УДК [004.4.22+004.94]:303.21:629.7 EDN: HYDTTL

МЕТОДИКА СИСТЕМНОГО ПРОЕКТИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ ИСПОЛНЯЕМОГО УНИФИЦИРОВАННОГО ЯЗЫКА ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ НА ПРИМЕРЕ ПРОГРАММНО-АППАРАТНЫХ КОМПЛЕКСОВ КОНТРОЛЯ ПОЛЕТА РАКЕТНО-КОСМИЧЕСКОЙ ТЕХНИКИ

А.И. Власов vlasovai@bmstu.ru
Д.Е. Широков shirokov.den@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Аннотация

Рассмотрено применение исполняемого унифицированного языка визуального моделирования при разработке программно-аппаратных комплексов ракетно-космической техники на примере комплекса контроля полета. Основное внимание уделено методике системного проектирования с использованием исполняемого унифицированного языка визуального моделирования xtUML. Проанализированы структура языка, его преимущества и способы интеграции средств визуального проектирования в процесс разработки сложных систем на примере программно-аппаратных комплексов контроля полета ракетно-космической техники. Предложена методика системного визуального проектирования сложных интегрированных, интеллектуальных компонентов ракетно-космических систем. Применение визуальных языков позволяет формализовать детальное представление как программных, так и аппаратных частей сложных систем, не разделяя их. Это достигается вследствие универсальности и гибкости используемого визуального инструментария. Показано взаимодействие между различными компонентами системы, что делает его незаменимым инструментом для междисциплинарных команд разработчиков системных проектов, требующих интеграции программного и аппаратного обеспечений. Использование предложенной методики позволяет сократить время разработки и структуризации системного проекта, т. е. оперативно вносить изменения, поскольку ком-

Ключевые слова

Системный проект, системный анализ, интеллектуальные интегрированные системы, визуальные языки, диаграмма деятельности, диаграмма предметной области, диаграмма состояний, маршрут разработки, программно-аппаратная система, xtUML

поненты и их взаимодействие четко определены. Такая технология системного проектирования позволяет разработчикам быстрее адаптироваться к изменениям и внедрять новые функции без необходимости глубокого анализа всей системы. В результате системный проект становится более гибким и устойчивым к изменениям требований и удовлетворяет требованиям синхронного системного проектирования

Поступила 17.03.2025
Принята 04.09.2026
© Автор(ы), 2026

Отдельные результаты работы получены в рамках выполнения государственного задания FSFN-2024-0086

Введение. В рамках современных тенденций цифровой трансформации ключевое значение приобретает широкое использование систем, которые в режиме реального времени собирают информацию с датчиков, анализируют ее, проводят интеллектуальную обработку и передают результаты анализа человеку или следующим устройствам [1–3]. При увеличении масштабов подобных проектов, создание и поддержание таких систем становится сложной и трудоемкой задачей, требующей использования средств автоматизации и каскадирования процесса разработки [4]. Во время поэтапной разработки в рамках комплексного системного проекта важно использовать определенный, заранее согласованный и формализованный уровень абстракции для представления проектируемых объектов и процессов. Достигнуть этого помогает использование исполняемого унифицированного языка визуального моделирования xtUML (executable and translatable UML) [5]. В отличие от традиционных визуальных языков (IDEF, UML, BPMN и др.), xtUML позволяет описывать структуру и логическое поведение систем и автоматически генерировать готовый программный код. Исполняемый язык xtUML широко используют для высокоуровневого проектирования цифровой компонентной базы.

Цель настоящей работы — разработать методику проектирования сложных систем, используя исполняемый унифицированный визуальный язык моделирования xtUML, на примере программно-аппаратных комплексов контроля полета летательного аппарата как сложной аппаратно-программной системы [6–8].

В общем случае визуальные языки представляют собой графические метафоры, используя которые разработчики создают и модифицируют системы без необходимости разработки непосредственно кода. Это упрощает процесс разработки, снижает вероятность ошибок и ускоряет время выхода системы на рынок [5].

Предлагаемая авторская методика системного проектирования с использованием исполняемого унифицированного языка визуального моделирования позволяет создавать модели сложных систем, их компонентов и связей между ними на разных уровнях абстракции, связанных в единое целое. Это обеспечивает лучшее понимание структуры системы, позволяет выявить и формализовать информационные и управляющие потоки, упростить ее проектирование. Новизна предлагаемого решения при проектировании цифровой компонентной базы заключается в упрощении процесса формализованного перехода от высокоуровневого (системного) описания к этапам схемного и топологического проектирования, что позволяет связать в рамках единого маршрута проектирования высоко абстрактное представление цифровой элементной базы с ее аппаратной реализацией.

Одним из распространенных визуальных языков системного проектирования является унифицированный язык моделирования UML [5–8] — набор графических примитивов, составляющих диаграммы, которые позволяют создавать модели систем, описывающих их структуру, поведение и взаимодействие с окружением. Исполняемый язык UML (xtUML) [6, 7] формирует расширение языка UML, применяемое для формализации логических цифровых схем на начальных стадиях проектирования, адаптированное для создания исполняемых, тестируемых и переводимых моделей программного и аппаратного обеспечения. Рассмотрим особенности применения языка xtUML на этапах системного проектирования цифровых компонентов.

Исполняемый язык UML — xtUML — является одновременно методом разработки и высокоабстрактным языком проектирования [9–12]. Функциональность xtUML предоставляет возможность генерации кода на основе созданных визуальных моделей, обеспечивая комплексное моделирование, симуляцию и отладку моделей для проверки логики, трансляцию модели, системную интеграцию и документирование. Преимущества использования xtUML по сравнению с другими визуальными инструментами системного проектирования кроме традиционных следующие: визуализация, улучшенное представление логики системы, генерация кода в полуавтоматическом режиме и возможность ведения стандартизированной документации, а также генерация и исполняемость кода, независимость от платформы, возможность раннего обнаружения логических ошибок [11].

Методы и модели. Модель xtUML использует принципы модельно-ориентированной архитектуры MDA (Model-Driven Architecture). Обобщенный способ организации взаимодействия модулей при преобразовании модели xtUML в программный код приведен на рис. 1.

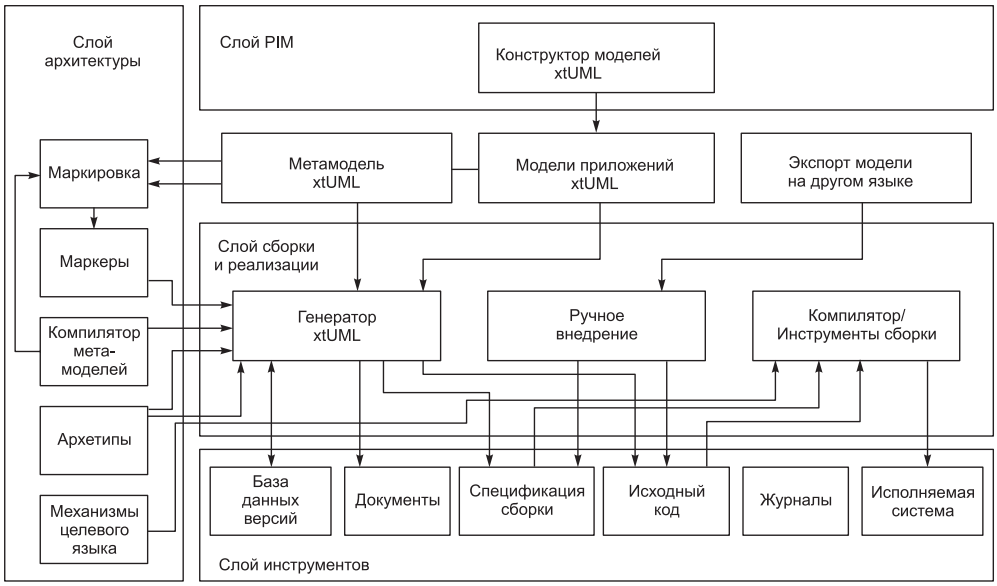


Рис. 1. Обобщенный способ организации модулей при преобразовании модели xtUML в программный код

Сначала формируется абстрактная модель логического модуля PIM (Platform Independent Model). Она определяет логику модуля, данные и поведение без привязки к конкретной платформе реализации.

