На правах рукописи

Кознов
Дмитрий Владимирович

МЕТОДОЛОГИЯ И ИНСТРУМЕНТАРИЙ ПРЕДМЕТНО-ОРИЕНТИРОВАННОГО МОДЕЛИРОВАНИЯ

05.13.11 — Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат диссертации на соискание учёной степени

доктора технических наук

Санкт-Петербург
2015

Работа выполнена на кафедре системного программирования Санкт-Петербургского государственного университета

Научный консультант:

ТЕРЕХОВ Андрей Николаевич, доктор физико-математических наук, профессор

Официальные оппоненты:

ПОЗИН Борис Аронович, доктор технических наук, профессор, ЗАО «ЕС-лизинг», технический директор

КАЛЯНОВ Георгий Николаевич, доктор технических наук, профессор, Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В. А. Трапезникова Российской академии наук, профессор

ВОДЯХО Александр Иванович, доктор технических наук, профессор, Федеральное государственное автономное образовательное учреждение высшего образования "Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И. Ульянова (Ленина)", профессор

Ведущая организация:

Федеральное государственное бюджетное учреждение науки Институт системного программирования Российской академии наук

Защита состоится 21 апреля 2016 года в 15 часов 30 минут на заседании диссертационного совета Д 212.232.51 на базе Санкт-Петербургского государственного университета по адресу: 198504, Санкт-Петербург, Университетский пр. 28.

С диссертацией можно ознакомиться в библиотеке Санкт-Петербургского государственного университета и на сайте СПбГУ: http://spbu.ru/science/disser/.

Автореферат разослан _____ 2016 года.

Учёный секретарь
диссертационного совета Демьянович Юрий Казимирович

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования. Идея использовать визуальные модели при разработке программ была высказана ещё Дж. фон Нейманом (J. von Neumann) в конце 40-х годов. В последние пятьдесят лет аналогия с чертёжным проектированием в строительстве, машиностроении и других инженерных областях вдохновляла исследователей в попытках создать надёжный метод разработки программного обеспечения (ПО) на основе чертежей программ — визуальных моделей. В этом направлении работали D. Ross, L. Constantine, T. DeMarco, M. Jack-son, E. Yourdan, P. Coad, D. deChampeaux, G. Booch, I. Jacobson, J. Rumbaugh, B. Se-lic, D. Harel, S. Shlaer, P. Kruchten, M. Fowler, G. Rossi и др. Их усилиями данный подход был детально разработан и внедрён в индустрию. В результате был создан унифицированный язык моделирования UML (Unified Modeling Language), являющийся в настоящее время общепринятым стандартом. Однако в последнее время эффективность UML подвергается серьёзной критике (см., например, исследование M. Petre): не вдаваясь в детали можно сказать, что при попытках использования UML в индустриальных компаниях не хватает соответствующего контекста. Обзор литературы показывает, что полностью стандартизовать визуальное моделирование не удаётся: на практике требуется существенная адаптация концепций, методов, языков и инструментов.

На решение этой проблемы направлено предметно-ориентированное моделирование (Domain-Specific Modeling, DSM). Данный подход предназначен для разработки ПО на основе визуальных моделей, ориентированных на потребности *узкой предметной области*. При этом визуальные модели приобретают нужный контекст, превращаясь в предметно-ориентированные, и в связи с этим допускают эффективную формальную обработку, в частности, генерацию программного кода. Создание программных средств поддержки для многочисленных предметно-ориентированных языков в рамках приемлемых затрат стало возможным ввиду активного развития инструментов поддержки DSM-подхода (MetaEdit+, GMF, Microsoft Modeling SDK и др.), включая средства расширения стандартных систем моделирования (открытые программные интерфейсы, шаблоны и пр.). К настоящему моменту разработано большое количество целевых DSM-решений. Предметно-ориентированное моделирование начинает активно использоваться в различных приложениях, например, при разработке корпоративных ИТ-архитектур (см. работы H. Agt, S. Ali, I. Alloush, R. Z. Frantz, V. Kovanovic), в робототехнике (см. стартовавший в 2014 году семинар «Workshop on Model-Driven Robot Software Engineering», а также работы Ю. Литвинова, Д. Мордвинова, Т. Брыксина).

Однако для широкого промышленного применения предметно-ориентированного моделирования важно довести разрозненные исследования проблем разработки DSM-решений до единой методологии. При этом необходимо выполнить

стандартизацию и унификацию процесса разработки и сопровождения DSM-решения и анализ рисков, специфицировать итоговую поставку (Final Deliveries, Final Work Products) DSM-проекта, предложить методы реализации дополнительных функциональных компонент, отсутствующих в инструментах поддержки DSM-подхода. Таким образом, встал вопрос о создании новой единой методологии, которая позволит эффективно поддерживать промышленные DSM-решения на всех стадиях их жизненного цикла и расширить спектр применения предметно-ориентированного моделирования. В контексте разработки данной методологии актуальны следующие задачи: (i) создание моделей и алгоритмов для реализации дополнительных функциональных компонент (обеспечение качества визуальных спецификаций, версионный контроль, работа с большими моделями); (ii) разработка на основе DSM-подхода методов анализа и проектирования ПО различных видов; (iii) применение DSM для отдельных видов деятельности процесса разработки ПО, в частности, для разработки документации. Также представляют интерес практические примеры использования предметно-ориентированного моделирования на основе единой методологии.

