость.

ния проблем проектирования на уровне микроархитектуры. Среди решаемых проблем: явное определение общих проектных принципов, устойчивых к различным реализациям; минимизация привязки к конкретной платформе; фокусировка на архитектурных решениях и их свойствах без погружения в детали конкретной технической реализации.

ПМ наиболее эффективен при проектировании подсистем с высокой внутренней вариативностью и множеством технологических альтернатив. Такой подход избыточен для проектов с однозначным выбором технологии (диктуется стандартом или платформой) или когда функциональные требования полностью определяют реализацию. В таких случаях достаточно традиционного ADR для фиксации решения.

Ограничения подхода связаны с необходимостью явной формулировки метрик, но не все проектные компромиссы поддаются количественной оценке (например, «удобство интеграции» или «зрелость экосистемы»). В этих случаях ПМ дополняется экспертными оценками. Кроме того, извлечение архитектурной логики требует аналитических усилий, что может быть неоправданно для одноразовых решений без перспективы переиспользования.

Абстракции проектирования, осознанно применяемые разработчиками встраиваемых систем на всех этапах проектирования, играют определяющую роль в повышении производительности труда. Пространство проектных решений, с которым работает инженер в области компьютинга, объективно трудно формализуемо, особенно на этапах архитектурного (микроархитектурного) проектирования. Следует отметить, что процесс освоения технологий проектирования на основе такого рода абстракций, как представленный ПМ, в качестве альтернативы «привычному» манипулированию техническими решениями (конструированию) уровня реализации требует значительных усилий и желания, и должен активнее демонстрироваться в образовательных учреждениях.

Литература

1. Кольчурин М.В., Пинкевич В.Ю., Платунов А.Е. Усиление роли микроархитектурных этапов проектирования встраиваемых систем // Научно-технический вестник информационных технологий, механики и оптики. 2022. Т. 22. № 4. С. 716–724. https://doi.org/10.17586/2226-1494-2022-22-4-716-724
2. Kolchurin M., Platunov A., Asminkin F., Pinkevich V. Architectural abstractions in the design of distributed embedded systems platforms // Proc. of the 24th International Multidisciplinary Scientific GeoConference SGEM. 2024. V. 24. N 2.1. P. 19–26. https://doi.org/10.5593/sgem2024/2.1/s07.03
3. Pinkevich V., Platunov A., Gorbachev Ya. Design of embedded and cyber-physical systems using a cross-level microarchitectural pattern of the computational process organization // CEUR Workshop Proceedings. 2020. V. 2893. P. 1–14.
4. Technical Report: SysML v1 to SysML v2 Model Conversion Approach. Office of Systems Engineering and Architecture. Washington, D.C., 2024. 49 p.
5. Alenazi M., Niu N., Savolainen J. SysML modeling mistakes and their impacts on requirements // Proc. of the IEEE 27th International Requirements Engineering Conference Workshops (REW). 2019. P. 14–23. https://doi.org/10.1109/REW.2019.00010
6. Neider D., Roy R. What is formal verification without specifications? A survey on mining LTL specifications // Lecture Notes in Computer Science. 2025. V. 15262. P. 109–125. https://doi.org/10.1007/978-3-031-75778-5_6
7. Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume I: Planning and Technology Insertion. Technical report NASA-GB-002-95. Washington, DC: NASA, 1995. 227 p.
8. Kotusev S. TOGAF-based enterprise architecture practice: An exploratory case study // Communications of the Association for Information Systems. 2018. V. 43. N 1. P. 321–359. https://doi.org/10.17705/1CAIS.04320
9. Sangiovanni-Vincentelli A., Carloni L., De Bernardinis F., Sgroi M. Benefits and challenges for platform-based design // Proc. of the 41st annual Design Automation Conference. 2004. P. 409–414. https://doi.org/10.1145/996566.996684
10. Keutzer K., Malik S., Newton A.R., Rabaey J.M., Sangiovanni-Vincentelli A. System-level design: orthogonalization of concerns and platform-based design // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2000. V. 19. N 12. P. 1523–1543. https://doi.org/10.1109/43.898830
11. Buchgeher G., Schöberl S., Geist V., Dorninger B., Haindl P., Weinreich R. Using architecture decision records in open source projects — an MSR study on GitHub // IEEE Access. 2023. V. 11. P. 63725–63740. https://doi.org/10.1109/ACCESS.2023.3287654