Модель xtUML позволяет создавать исполняемые модули PIM, которые затем преобразуются (транслируются) в PSM (Platform Specific Model) или напрямую в код для целевой среды (слой сборки и реализации). Для этого в модели xtUML используются различные элементы описания архитектуры, такие как классы, состояния и события, которые позволяют разработчикам визуализировать систему на высоком уровне абстракции (слой архитектуры). Эти элементы моделируют логику и взаимодействие между объектами, что делает модель понятной и легко изменяемой.

После завершения моделирования инструменты генерации кода (слой инструментов) позволяют автоматически преобразовать созданную модель в исполняемый код на выбранном языке программирования. Генерация кода включает в себя создание классов, методов и интерфейсов, которые точно соответствуют описанным элементам. При этом поддерживаются принципы инкапсуляции и абстракции, что способствует созданию чистого и поддерживаемого кода.

Процесс перевода включает в себя автоматическое создание тестов на основе моделей, что позволяет убедиться в правильности работы системы еще до ее реализации. Это существенно снижает риски, связанные с внед-

рением новых функций и изменений. Отметим, что модель xtUML поддерживает итеративный подход к разработке, что позволяет корректировать модель и генерировать код без ручного редактирования [9]. Такой подход значительно ускоряет процесс разработки и повышает качество конечного продукта. Модели также могут быть использованы для документирования системы, что облегчает понимание архитектуры как настоящими, так и будущими разработчиками [10–12].

Моделирование на языке xtUML происходит на нескольких уровнях абстракции[1]. Объектом моделирования на языке xtUML являются преимущественно исполняемые (встраиваемые) системы, в связи с этим моделирование фокусируется на структуре и логике поведения системы. Результатами такого моделирования являются четыре основных диаграммы: предметной области, классов (class diagram), состояний (statechart) и деятельности (activity).

Диаграмма предметной области — специализированная диаграмма классов, описывающая статические структуры данных, атрибуты и связи между ними внутри конкретного домена. В отличие от классического языка UML, xtUML фокусируется на создании исполняемых моделей, где классы, атрибуты и отношения преобразуются непосредственно в программный код. Диаграмма предметной области позволяет разделить задачу на отдельные сущности — домены. Разделение задач позволяет экспертам разрабатывать и проверять каждый домен в системе независимо от других. Связи между доменами называются мостами, они описывают требования, предъявляемые доменами друг к другу.

Диаграмма классов в xtUML — визуальное представление иерархии классов и их отношений; классы — это основные элементы диаграммы, на которой кроме классов и отношений между ними отображаются их свойства (атрибуты) и методы [13, 14]. Отношения между классами указывают, каким образом классы взаимодействуют друг с другом (наследование, ассоциация, агрегация или композиция).

Диаграмма состояний — визуальное представление переходов между различными состояниями объекта или системы в течение их жизненного цикла. Диаграмма описывает, как объект может менять свое состояние в ответ на внешние события или внутренние условия. Примерами состояний могут служить начальное и активное состояния, ожидание, ошибка и др.

[1] Спецификация. URL: https://xtuml.org (дата обращения: 16.03.2025).

Диаграмма деятельности отражает последовательность действий, которые выполняются объектом или системой в процессе их работы. Диаграмма деятельности наглядно представляет, какие действия выполняются, в каком порядке и как они влияют на состояние объекта или системы. Она включает в себя действия, связанные в виде последовательностей, условий или циклов, описывая логику работы системы. Таким образом, можно сказать, что диаграмма деятельности является аналогом алгоритмического описания работы системы.

Используя возможности визуального языка xtUML, можно сформировать основу маршрута разработки цифровой платформы, построенную на итеративной разработке соответствующих моделей и диаграмм:

– анализ технического задания (ТЗ);

– выделение основных частей системы, определение их взаимных требований и связей; формирование списка доменов и диаграммы предметной области;

– выделение групп элементов каждого домена, выполняющих общие функции — классов, их атрибутов и операций; формирование диаграммы классов, выбор модульных схемотехнических решений, соответствующих классам;

– определение состояний системы и условий их изменения; формирование диаграммы состояний;

– определение взаимодействия частей системы друг с другом и внешней средой; формирование диаграммы деятельности, определение алгоритма работы итоговой программы;

– написание программного кода; формирование принципиальной электрической схемы.

При обнаружении неисправностей или логических ошибок на любом перечисленном этапе происходит возврат на предыдущий шаг.

Результаты. Рассмотрим подробнее методику системного проектирования исполняемых (встраиваемых) цифровых систем с использованием исполняемого унифицированного языка визуального моделирования xtUML на примере системы контроля полета ракетно-космической техники. Программно-аппаратная система контроля полета как сложная система включает в себя множество взаимосвязанных компонентов, для управления которыми надо выявить и формализовать разнородные информационные потоки.

Этап 1. Анализ ТЗ. В качестве объекта, для которого будет разрабатываться программно-аппаратная система контроля полета, выступает физи-

ческая модель ракеты (ФМР). Проанализируем исходные данные, требования и ограничения. Разрабатываемая система контроля и передачи информации является бортовой системой и работает в автономном режиме. Основные требования и ограничения к программно-аппаратной системе контроля полета следующие.

1. После запуска система работает в автономном режиме без вмешательства оператора.

2. Система выполняет две основные задачи: получает и передает полетную информацию; запускает подсистему спасения на основании значений полетной информации.

3. Передача полетной информации может осуществляться через радиосвязь ОВЧ/ВЧ-диапазона (очень высокочастотного/высокочастотного диапазона), спутниковые каналы, системы передачи данных ACARS (Aircraft Communications Addressing and Reporting System), транспондеры (режим S) и проводные сети AFTN/AMHS (Aeronautical Fixed Telecommunication Network/Aeronautical Message Handling Systems), применяются цифровые сообщения, голосовая связь и телеграммы для обмена планами полетов, метеоданными и трекингом судов в реальном времени.

4. Полетная информация принимается радиостанцией, подключаемой к компьютеру по USB, и выводится в установленном формате.

5. Установленный формат передачи полетной информации предполагает передачу: уникального кода, времени с момента запуска системы, напряжения питания, высоты полета, модуля вектора ускорения, информации (в формате true/false) о старте полета, достижении апогея траектории, запуске и срабатывании системы спасения, приземлении.