Степень разработанности темы работы. Предметно-ориентированное моделирование развито в работах J.–P. Tolvanen, S. Kelly, J. Greenfield, K. Short, R. C. Gronback, S. Cook, K. Czarnecki, U. Eisenecker, Ö. Turhan, J. Cabot и др. Данные исследования снижают степень общности проблемы практического применения визуальных моделей, делая её разрешимой, и обращаются к более технической стороне моделирования. Однако всё ещё сохраняется значительный разрыв между методами моделирования и инструментарием. При этом существующие подходы разрозненны и не решают комплексных задач, возникающих при практическом применении DSM. Большинство из них являются неформальными (исследования J. P. Tolvanen и S. Kelly), или носят энциклопедический характер (K. Czarnecki и U. Eisenecker), или исходят из соображений, что программный инструментарий поддержки DSM является вспомогательной задачей (J. Greenfield, K. Short с коллегами). Нехватка концептуальных разработок в данной области косвенно подтверждается отсутствием современных обзоров, а также смешиванием DSM с DSLs (Domain Specific Languages, предметно-ориентированные текстовые языки программирования) — см., например, обзор 2013 года S. Erdweg с коллегами «The state of the art in language workbenches».

Говоря о развитии предметно-ориентированного моделирования в России, следует обратить внимание на работы Я. М. Барзиня, А. В. Карабегова, Н. Н. Мансурова, посвящённые языку SDL (Specification and Description Language), и исследования А. А. Шалыто, изучающего автоматное программирование и его приложения. Необходимо указать на коллектив А. Н. Терехова, который занимается предметно-ориентированным моделированием с середины 80-х годов (технологии RTST, Real,

Real-IT, QReal, QReal:ROBOTS). Целесообразно упомянуть о работах Ф. А. Новикова, а также об исследованиях межвузовского коллектива в составе Л. Н. Лядовой, А. О. Сухова и др. Кроме того следует указать на работы «Межвузовского академического центра компетенций по архитектуре предприятия «EA Lab», использующего DSM при разработке бизнес- и ИТ-архитектур.

Одной из центральных задач при разработке DSM-решений является обеспечение качества моделей. В этой области существует множество исследований (см. обзоры F. Lucas с коллегами и P. Mohagheghi с коллегами). В них акцентируется проблема нехватки полноценных программных средств, реализующих данные методы, интеграционные сложности и недостаточная масштабируемость. Не менее важной является задача разработки, анализа и контроля качества больших моделей. Данная задача частично исследована в работах B. Berenbach, F. Weil с коллегами, Д. Бабурина с коллегами. Однако в этих работах представлены лишь частные решения. Задача версионного контроля визуальных спецификаций до сих пор не решена, в отличие от версионирования исходного кода. Существует значительное количество алгоритмов для нахождения разницы (Diff) и слияния (Merge) XML-файлов (3DM, So6, DeltaXML и др.), однако при практическом использовании они нуждаются в модернизации. Имеется инструмент EMF DiffMerge, однако он поддерживает лишь простые случаи и ориентирован на создание средств, работающих только с моделями в формате Ecore. Очевидно, что во всех этих областях требуются дополнитель-ные исследования.

Следует отметить, что в области DSM все ещё существует разрыв между общими методами (исследования J. P. Tolvanen и S. Kelly с коллегами, J. Greenfield и K. Short с коллегами) и исследованиями, ориентированными на конкретные классы задач. Последние разрозненны и зачастую сводятся к изложению опыта отдельных проектов. При этом исследования в области DSM концентрируются, главным образом, на генерации кода по моделям (см., например, исследования J. P. Tolvanen и S. Kelly с коллегами). Возможности применения DSM другими способами исследованы существенно меньше. В частности, отсутствуют исследования, использующие DSM для разработки документации.

Таким образом, имеется потребность в комплексном исследовании вопросов практической применимости DSM в индустриальных проектах и создании единой методологии.

Объектом исследования диссертационной работы являются модели, методы, алгоритмы, языки, технологии и программные средства предметно-ориентированного моделирования, предназначенные для проектирования и анализа алгоритмов и программ.

Целью данной работы является создание методологии для поддержки разработки сложных предметно-ориентированных программных решений на основе визуальных моделей. Для достижения этой цели были сформулированы следующие **задачи**.