References

1. Kolchurin M.V., Pinkevich V.Yu., Platunov A.E. Strengthening the role of microarchitectural stages of mbedded systems design. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2022, vol. 22, no. 4, pp. 716–724. (in Russian). https://doi.org/10.17586/2226-1494-2022-22-4-716-724
2. Kolchurin M., Platunov A., Asminkin F., Pinkevich V. Architectural abstractions in the design of distributed embedded systems platforms. *Proc. of the 24th International Multidisciplinary Scientific GeoConference SGEM*, 2024, vol. 24, no. 2.1, pp. 19–26. https://doi.org/10.5593/sgem2024/2.1/s07.03
3. Pinkevich V., Platunov A., Gorbachev Ya. Design of embedded and cyber-physical systems using a cross-level microarchitectural pattern of the computational process organization. *CEUR Workshop Proceedings*, 2020, vol. 2893, pp. 1–14
4. *Technical Report: SysML v1 to SysML v2 Model Conversion Approach*. Office of Systems Engineering and Architecture. Washington, D.C., 2024, 49 p.
5. Alenazi M., Niu N., Savolainen J. SysML modeling mistakes and their impacts on requirements. *Proc. of the IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019, pp. 14–23. https://doi.org/10.1109/REW.2019.00010
6. Neider D., Roy R. What is formal verification without specifications? A survey on mining LTL specifications. *Lecture Notes in Computer Science*, 2025, vol. 15262, pp. 109–125. https://doi.org/10.1007/978-3-031-75778-5_6
7. *Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume I: Planning and Technology Insertion*. Technical report NASA-GB-002-95. Washington, DC: NASA, 1995. 227 p.
8. Kotusev S. TOGAF-based enterprise architecture practice: An exploratory case study. *Communications of the Association for Information Systems*, 2018, vol. 43, no. 1, pp. 321–359. https://doi.org/10.17705/1CAIS.04320
9. Sangiovanni-Vincentelli A., Carloni L., De Bernardinis F., Sgroi M. Benefits and challenges for platform-based design. *Proc. of the 41st annual Design Automation Conference*, 2004, pp. 409–414. https://doi.org/10.1145/996566.996684
10. Keutzer K., Malik S., Newton A.R., Rabaey J.M., Sangiovanni-Vincentelli A. System-level design: orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2000, vol. 19, no. 12, pp. 1523–1543. https://doi.org/10.1109/43.898830
11. Buchgeher G., Schöberl S., Geist V., Dorninger B., Haindl P., Weinreich R. Using architecture decision records in open source projects — an MSR study on GitHub. *IEEE Access*, 2023, vol. 11, pp. 63725–63740. https://doi.org/10.1109/ACCESS.2023.3287654

Авторы

Кольчурин Максим Вячеславович — инженер, ООО «ЛМТ», Санкт-Петербург, 199034, Российская Федерация, SC 57225126062, https://orcid.org/0000-0002-7061-9357, maxim.kolchurin@gmail.com
Платунов Алексей Евгеньевич — доктор технических наук, профессор, профессор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 35318291200, https://orcid.org/0000-0003-3003-3949, aeplatunov@itmo.ru

Authors

Maxim V. Kolchurin — Engineer, LMT Ltd., Saint Petersburg, 199034, Russian Federation, SC 57225126062, https://orcid.org/0000-0002-7061-9357, maxim.kolchurin@gmail.com
Alexey E. Platunov — D.Sc., Full Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 35318291200, https://orcid.org/0000-0003-3003-3949, aeplatunov@itmo.ru

Статья поступила в редакцию 30.09.2025
Одобрена после рецензирования 16.10.2025
Принята к печати 17.11.2025

Received 30.09.2025
Approved after reviewing 16.10.2025
Accepted 17.11.2025