6. Подсистема спасения запускается в апогее траектории полета или после его прохождения.

7. Для оценки работы системы необходимы световой индикатор включения системы, звуковой и световой индикаторы работы подсистемы передачи полетной информации.

8. Для управления работой системы необходимы следующие переключатели: кнопка включения, кнопка тестового срабатывания подсистемы спасения, кнопка отключения подсистемы передачи полетной информации.

9. Необходимо минимизировать массу и размеры системы, обеспечить ее ремонтопригодность.

В результате предварительного анализа исходных данных можно сделать вывод о том, что разрабатываемая система будет содержать значительное число взаимосвязанных элементов. Для решения этой задачи

предлагается использовать средство визуализации и структурирования — исполняемый унифицированный язык моделирования.

Если для решения задачи требуется применение специального вычислителя, то важно уже на этапе анализа исходных данных определить используемый метод программирования [13, 14].

Этап 2. Формирование диаграммы предметной области. С ее помощью можно разделить задачу на составляющие — домены. Это позволяет специалистам в каждой области разрабатывать и проверять свои домены независимо друг от друга.

Цель формирования диаграммы предметной области — ответ на следующие вопросы.

1. Какие сущности (домены) предметной области существуют в системе?

2. Как сущности предметной области друг с другом связаны?

3. Какие действия можно выполнять с сущностями предметной области?

Важно понимать различие диаграммы предметной области и диаграммы классов. Диаграмма предметной области описывает элементы реального мира, для которых программное обеспечение должно предоставить решение, диаграмма классов описывает статическую структуру системы в терминологии классов объектно-ориентированного программирования [13]. Следовательно, основное различие между ними заключается в том, что диаграмма предметной области описывает элементы реального мира, а диаграмма классов — статическую структуру модели системы на основе классов [13, 14].

Диаграмма предметной области включает в себя домены, мосты, методы и атрибуты домена. В ТЗ обозначены две основные подсистемы — спасения и приема/передачи информации. Обе подсистемы управляются контроллером. Все подсистемы физически связаны с моделью ракеты. Соответственно, обозначим на диаграмме предметной области пять доменов: ФМР, подсистемы спасения, приема информации, передачи информации и контроллер. Подсистема приема/передачи разделена на два дочерних домена, так как включает два отдельных устройства. Атрибуты, методы и связи доменов диаграммы предметной области для ФМР приведены на рис. 2.

На этапе формирования диаграммы предметной области разработчики формируют общее представление о разрабатываемой системе и распределении выполняемых функций между подсистемами. В крупных проектах работа над отдельными доменами может передаваться разным разработчикам или командам.

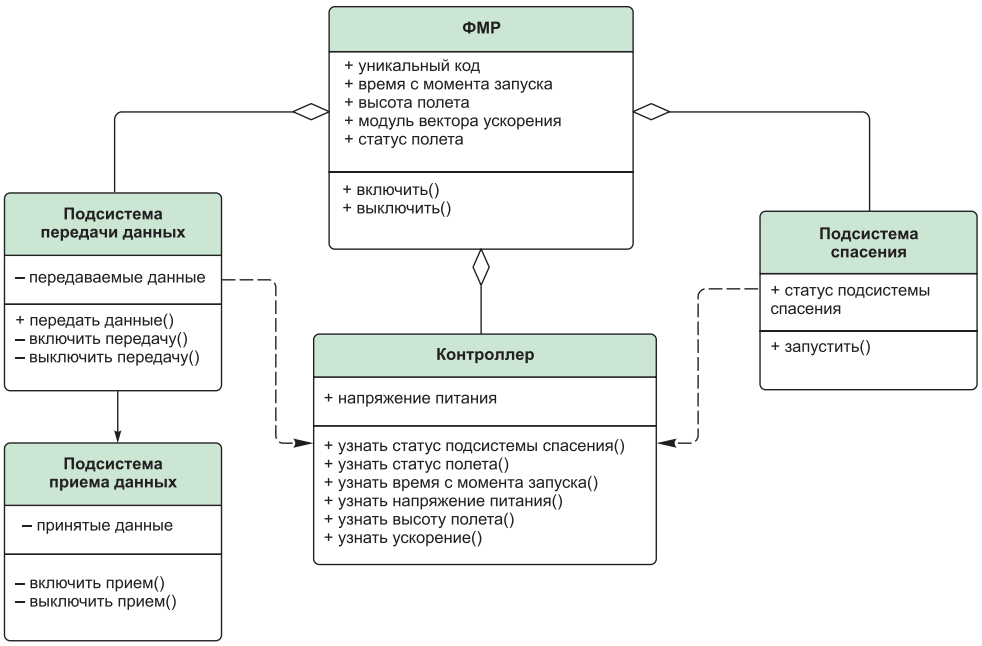


Рис. 2. Атрибуты, методы и связи доменов диаграммы предметной области для ФМР

Этап 3. Формирование диаграммы классов. Диаграмма классов в xtUML — инструмент для визуализации структуры и взаимодействия классов в объектно-ориентированной системе, она помогает разработчикам лучше понять и описать функциональность программного продукта. Диаграмма классов визуализирует структуру системы, описывает ее функциональность, взаимодействие классов, помогает анализировать систему, облегчает общение в команде, используется при проектировании и служит документацией.

Домены в xtUML представляют собой независимые функциональные области, описывающие предметную область (например, «Управление», «Контроль») через модели классов, состояний и действий. Они позволяют разрабатывать модульные системы, где каждый домен моделирует реальность, а не программную реализацию, что обеспечивает высокую степень абстракции.

Классы выражают отдельную понятийную сущность предметной области и характеризуются пользовательским наименованием этой сущности.

Создание диаграммы классов в xtUML включает в себя следующие шаги.

1. *Определение классов.* Определите классы, которые будут приведены на диаграмме; класс — набор объектов, которые имеют общие свойства и поведение.

2. *Определение атрибутов.* Определите атрибуты, которые будут представлены на диаграмме; атрибут — свойство класса.

3. *Определение методов.* Определите методы, которые будут приведены на диаграмме; метод — поведение класса.

4. *Определение отношений.* Определите отношения между классами, которые будут представлены на диаграмме. Отношения могут быть разных типов: ассоциация, композиция, наследование и др.

5. *Создание диаграммы.*

Диаграмма классов в xtUML включает в себя следующие основные типы отношений [6, 7].

1. Ассоциации (Association) характеризуют связи между классами (направленные/ненаправленные, единичные/множественные).

2. Агрегации (Aggregation) указывают на то, что один класс содержит экземпляры другого независимого класса.

3. Композиции (Composition) указывают на то, что один класс содержит экземпляры другого класса.

4. Обобщения (Generalization) характеризуют иерархии наследования.

5. Зависимости (Dependency) указывают на то, какие зависимости между классами будут приведены — односторонние или двусторонние.

6. Наследование (Inheritance) характеризует иерархию классов и отражает, как классы могут наследовать свойства и методы от других классов.

Диаграмма классов для ФМР приведена на рис. 3. Рассматриваемая система состоит из небольшого числа классов, поэтому для наглядности диаграмма не разделена на части, соответствующие доменам. Однако в крупных проектах целесообразно проводить разделение, чтобы не возникало искусственного укрупнения классов. После формирования диаграммы для каждого домена обязательно необходимо их объединение для проверки полноты и избыточности.