1. Исследовать проблемы предметно-ориентированных разработок и выделить основные шаги и элементы, нуждающиеся в формализации и поддержке.
2. Разработать методологию предметно-ориентированного моделирования, предоставляющую средства для спецификации итоговой поставки DSM-проекта, описывающую дополнительные функциональные компоненты (не реализованные существующими техническими средствами), включающую средства для создания процесса разработки и сопровождения DSM-решения, а также для анализа рисков.
3. Разработать и проверить на практике методы решения следующих технологических задач:
   - создание моделей и алгоритмов разработки дополнительных функциональных компонент предметно-ориентированных решений: для работы с большими моделями, для обеспечения качества предметно-ориентированных моделей, для слияния (Merge) моделей при работе в Интернете;
   - алгоритмизация анализа и проектирование DSM-решений для отдельных классов ПО;
   - применение DSM для разработки промышленной документации.
4. Оценить эффективность использования предложенных решений на широком классе практических задач.

Постановка цели и задач исследования соответствует следующим пунктам паспорта специальности 05.13.11: модели, методы и алгоритмы проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования (пункт 1); системы управления базами данных и знаний (пункт 4); оценка качества, стандартизация и сопровождение программных систем (пункт 10).

Методология и методы исследования. Методология исследования базируется на идеях и подходах программной инженерии по разработке методов создания ПО. Используются также концепции визуального и предметно-ориентированного моделирования.

В работе использовались методы MSF и Scrum, а также стандарт CMM. Для спецификации предметно-ориентированных языков применялось метамоделирование, расширенные графические грамматики и язык XML. Были использованы методы Ерзабека-Бассета и Feature Diagrams, подход к поиску клонов в ПО (Software Clone Detection), метод и-карт (Mind Maps). Применялись следующие стандарты: UML, BPMN, SDL, Ecore, OCL, ATL, QVT, DocBook. Для реализации предложенных в работе результатов использованы следующие технологии: Microsoft Visio, Eclipse, Microsoft

Modeling SDK, GMF, ARIS, KIELER Eclipse project, Dresden OCL Toolkit, ATL, CloneMiner. Реализация приложений была выполнена с помощью языков программирования C#, Java, JavaScript, Visual Basic, Haxe.

Положения, выносимые на защиту.

1. Предложена методология предметно-ориентированного моделирования, предназначенная для разработки инструментов анализа и проектирования программного обеспечения на основе визуальных моделей и предоставляющая средства для спецификации итоговой поставки DSM-проекта, описывающая дополнительные функциональные компоненты, не реализованные существующими техническими средствами, включающая средства для создания процесса разработки и сопровождения DSM-решения, а также для анализа рисков.
2. Разработан алгоритм слияния двух версий и-карт (Mind Maps) в контексте групповой разработки требований к ПО (первичный сбор и анализ требований) средствами Интернет после обрыва и восстановления сетевого соединения.
3. Создана модель v2v-трансформаций для автоматизированной разработки диаграммных сервисов с целью решения проблем навигации и сопровождения больших моделей.
4. Предложен метод контроля качества (корректности) предметно-ориентированных спецификаций, включающий средства для автоматизированного создания валидаторов спецификаций на основе OCL-ограничений (Object Constraint Language).
5. Предложена модель для проектирования и разработки продуктов семейств программно-аппаратных систем, основанных на программных конструкторах и конфигурировании ПО целевых продуктов с помощью компоновки аппаратной части. Модель включает предметно-ориентированный язык THCL (Telecommunication Hardware Configuration Language) и архитектуру технологии графического проектирования.
6. Создан метод FSS (Formal Services Specification), предназначенный для анализа и проектирования программных систем, реализующих электронный доступ к государственным и публичным услугам.
7. Предложена модель КИТ-решения (Корпоративная ИТ-архитектура), предназначенная для разработки и сопровождения ИТ-архитектур крупных компаний.
8. Создан метод DocLine для разработки и сопровождения документации ПО на основе повторного использования. Предложен предметно-ориентированный язык DRL/GR (Documentation Reuse Language/Graphical Representation) для проектирования документации линеек программных продуктов. Создан

алгоритм обнаружения повторов и рефакторинга документов на основе техники поиска клонов в ПО.

Научная новизна представленных результатов заключается в следующем.