Следует учитывать, что составляемая диаграмма должна отражать программную и логическую (схемотехническую) части проекта как единое целое. При разработке встраиваемых систем и компонентной базы диаграммы классов в xtUML может служить заменой структурной и функциональной электрических схем, поскольку отражает как функциональные части изделия, так и взаимосвязи между ними и их функции (в виде методов). При составлении названных схем следует убедиться, что они значительно не расходятся по структуре с диаграммой классов.

Важный результат составления диаграммы классов — проверка соответствия разработанной системы функциональным требованиям, для этого

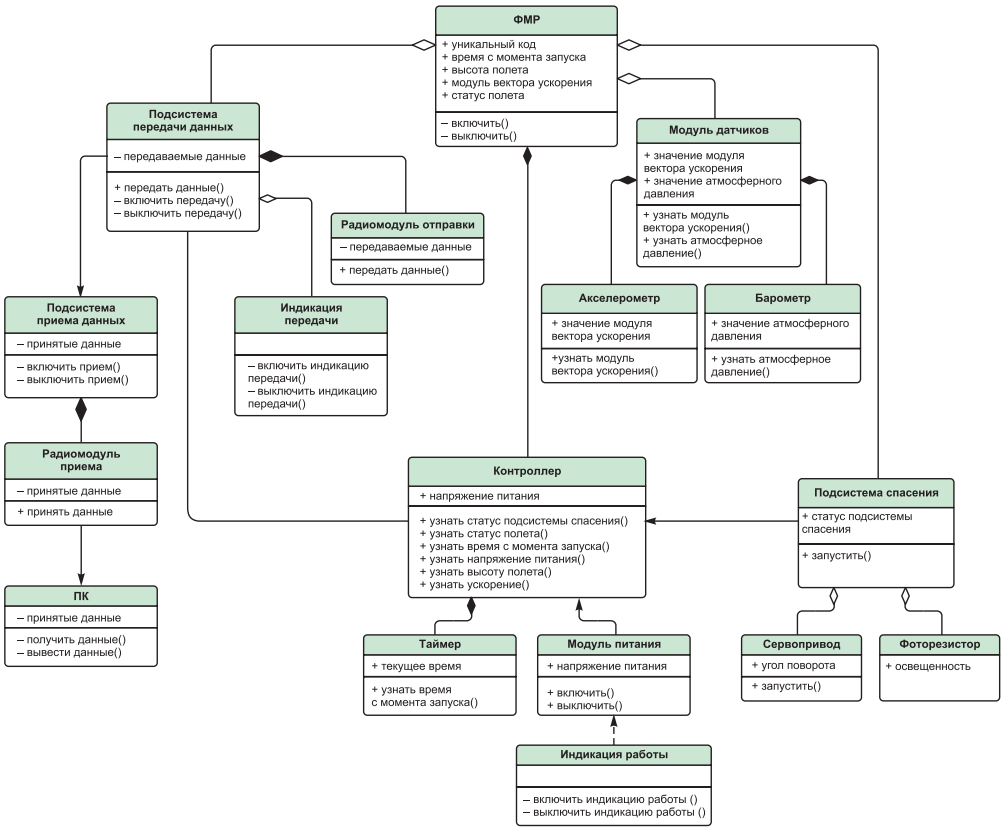


Рис. 3. Диаграмма классов для ФМР

необходимо сопоставить заданные функции системы с методами, обозначенными на диаграмме. Пример соответствия функций разрабатываемой системы методам на диаграмме классов приведен в табл. 1. Если система имеет функции, которым не соответствует не один метод на диаграмме классов, то необходимо добавить этот метод к одному из имеющихся классов или пересмотреть структуру в целом и добавить новый класс.

Этап 4. Формирование диаграммы состояний. Диаграмма состояний показывает все возможные состояния объекта и переходы между ними. Она используется для описания поведения объекта в течение его жизненного цикла. Диаграмма состояний помогает описать, как система изменяет состояние в ответ на различные воздействия и события. Спецификация основных элементов диаграммы состояний приведена ниже [6, 7].

1. Состояние — положение, в котором может находиться объект; оно может быть активным (объект выполняет какие-то действия) или пассивным (объект ожидает какого-то события).

2. Переход — отношение между двумя состояниями, показывающее, что объект, находящийся в первом состоянии, будет выполнять некоторые действия и перейдет в другое состояние как только наступит некоторое событие, при этом будут удовлетворены необходимые сторожевые условия.

3. Действие — поведение системы при переходе из одного состояния в другое, может быть простым (например, вывод сообщения на экран) или сложным (например, выполнение алгоритма).

Таблица 1

Соответствие функций разрабатываемой системы методам на диаграмме классов

Функция	Ответственный класс	Метод
Передача полетной информации в установленном формате: уникальный код; время с момента запуска системы; напряжение питания; высота полета; модуль вектора ускорения; информация (в формате true/false) о старте полета, достижении апогея траектории, запуске и срабатывании системы спасения, приземлении	Подсистемы передачи и приема данных; контроллер; модуль датчиков	Передать и принять данные(); узнать: время с момента запуска(); напряжение питания(); высоту полета(); модуль вектора ускорения(); статус полета(); статус системы спасения()
Срабатывание системы спасения	Подсистема спасения	Запустить()
Индикация работы системы передачи данных	Подсистема передачи данных; индикация передачи	Включить и выключить индикацию передачи()

Согласно xtUML, система может находиться в состоянии действия или ожидания события. Конец и начало жизненного цикла системы обозначаются начальным и конечным состояниями.

Цель разработки диаграммы состояний — переход от описания статичной системы как набора классов, способных выполнять определенные функции, к динамично изменяющейся системе, переходящей между раз-

ными состояниями под действием внешних и внутренних факторов. Если диаграммы предметной области и классов приводят к пониманию структуры схемотехнической части, то для формирования и понимания алгоритма работы программного кода необходимы еще два шага: диаграмма состояний и диаграмма действий [14].

Диаграмма состояний для решаемой задачи, включающая в себя восемь состояний, приведена на рис. 4. В квадратных скобках обозначено условие перехода в новое состояние, без скобок — действие, сопровождающее переход. В линейном, на первый взгляд, процессе полета ФМР составленная диаграмма позволяет увидеть неочевидное развитие событий, вызванное тестовым запуском системы спасения (предусмотрено ТЗ). В этом случае необходимо предусмотреть перезапуск всех систем, переводящий модель в начальное состояние. Отметим, что на диаграмме не отражена работа подсистемы приема/передачи данных, поскольку она не относится к состоянию всей системы напрямую, уточнение ее функционала будет происходить на следующих шагах.