1. *Мет одология предмет но-ориент ированного моделирования.* Новым является комплексный системный подход к разработке предметно-ориентированных решений, отличающийся от неформальных (исследования J. P. Tolvanen и S. Kelly) и энциклопедических подходов (K. Czarnecki и U. Eisenecker), и в то же время не превращающийся в руководство по использованию конкретной технологии (подходы Eclipse Modeling Project и Microsoft Modeling SDK). Также новым является обобщение и перенос DSM на различные предметные области, что отсутствует в других концептуальных подходах.
2. *Новый алгорит м слияния нескольких версий и-карт* отличается от существующих (3DM, So6, DeltaXML и др.) неравнозначностью локальной и серверной версий, возможностью пользовательского разрешения конфликтов, а также генерацией итоговой версии по текущей серверной, а не исходной, версии. Представляет новизну предложенный подход к идентификации (matching) элементов в сливаемых версиях, которые имеют общее происхождение. Существующие подходы (3DM, работа P. Shvaiko с коллегами и др.) производят идентификацию в произвольной ситуации, то есть, фактически, решают другую задачу.
3. *Предлож енная модель v2v-т рансформаций* отличается от существующих применением трансформаций к представлениям моделей, а не к моделям, как это принято традиционно. Подобные исследования отсутствуют. Предложенная модель отличается от алгоритмов раскладки графов и моделей (см., например, систему Graphviz или технологию KIELLER) тем, что способна изменять состав выборки из модели, в то время как существующие алгоритмы манипулируют лишь фиксированными выборками. Также новой является технология разработки средств работы с большими моделями, автоматизирующая создание целевых навигационных сервисов, в то время как в других работах (B. Berenbach, Д. Бабурин с коллегами и др.) предлагаются лишь частные решения.
4. *Мет од конт роля качест ва (коррект ност и) предмет но-ориент ированных спецификаций* отличается решением задачи обеспечения корректности больших предметно-ориентированных моделей. Существующие подходы (B. Berenbach, F. Weil и др.) рассматривали большие модели, но только для фиксированных языков моделирования. Имеются методы верификации/валидации предметно-ориентированных моделей (см. работы F. Zalila с коллегами, B. Combemale с коллегами и др.), но они не применимы для работы с большим индустриальными моделям. Новой является идея контроля корректности не только самих моделей, но и их представлений (диаграмм, расположения элементов модели в папках репозитория и пр.).

5. *Модель средст в разработ ки семейст в программно-аппарат ных сист ем*. Новой является идея автоматической конфигурирования и сборки ПО продуктов линейки программно-аппаратных систем на основе спецификации аппаратной части, задаваемой с помощью визуальных моделей. Такой класс задач в контексте линеек продуктов до сих пор не рассматривался.
6. *Модель КИТ-решения*. Новым является рассмотрение проектов по разработке средств управления корпоративными ИТ-архитектурами как ИТ-проектов. Существующие методы разработки ПО (RUP, CMMI, Scrum, MSF и др.) не рассматривают разработку таких проектов, стандарты разработки архитектуры предприятия (TOGAF, GERAM, FEA и др.) не применяют методы программной инженерии к таким проектам. Новым также является использование для спецификации результатов таких проектов формальной модели ИТ-решения (метод MSF).
7. *Мет од FSS*. Новизна метода заключается в дополнении модели бизнес-процессов (BPMN) моделью документации, описывающей иерархию документов на основе Feature Modeling. Подобные расширения BPMN отсутствуют в литературе. Также новой является идея метафор Web-визуализации. Ближайшим аналогом таких метафор являются результаты исследовательской области Model-Based Interface Development (разработка типовых экранные форм, автоматически генерируемых, например, по схеме базы данных); однако данные результаты не распространялись на сферу разработки ПО для электронных и публичных услуг.
8. *Мет од DocLine, язык DRL/GR, алгорит м поиска повт ора и рефакт оринга документ ации*. Новизна DocLine заключается в интеграции планового повторного использования в процесс разработки документации ПО, что отсутствует у существующих подходов (DocBook, DITA и др.). Новизна языка DRL/GR заключается в объединении доменного анализа и конфигурирования обобщённой модели семейства при создании документации для конкретного продукта на основе DSM (это отсутствует в подходах K. Czarnecki и U. Eisenecker, J. Green-field и K. Short и др.). Предложенный язык является адаптацией Feature Diagrams, которые до этого использовались только для доменного анализа и не применялись для разработки документации. Новизна алгоритма поиска повторов и рефакторинга заключается в применении метода поиска клонов в программном обеспечении (Software Clone Detection) к задаче поиска повторов в XML-документах и использовании полученных результатов для рефакторинга документации. Единственным аналогом являются работы A. Wingkvist и коллег, которые, однако, не реструктурируют документы на основе найденных клонов.

Теоретическая и практическая значимость работы. Полученные результаты обобщают достижения предметно-ориентированного моделирования: определяются такие понятия как DSM-решение, методика, платформа, вводится модель разработки

DSM-решения и модель рисков. Это позволяет исследовать предметно-ориентированное моделирование вне непосредственной связи с конкретными DSM-платформами, а также применять предложенную методологию при исследованиях и разработке DSM-решений для широкого спектра задач как в области разработки ПО, так и для создания систем управления знаниями в смежных областях. Кроме того, предложенная методология, а также подходы и решения для отдельных компонент методологии определяют направления дальнейших исследований в рамках создания эффективных промышленных технологий разработки ПО.

Результаты, полученные в ходе диссертационной работы, были апробированы в проектах компаний ЗАО «ЛАНИТ-ТЕРКОМ», ООО «Смарт Аркитект», АНО КМЦ «Бизнес-Инжиниринг», ООО «Digital Design», при разработке аналитического портала правительства города Москвы, а также в международном исследовательском проекте «Improving Social Services» (№ 2010-021-SE396). Получено пять актов о внедрении результатов диссертационной работы в индустрию.

Достоверность результатов работы подтверждается инженерными экспериментами, исследованиями существующих проектов в области разработки и применения DSM-решений, применением предложенных методов для разработки различных промышленных активов с анализом полученных результатов.

Результаты работы были доложены на 10 международных научных конферен-циях: International Conference on Knowledge Management and Information Sharing (KMIS 2008, 2011), Ershov Informatics Conference (PSI 2001, 2015), International Sym-posium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2004, 2005, 2008), Computers and Advanced Technology in Education (2011), 4th IFIP TC 2 Central and East European Conference on Software Engineering Techni-ques (CEE-SET 2009), European Conference on Software Maintenance and Reengine-ering (CSMR, 2002), Software Engineering Conference (Russia) (SEC(R), 2008).

Дополнительной апробацией результатов является поддержка исследований, представленных в диссертации, следующими грантами: РФФИ, проект № 14-01-00407-а «Предметно-ориентированное моделирование при разработке и анализе архитектур бизнес-предприятий»; РФФИ, проект № 12-01-00415-а «Визуальное моделирование электронных государственных услуг»; ENPI CBC 2007-2013, проект № 2010-021-SE396 «Improving Social Services»; РФФИ, проект № 11-01-00622-а «Циклическая разработка в моделировании ПО»; Совет Министров Северных Стран, проект № NCM-RU-PA-2009/10661 «Software Engineering Learning»; РФФИ, проект № 08-01-00716-а «Разработка технической документации на основе повторного использования»; грант компании Hewlett-Packard, «Comapping for teaching»; МинОбрНауки; проект № 02.442.11.7495 «Визуальное моделирование в проектировании аппаратной части телекоммуникационных систем»; РФФИ, проект № 05-01-00907-а «Автоматизирован-

ная генерация пользовательского интерфейса распределённых информационных систем»; МинОбрНауки, проект № 02.442.11.7211 «Визуальное моделирование в проектировании информационных систем телевидения»; мэрия СПб, проект № PD05-2.0-280 «Создание UML 2.0 студии»; мэрия СПб, проект № PD04-2.0-276 «Разработка и специализация средств визуального моделирования»; мэрия СПб, проект № PD03-2.0-116 «Управление проектами: разработка ПО, бизнес, творчество»; МинОбрНауки, проект № РД02-2.8-145 «Визуальное моделирование в управлении разработкой программного обеспечения».

Публикации по теме диссертации. Все результаты диссертации опубликованы в 67 печатных работах, зарегистрированных в РИНЦ, из них 1 монография (единоличная). 21 статья издана в журналах из «Перечня российских рецензируемых научных журналов, в которых должны быть опубликованы основные научные результаты диссертаций на соискание учёных степеней доктора и кандидата наук», рекомендованного ВАК. Там же опубликованы все основные результаты диссертации. 7 статей опубликованы в изданиях, входящих в базу цитирования SCOPUS, из них 3 — в изданиях, входящих в базы цитирования Web of Science. По результатам диссертации было получено 1 свидетельство о регистрации программы для ЭВМ. Сверх указанного было опубликовано 2 учебных пособия, прошедших научное рецензирование.

Вклад автора в публикациях, написанных в соавторстве, распределён следующим образом. В работе [2] автору принадлежит формализация метода и архитектура программной реализации в рамках технологий GMF/ATL/KIELLER. Соавторы выполнили реализацию предложенной автором архитектуры, эксперименты, сформулировали направления дальнейшего применения предложенного подхода. В работе [4] автору принадлежит идея КИТ-модели, также он выполнил работы по её формализации. Соавторы предложили классификацию рабочих продуктов и вместе с автором участвовали в апробации предложенной модели. В работе [5] автор предложил применять повторное использование для произвольной технической документации (не только для документации линеек ПО, как это рассматривалось ранее). Соавторы спроектировали и реализовали алгоритм поиска нечётких клонов. В [6] автор выполнил всё исследование, соавтором были выполнены эксперименты. В [7] автор разработал архитектурную модель, соавторы предложили ряд компонентов этой модели (внутренние редакторы и отчёты), а также выполнили программную реализацию. В [9] автору принадлежит метод исследования, выбор продуктов для исследования и написание текста работы. Соавторы выполнили исследование по методу, предложенному автором. В [10] автор выполнил строгую типизацию элементов языка ОРГ-Мастера. Определение прагматики, а также программная реализация предложенных идей принадлежит соавторам. В [12] автор предложил и разработал метод FSS, соавторы выполнили разработку примеров из области городского хозяйства Санкт-Петербурга, а также ре-