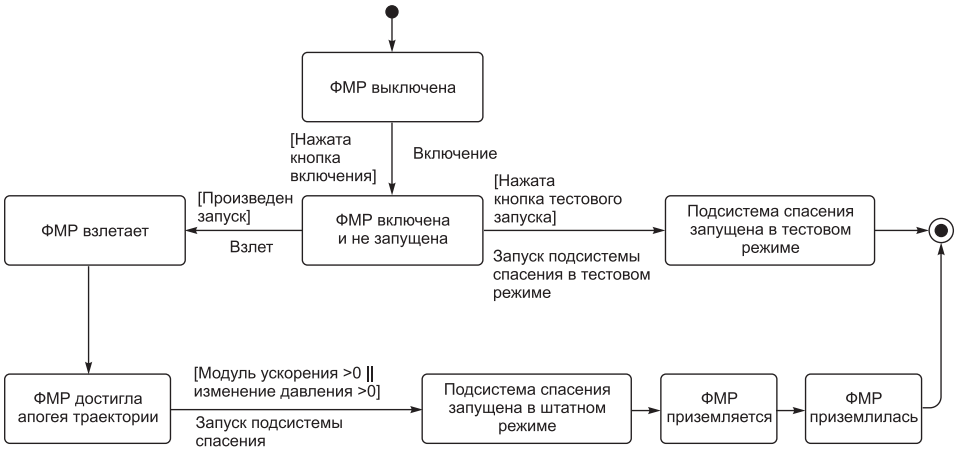


Рис. 4. Диаграмма состояний ФМР

Диаграмма состояний — графическое представление конечного автомата, где каждое состояние отражает определенное состояние системы, а переходы между ними — возможные изменения состояния в ответ на события [11]. Диаграмма состояний отражает поведение системы или процесса и помогает разработчикам понять жизненный цикл объекта и условия, влияющие на его работу. На этом этапе важно предусмотреть различные режимы работы и состояния, вызванные нестандартными воздействиями, чтобы обеспечить корректное функционирование системы в любых условиях.

Этап 5. Формирование диаграммы деятельности. Диаграмма представляет собой инструмент визуализации потока управления в системе, позволяет представить последовательность операций, проанализировать процесс работы системы. Важной особенностью данной диаграммы является наличие дорожек-полос (swimlane), разделяющих процесс. На ней группируются действия, относящиеся к одной категории, например, действующие лица (роль или отдел), зона ответственности, стадия процесса или что-либо еще, что должно быть подчеркнуто [13–15]. В случае моделирования программно-аппаратной системы целесообразно разделять процесс на дорожки, соответствующие либо доменам диаграммы предметной области (рис. 5), либо ключевым состояниям диаграммы.

Диаграмма отражает связь параллельно происходящих процессов на контроллере ФМР и в наземной подсистеме приема данных(при этом вычисления выполняются только на одном контроллере, другие параллельные действия на диаграмме отсутствуют) и отображает как последовательность действий, так и поток данных, собираемых в процессе работы и объединяемых в объект «Полетные данные».

Диаграмма деятельности — последний подготовительный шаг для перехода к завершению процесса разработки и написанию программного кода.

Этап 6. Завершение процесса разработки. После создания необходимых диаграмм в xtUML подготовительные этапы разработки можно полагать завершенными. К этому моменту разработчики имеют полное понимание структуры как программной, так и аппаратной частей системы. Сформированные диаграммы дополняют, расширяют друг друга и дают возможность оценить правильность стратегии разработки. Например, разработка визуальных моделей элементов сенсорных модулей для регистрации интегральной поглощенной дозы радиационных излучений (РИ) на основе радиационно-чувствительных транзисторов со структурой металл–диэлектрик–полупроводник (МДП, RADFET–Radiation–Sensitive Field Effect Transistor) [16]. Полупроводниковый кристалл с МДП-сенсорами РИ и диаграмма доменов цифрового фильтра приведены на рис. 6.

Реализация радиационно-чувствительных датчиков на основе МДП-структур дает возможность формировать на одном полупроводниковом кристалле (см. рис. 6) как чувствительный элемент, так и устройство обработки сигналов [16]. Такой подход создает предпосылки для разработки линейки датчиков РИ нового поколения.

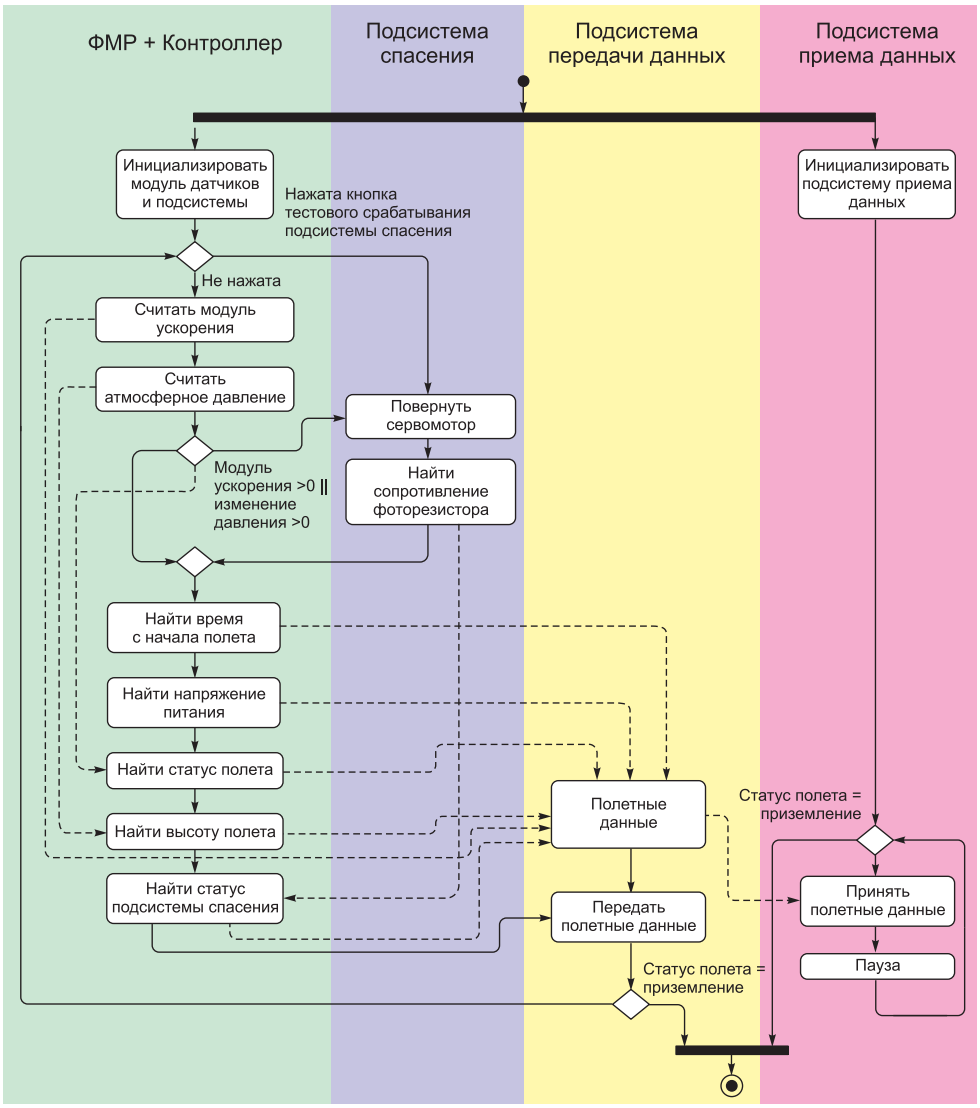


Рис. 5. Диаграмма деятельности ФМР

Обсуждение результатов. Язык xtUML может интегрироваться с различными методологиями управления проектами, такими как Agile или Waterfall [17–20]. Диаграммы, созданные с помощью специализированных инструментов, легко редактировать и экспортировать. Модель xtUML позволяет генерировать код на различных языках программирования, таких как Java, C#, Python и др. Необходимо создать диаграммы, которые описывают структуру программы, и выбрать опции генерации кода в настройках инструмента. В табл. 2 и на рис. 7 систематизированы и обобщены этапы маршрута визуального проектирования и варианты моделирования

на языке xtUML. Информация, полученная на предыдущих этапах, позволяет приступить к разработке программного кода и принципиальной электрической схемы.

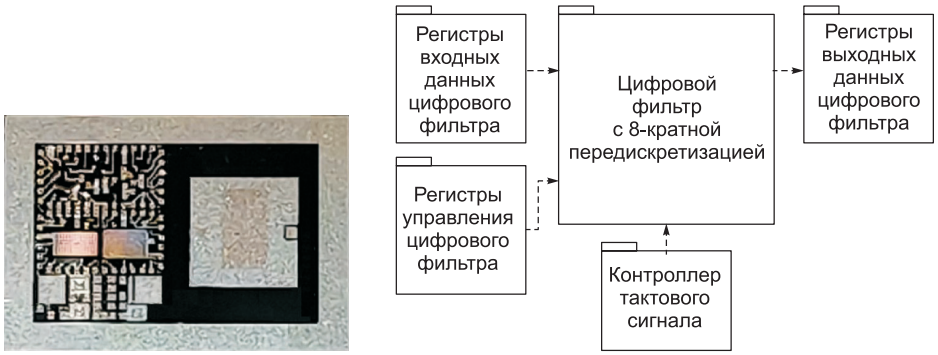


Рис. 6. Полупроводниковый кристалл с МДП-сенсорами РИ (*а*) и диаграмма доменов цифрового фильтра (*б*)

Выбор языка программирования зависит от требований проекта и предпочтений разработчиков. Принципиальная схема устройства должна соответствовать требованиям безопасности, надежности и эффективности работы устройства. Оба этапа — разработка программного кода и принципиальной схемы устройства — требуют тщательного анализа и проектирования, позволяют реализовать функциональность, определенную в xtUML, и создать работающее устройство.

Заключение. Методика системного проектирования с использованием исполняемого унифицированного языка визуального моделирования xtUML позволяет создавать исполняемые и переносимые модели с возможностью последующей кодогенерации. Создаваемые модели встраиваемых цифровых компонентов имеют высокий уровень абстракции, возможность проверки логики через симуляцию, верификацию и валидацию, платформенную независимость и меньшее число ошибок по сравнению с традиционными моделями.

Кроме непосредственного моделирования, диаграммы в xtUML являются эффективным средством документирования и формального описания проектов элементов цифровой техники. В отличие от традиционных методов документирования, таких как электрические схемы или блок-схемы программного кода, язык xtUML позволяет визуализировать системы, включающие программные и аппаратные части, не разделяя их между собой. Это достигается благодаря его универсальности и гибкости.

Таблица 2

Маршрут визуального проектирования

<table>
<tr><td rowspan="2">Этап</td><td colspan="2">Задачи при работе с частью</td></tr>
<tr><td>программной</td><td>аппаратной</td></tr>
<tr><td>1. Анализ ТЗ</td><td>Решение о наличии программной части. Выбор парадигмы, методологии и языка программирования. Выбор контроллера</td><td>Составление списка задач, выполняемых системой. Предварительный выбор типа датчиков и исполнительных устройств</td></tr>
<tr><td>2. Формирование доменов предметной области</td><td colspan="2">Выделение основных компонентов системы (доменов), определение их взаимных требований и связей. На следующих шагах маршрут может быть разделен на параллельные независимые ветви, соответствующие выделенным доменам</td></tr>
<tr><td rowspan="3">3. Формирование доменов классов</td><td colspan="2">Разделение каждого домена на более мелкие компоненты (классы), представляющие определенные модули или выполняемые задачи</td></tr>
<tr><td>Определение атрибутов, важных для описания класса и его взаимодействия с другими классами. Определение методов каждого класса за счет распределения между ними функций системы и необходимых функций компонентов. Выделенные классы далее могут использоваться в программном коде</td><td>Выбор схемотехнических решений или готовых модулей для выполнения определенных ранее задач. Описание функций, входных и выходных параметров каждого элемента или группы. При необходимости формирование структурной и функциональной схем. Возможен переход к этапу 6</td></tr>
<tr><td colspan="2">Проверка полноты составленной диаграммы постановкой в соответствие методов классов и функций системы, описанных в ТЗ</td></tr>
</table>

4. Формирование доменов состояний	Составление диаграммы состояний для всей системы или каждого домена как представление конечного автомата. Проверка выполнимости состояний средствами, определенными диаграммой классов	
5. Формирование доменов деятельности	Описание алгоритма действий системы для отдельных доменов и всей системы. Аналог блок-схемы программного кода	
6. Завершение процесса разработки	Разработка или автоматическая генерация программного кода	Формирование принципиальной схемы и перечня элементов

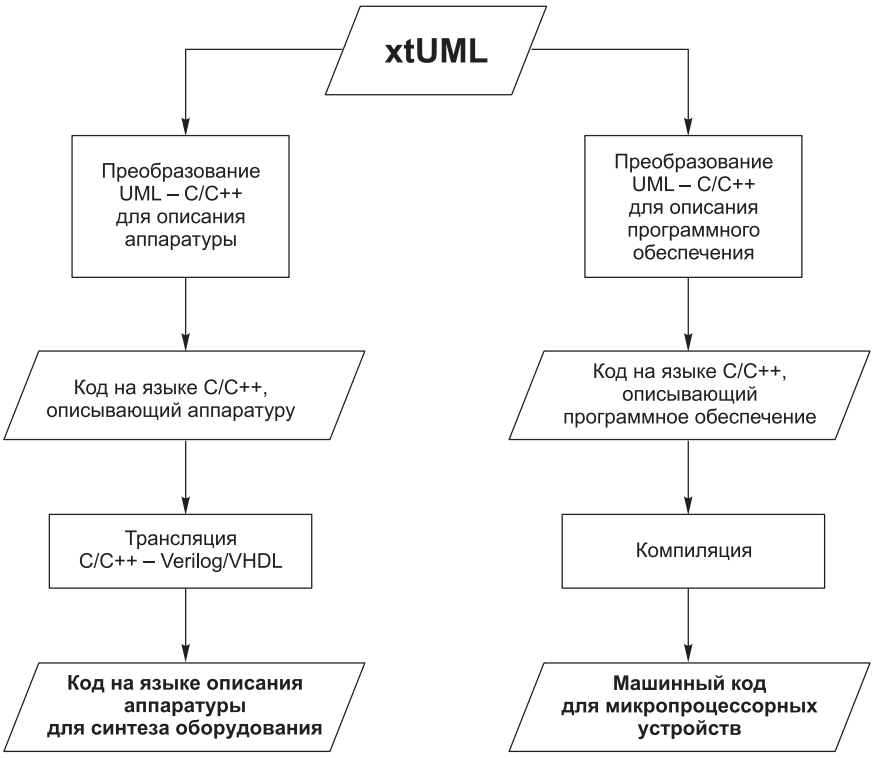


Рис. 7. Варианты моделирования на языке xtUML

Язык визуального моделирования xtUML предоставляет возможность создавать диаграммы, которые объединяют элементы как программного, так и аппаратного обеспечения, что значительно упрощает понимание и управление сложными системами. Такой подход особенно важен в современных проектах, где границы между программным и аппаратным обеспечением становятся все более размытыми, например, в области интернета вещей или встраиваемых систем.