ализацию предложенных автором метафор Web-визуализации. В [13] автору принадлежит постановка задачи о поиске повторов в документации. Автор также разработал алгоритм поиска точных повторов в документации и выполнил интеграцию результата поиска клонов с рефакторингом документации. Соавторы выполнили программную реализацию алгоритма, провели эксперименты. В [15] автором была предложена формализация задачи слияния и-карт при разработке требований к ПО, спроектированы изменения 3DM-алгоритма. Соавторы выполнили спецификацию обработки конфликтов, реализацию алгоритма и его тестирование. В [16] автору принадлежит идея метода автоматического отслеживания изменений в пользовательской документации Web-приложений. Соавторы выполнили формализацию и программную реализацию метода, а также эксперименты. В [17] автору принадлежит постановка задачи и формализация основных результатов. Метод исследования и его пилотная программная реализация были предложены соавторами. В [18] автору принадлежат основные идеи DocLine. Соавтору принадлежит разработка и формализация подхода в целом, создание архитектуры пакета инструментальных средств, а также программная реализация. В [20] автору принадлежит постановка задачи исследования, разработка общей архитектурной модели. Уточнение деталей и программная реализация были выполнены соавторами. В [21] автору принадлежит разработка метода контроля качества предметно-ориентированных моделей. Соавтор выполнил детальное проектирование генератора валидаторов. Более подробная характеристика личного вклада Д. В. Кознова содержится в тексте диссертационной работы.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения, списка литературы и приложений. Полный объем диссертации составляет 430 страниц текста, 104 рисунка, 13 таблиц и 3 приложения. Список литературы содержит 452 наименования.

СОДЕРЖАНИЕ РАБОТЫ

Первая глава содержит анализ современного состояния научных исследований в данной области и формирует концептуальный и теоретический базис разработанной методологии.

Вторая глава посвящена разработке методологии предметно-ориентированного моделирования (далее — Методологии). На рис. 1 представлена концептуальная схема Методологи, на рис. 2 — сценарий разработки DSM-решения, определяемый Методологией. Методология включает совокупность методов и шагов по планированию, разработке и сопровождению предметно-ориентированных решений. Предлагается специфицировать поставку DSM-проекта (для этого вводится понятие DSM-решения), выделить и спроектировать дополнительные функциональные компоненты DSM-решения, не поддерживаемые DSM-платформами — средства версионного кон-

троля (слияние моделей, поиск разницы) и обеспечения качества визуальных спецификаций, средства поддержки работы с большими моделями, а также средства интеграции с другими программными системами. Предлагается проанализировать проектные риски (для этого определяется модель рисков). Итоговые DSM-решения разделяются на две категории: (i) средства разработки ПО (для определённых сообществ или для некоторого коллектива); (ii) средства управления знаниями на основе предметно-ориентированного моделирования для произвольных областей — бизнес-инжиниринга, робототехники, образования и т.д.

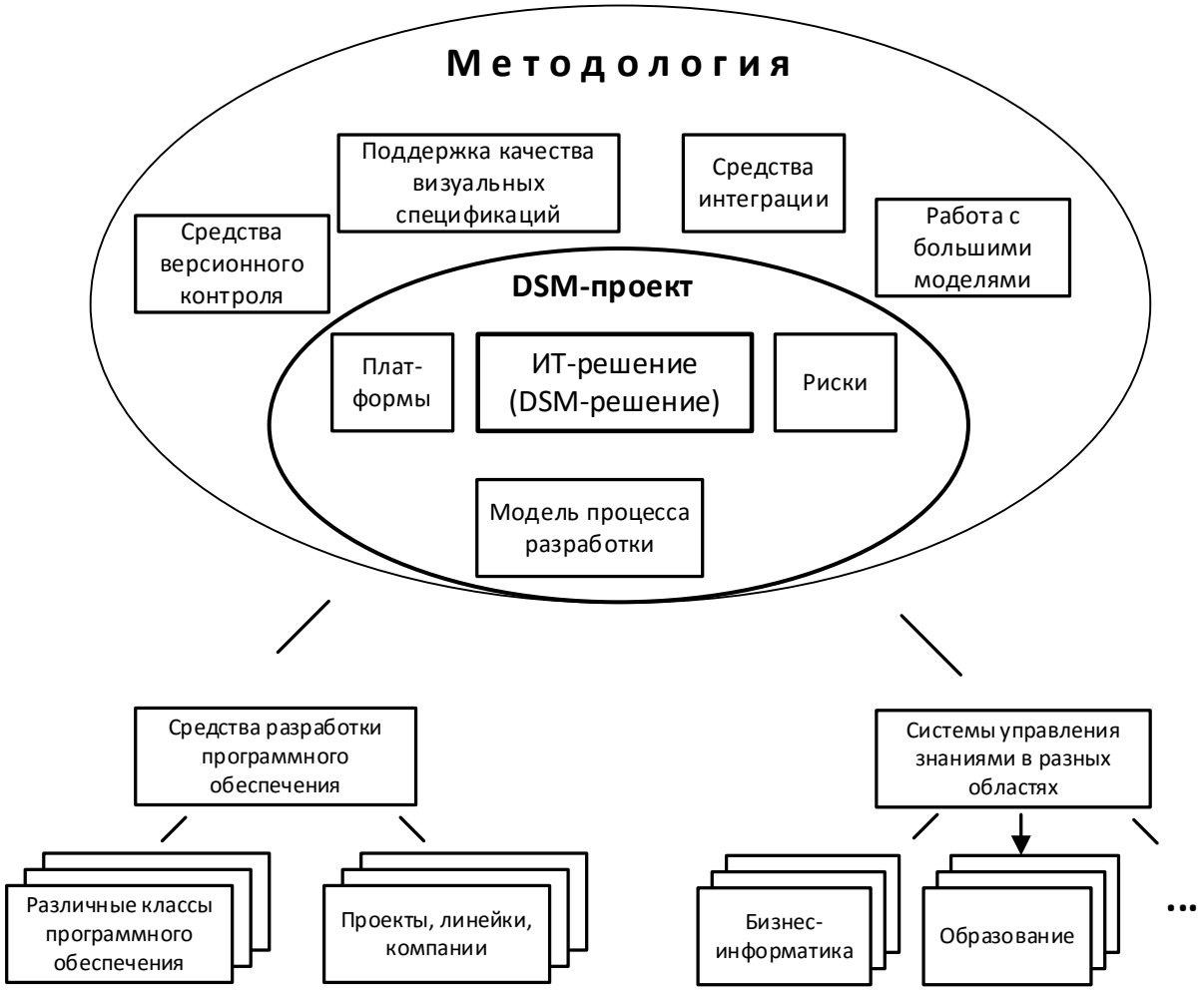

Рис.1 Концептуальная схема Методологии

Важно отметить, что DSM-проекты создаются для коллективов и сообществ, выполняясь, например, внутри компании для удовлетворения её собственных нужд. Поэтому при создании DSM-решений общепринятых практик разработки ПО недостаточно, и требуются дополнительные средства. Именно таким средством является предлагаемая модель процесса разработки (рис. 2). Кратко опишем эту модель.

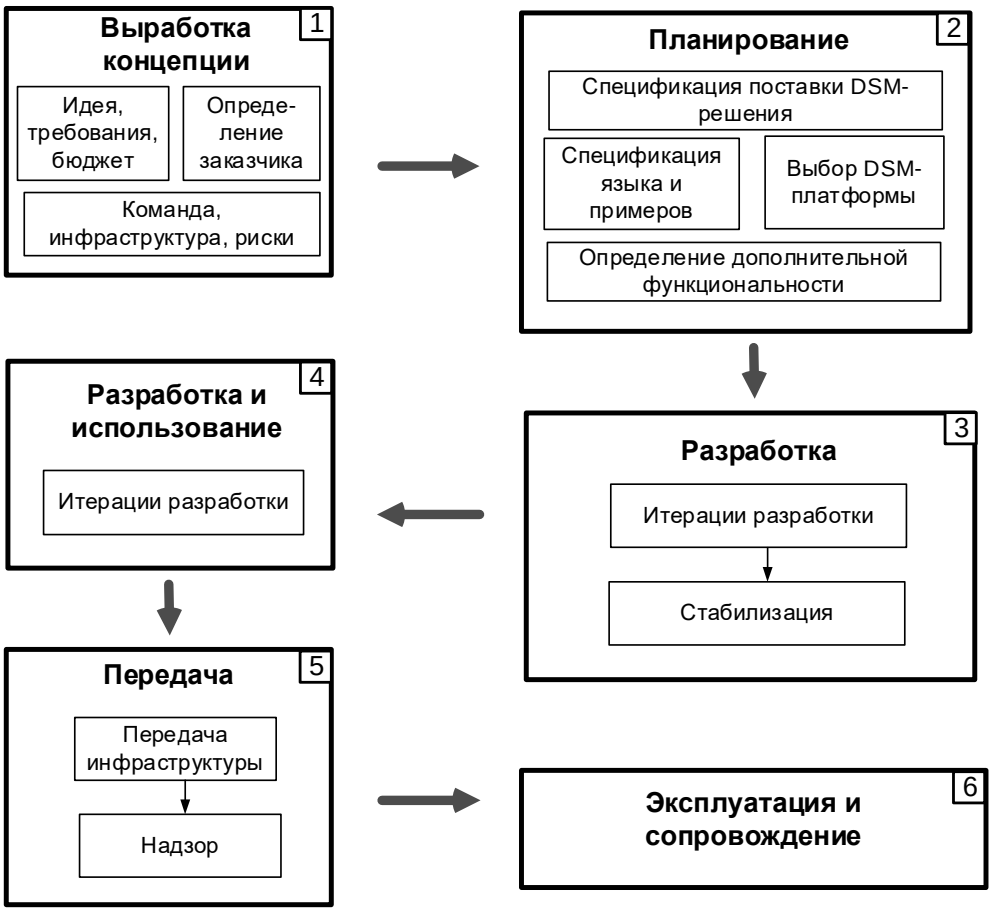

Рис. 2. Сценарий разработки DSM-решения (модель процесса разработки)