Язык xtUML наглядно отображает взаимодействие между различными компонентами системы, что делает его незаменимым инструментом для междисциплинарных команд, работающих над проектами, требующими интеграции программного и аппаратного обеспечений.

С помощью предложенной методики можно обеспечить управление сложностью проекта посредством его иерархической декомпозиции, работать с абстракциями высокого уровня. Структурированность системного проекта, обеспеченная диаграммами в xtUML, упрощает внедрение новых технических решений, так как все компоненты и их взаимодействие четко определены, и сокращает общее время разработки. Это позволяет разработчикам быстрее адаптироваться к изменениям и внедрять новые функ-

ции без необходимости глубокого анализа всей системы. В результате, проект становится более гибким и устойчивым к изменениям требований и удовлетворяет требованиям синхронного системного проектирования.

Предлагаемая методика системного проектирования цифровой компонентной базы на основе языка визуального моделирования xtUML может быть востребована разработчиками сложных систем в таких отраслях, как электронная и аэрокосмическая, автомобилестроение (например, микроэлектроника, распределенные системы управления, встраиваемые системы и крупные инфраструктурные проекты).

ЛИТЕРАТУРА

[1] Shakhnov V.A., Kurnosenko A.E., Demin A.A., et al. Industry 4.0 visual tools for digital twin system design. In: *Software engineering perspectives in intelligent systems*. Springer, 2020, vol. 2, pp. 864–875. DOI: https://doi.org/10.1007/978-3-030-63319-6_80

[2] Малежик О.И., Остапенко О.С., Малежик В.А. и др. Принципы построения программного обеспечения оперативного контроля полетов воздушных судов по полетной информации. *Проблемы информатизации и управления*, 2007, т. 3, № 21, с. 104–108. EDN: ZEEWNN

[3] Преображенский Л.А., Процкая Л.Г. Проблемы обеспечения длительного жизненного цикла наземно-бортового комплекса контроля и регистрации полетной информации. *Авиационные системы в XXI веке. Сб. тез. докл*. Т. 1. М., ГосНИИАС, 2016, с. 379–384. EDN: YJJRKP

[4] Vlasov A.I., Shakhnov V.A. Visual methodology for the multi-factor assessment of industrial digital transformation components. In: *Digital transformation in industry.* Springer, 2021, pp. 57–65. DOI: https://doi.org/10.1007/978-3-030-73261-5_6

[5] Кознов Д.В., Ольхович Л.Б. Визуальные языки проектов. *Системное программирование*, 2005, т. 1, с. 148–167. EDN: NJVHCT

[6] Raistrick C. Model driven architecture with executable UML. Cambridge, Cambridge University Press, 2004.

[7] Mellor S.J., Balcer M.J. Executable UML. A foundation for model-driven architecture. Boston, Addison-Wesley Professional, 2002.

[8] Nuzzo P., Sangiovanni-Vincentelli A.L., Bresolin D., et al. A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proc. IEEE*, 2015, vol. 103, no. 11, pp. 2104–2132.
DOI: https://doi.org/10.1109/JPROC.2015.2453253

[9] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Application of visual tools for system modeling of digital integrated circuits. *Problems of Advanced Micro- and Nanoelectronic Systems Development (MES)*, 2020, no. 1, pp. 8–14.

[10] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Visual methods of high-level system design for digital hardware components. *J. Phys.: Conf. Ser.*, 2020, vol. 1515, art. 42024. DOI: https://doi.org/10.1088/1742-6596/1515/4/042024

[11] Власов А.И., Расюк А.А., Хохлунова О.Д. Системный проект ПЛИС с использованием визуальных языков. *Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС)*, 2022, № 3, с. 18–23. EDN: QTBUTO

[12] Хальзев С.Е., Власов А.И., Шахнов В.А. Использование языка диаграмм xtUML для высокоуровневого проектирования цифровой компонентной базы. *Датчики и системы*, 2021, № 1 (254), с. 18–24. DOI: https://doi.org/10.25728/datsys.2021.1.3

[13] Brown A.W. Model driven architecture: principles and practice. *Softw. Syst. Model.*, 2004, vol. 3, no. 4 (4), pp. 314–327. DOI: https://doi.org/10.1007/s10270-004-0061-2

[14] Ramstein M., Jacobson I., Booch G. Software engineering with UML. Auerbach Publications, 2017.

[15] Graics B., Molnár V., Vörös A., et al. Mixed-semantics composition of statecharts for the component-based design of reactive systems. *Softw. Syst. Model.*, 2020, vol. 19, no. 6, pp. 1483–1517. DOI: https://doi.org/10.1007/s10270-020-00806-5

[16] Andreev D.V. Accumulation and suppression of radiation-induced charge in MOS structures. *J. Surf. Investig.*, 2024, vol. 18, no. 2, pp. 461–465. DOI: https://doi.org/10.1134/s102745102402023x

[17] Карпунин А.А., Стерликов М.С., Варибрус Т.В. и др. AGILE — сравнение с другими методологиями. *Технологии инженерных и информационных систем*, 2023, № 4, с. 133–146.

[18] Демин А.А., Карпунин А.А., Ганев Ю.М. Методы верификации и валидации сложных программных систем. *Программные продукты и системы*, 2014, № 4, с. 229–233. DOI: http://dx.doi.org/10.15827/0236-235X.108.229-233

[19] Карпунин А.А., Ганев Ю.М., Чернов М.М. Методы обеспечения качества при проектировании сложных программных систем. *Надежность и качество сложных систем*, 2015, № 2 (10), с. 78–84. EDN: UIQABT

[20] Ванройе Н.К., Карпунин А.А. Анализ принципов бизнес-планирования ИТ-проекта. *Информационные технологии в проектировании и производстве*, 2018, № 4 (172), с. 31–35. EDN: YOFBDN

Власов Андрей Игоревич — канд. техн. наук, доцент кафедры «Проектирование и технология производства электронной аппаратуры» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Широков Денис Евгеньевич — инженер НОЦ «Функциональные Микро/Наносистемы» МГТУ им. Н.Э. Баумана (Российская Федерация, 105005, Москва, 2-я Бауманская ул., д. 5, стр. 1).

Просьба ссылаться на эту статью следующим образом:

Власов А.И., Широков Д.Е. Методика системного проектирования с использованием исполняемого унифицированного языка визуального моделирования на примере программно-аппаратных комплексов контроля полета ракетно-космической техники. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2026, № 3 (156), с. 102–126. EDN: HYDTTL

A SYSTEM DESIGN METHODOLOGY USING AN EXECUTIVE UNIFIED VISUAL MODELING LANGUAGE USING THE EXAMPLE OF HARDWARE AND SOFTWARE COMPLEXES FOR FLIGHT CONTROL IN ROCKET AND SPACE TECHNOLOGY

A.I. Vlasov vlasovai@bmstu.ru
D.E. Shirokov shirokov.den@bmstu.ru

BMSTU, Moscow, Russian Federation

Abstract

The article examines the use of an executable unified visual modeling language in the development of software and hardware complexes for rocket and space technology, using the flight control complex as an example. The focus is on system design techniques using the executable unified visual modeling language (xtUML). The structure of the language, its advantages and methods of integrating visual design tools into the development of complex systems are analyzed using the example of software and hardware systems for controlling the flight of rocket and space technology. As a result of the study, a methodology for systematic visual design of complex integrated, intelligent components of rocket and space systems was proposed. Unlike traditional methods of documenting a system project, the use of visual languages allows you to formalize a detailed representation of both software and hardware parts of a complex system without dividing them among themselves. This is due to the versatility and flexibility of the visual tools used. This approach allows you to visualize the interaction between different components of the system, which makes it an indispensable tool for interdisciplinary development teams working on system projects that require software and hardware integration. Use of the proposed method allows reducing

Keywords

System design, systems analysis, intelligent integrated systems, visual languages, activity diagram, domain diagram, state diagram, development path, firmware, xtUML

development time. The structure of the system project allows you to quickly make changes, since all components and their interaction are clearly defined. This system design technology allows developers to quickly adapt to changes and implement new features without the need for in-depth analysis of the entire system. As a result, the system design becomes more flexible and resistant to changes in requirements and meets the requirements of synchronous system design

Received 17.03.2025
Accepted 04.09.2026
© Author(s), 2026

Separate results of work were obtained within the framework of the State Assignment (FSFN-2024-0086)

REFERENCES

[1] Shakhnov V.A., Kurnosenko A.E., Demin A.A., et al. Industry 4.0 visual tools for digital twin system design. In: *Software engineering perspectives in intelligent systems*. Springer, 2020, vol. 2, pp. 864–875. DOI: https://doi.org/10.1007/978-3-030-63319-6_80

[2] Malezhik O.I., Ostapenko O.S., Malezhik V.A., et al. Principles of planning of the program providing of operational control of the aircrafts flights, using the flight information. *Problemy informatizatsii i upravleniya* [Problems of Informatization and Management], 2007, vol. 3, no. 21, pp. 104–108 (in Russ.). EDN: ZEEWNN

[3] Preobrazhenskiy L.A., Protskaya L.G. [Challenges of ensuring a long life cycle for ground-airborne flight data monitoring and recording systems]. *Aviatsionnye sistemy v XXI veke. Sb. tez. dokl. T. 1* [Proc. Aviation Systems in the 21st Century. Vol. 1]. Moscow, GosNIIAS Publ., 2016, pp. 379–384 (in Russ.). EDN: YJJRKP

[4] Vlasov A.I., Shakhnov V.A. Visual methodology for the multi-factor assessment of industrial digital transformation components. In: *Digital transformation in industry*. Springer, 2021, pp. 57–65. DOI: https://doi.org/10.1007/978-3-030-73261-5_6

[5] Koznov D.V., Olkhovich L.B. Visual project languages. *Sistemnoe programmirovanie*, 2005, vol. 1, pp. 148–167 (in Russ.). EDN: NJVHCT

[6] Raistrick C. Model driven architecture with executable UML. Cambridge, Cambridge University Press, 2004.

[7] Mellor S.J., Balcer M.J. Executable UML. A foundation for model-driven architecture. Boston, Addison-Wesley Professional, 2002.

[8] Nuzzo P., Sangiovanni-Vincentelli A.L., Bresolin D., et al. A platform-based design methodology with contracts and related tools for the design of cyber-physical systems. *Proc. IEEE*, 2015, vol. 103, no. 11, pp. 2104–2132.
DOI: https://doi.org/10.1109/JPROC.2015.2453253

[9] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Application of visual tools for system modeling of digital integrated circuits. *Problems of Advanced Micro- and Nanoelectronic Systems Development (MES)*, 2020, no. 1, pp. 8–14.

[10] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Visual methods of high-level system design for digital hardware components. *J. Phys.: Conf. Ser.*, 2020, vol. 1515, art. 42024. DOI: https://doi.org/10.1088/1742-6596/1515/4/042024

[11] Vlasov A.I., Rasyuk A.A., Khokhlunova O.D. FPGA system design using visual languages. *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem (MES)* [Problems of Advanced Micro- and Nanoelectronic Systems Development (MES)], 2022, no. 3, pp. 18–23 (in Russ.). EDN: QTBUTO

[12] Khalzev S.E., Vlasov A.I., Shakhnov V.A. Using the xtUML diagram language for high-level design of a digital component base. *Datchiki i sistemy* [Sensors & Systems], 2021, no. 1 (254), pp. 18–24 (in Russ.). DOI: https://doi.org/10.25728/datsys.2021.1.3

[13] Brown A.W. Model driven architecture: principles and practice. *Softw. Syst. Model.*, 2004, vol. 3, no. 4, pp. 314–327. DOI: https://doi.org/10.1007/s10270-004-0061-2

[14] Ramstein M., Jacobson I., Booch G. Software engineering with UML. Auerbach Publications, 2017.

[15] Graics B., Molnár V., Vörös A., et al. Mixed-semantics composition of statecharts for the component-based design of reactive systems. *Softw. Syst. Model.*, 2020, vol. 19, no. 6, pp. 1483–1517. DOI: https://doi.org/10.1007/s10270-020-00806-5

[16] Andreev D.V. Accumulation and suppression of radiation-induced charge in MOS structures. *J. Surf. Investig.*, 2024, vol. 18, no. 2, pp. 461–465. DOI: https://doi.org/10.1134/s102745102402023x

[17] Karpunin A.A., Sterlikov M.S., Varibrus T.V., et al. AGILE — comparison with other methodologies. *Tekhnologii inzhenernykh i informatsionnykh system* [Technologies of Engineering and Information Systems], 2023, no. 4, pp. 133–146 (in Russ.).

[18] Demin A.A., Karpunin A.A., Ganev Yu.M. Verification and validation methods for complex software systems. *Programmnye produkty i sistemy* [Software & Systems], 2014, no. 4, pp. 229–233 (in Russ.). DOI: http://dx.doi.org/10.15827/0236-235X.108.229-233

[19] Karpunin A.A., Ganev Yu.M., Chernov M.M. Methods of quality assurance in the design of complex software systems. *Nadezhnost i kachestvo slozhnykh system* [Reliability & Quality of Complex Systems], 2015, no. 2 (10), pp. 78–84 (in Russ.). EDN: UIQABT

[20] Vanroye N.K., Karpunin A.A. Analysis of the principles of business planning of the it project. *Informatsionnye tekhnologii v proektirovanii i proizvodstve* [Information Technologies of CAD/CAM/CAE], 2018, no. 4 (172), pp. 31–35 (in Russ.). EDN: YOFBDN

Vlasov A.I. — Cand. Sc. (Eng.), Assoc. Professor, Department of Design and Production Technology of Electronic Equipment, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Shirokov D.E. — Engineer, Scientific and Educational Center of Functional Micro/Nanosystems, BMSTU (2-ya Baumanskaya ul. 5, str. 1, Moscow, 105005 Russian Federation).

Please cite this article in English as:

Vlasov A.I., Shirokov D.E. A system design methodology using an executive unified visual modeling language using the example of hardware and software complexes for flight control in rocket and space technology. *Herald of the Bauman Moscow State Technical University, Series Instrument Engineering*, 2026, no. 3 (156), pp. 102–126 (in Russ.). EDN: HYDTTL