Выработка концепции. На этом этапе происходит создание и оформление идеи DSM-решения, определяются верхнеуровневые требования, бюджет, а также заказчик проекта; создаётся команда и инфраструктура проекта, определяются риски. *Планирование* — спецификация итоговой поставки DSM-решения, разработка и формализация языка моделирования, выбор DSM-платформы. Определение дополнительной функциональности решения: во-первых, рассматриваются вопросы качества — цена ошибок в моделях, необходимость дополнительных средств; во-вторых, определяются требования к версионированию моделей (в частности, процедуре слияния); в-третьих, решаются вопросы работы с большими моделями — средства декомпозиции, навигации, создания отчётов (в частности, Web-отчётов) и поддержки дисциплины/процесса разработки/сопровождения моделей; в-четвертых, рассматривается интеграция DSM-решения с окружением. *Разработка* — создание работоспособной версии DSM-решения, стабилизация версии и передача её пользователям. Разработка выполняется итеративно с использованием метода Scrum. *Разработка и использование* — дальнейшая разработка решения, совмещённая с его использованием. DSM-решения часто создаются в контексте других проектов, и результаты требуются в кротчайшие сроки. *Передача* — сдача проекта и перевод его в режим поддержки и

сопровождения. *Эксплуатация и сопровождение* — обеспечение непрерывной поддержки DSM-решения: исправления ошибок, реализация новой функциональности, своевременный перенос решения на новые версии DSM-платформы, миграция моделей пользователей.

Для каждого шага/элемента Методологии предлагается набор методов, моделей и алгоритмов. Предполагается, что дальнейшие исследования в области предметно-ориентированного моделирования могут быть структурированы в рамках предложенной Методологии (в частности, могут быть созданы дополнительные модели процесса разработки DSM-решения).

Третья глава описывает созданные автором новые модели, методы и алгоритмы, посвящённые разработке дополнительных функциональных компонент DSM-решений, определённых в рамках Методологии: работа с большими моделями, версионный контроль, поддержка качества визуальных спецификаций. Представленные результаты облегчают практическое решение данных задач и показывают направление дальнейших исследований. Рассмотрим эти результаты.

Модель v2v-трансформаций. Данная модель посвящена созданию средств работы с большими моделями; необходимость этих средств была обоснована в Методологии. Пусть у нас имеется метамодель $\mathcal{M}$, определяющая некоторый язык моделирования. Определим метамодель как $\mathcal{M} = \langle \mathcal{T}, \mathcal{I}, \mathcal{A} \rangle$, где $\mathcal{T}$ — это множество основных элементов метамодели (типов сущностей и ассоциаций); $\mathcal{I}$ — отношения наследования между элементами $\mathcal{T}$, то есть $\mathcal{I} \subset \mathcal{T} \times \mathcal{T}$; $\mathcal{A}$ — множество атрибутов элементов из $\mathcal{T}$. Пусть $\mathrm{T} = \mathrm{t}_1, \ldots \mathrm{t}_n$ — это набор статических представлений (видов диаграмм) данного языка моделирования.

Определим модель как $\mathrm{M} = \langle \mathrm{E}, \mathrm{A}, \varphi \rangle$, где E — это множество объектов и связей модели (экземпляров элементов из $\mathcal{T}$), A — значения атрибутов объектов, а $\varphi: \mathrm{E} \longrightarrow \mathcal{T}$ является функцией, сопоставляющей каждому элементу модели соответствующий тип в метамодели. Будем обозначать как $\mathrm{M} \succ \mathcal{M}$ тот факт, что модель M является корректным экземпляром метамодели $\mathcal{M}$ (функция φ этого не обеспечивает, так как элементы соответствующих типов могут быть неправильно соединены, могут также не выполняться различные ограничения на модели, сформулированные с помощью OCL).

Пусть у нас есть некоторая выборка элементов из модели. Тогда *параметры отображения* — это задание наличия в выборке атрибутов сущностей, например, показывать или нет атрибуты и методы класса, а если показывать, то следует ли отображать значение видимости и т.д.

Определим множество всех возможных значений параметров отображения для элемента метамодели $\theta \in \mathcal{M}$ на выборках вида t, где $\mathrm{t} \in \mathrm{T}$, как $\mathrm{DO}_{\theta,\mathrm{t}}$ (DO является сокращением от Display Options). Если $\mathrm{DO}_{\theta,\mathrm{t}} = \emptyset$, значит, элементы типа θ в выборки данного типа не входят. Набор конкретных значений параметров отображения

элемента модели $e \in M$ в выборке вида t будем обозначать $\mathrm{do}_{e,t}$. При этом $\mathrm{do}_{e,t} \in \mathrm{DO}_{\theta,t}$, где $\theta = \varphi(e)$. Если $\mathrm{DO}_{\theta,t} = 1$, значит, элементы вида θ отображаются в выборках типа t единственным образом.

$$\mathrm{DO}_{\mathcal{M},t} = \bigcup_{\theta \in \mathcal{M}} \mathrm{DO}_{\theta,t}, \quad \mathrm{DO}_{\mathcal{M}} = \bigcup_{t \in T} \mathrm{DO}_{\mathcal{M},t}.$$

Обозначим через $\mathcal{VM}$ метамодель *динамических представлений*, определяющую структуру выборок из моделей, с которыми пользователь может работать посредством диаграмм. Динамическое представление — это выборка из модели вида t, где $t \in T$, которая может содержать повторы элементов модели, и каждый элемент этой выборки содержит параметры отображения. Фактически, $\mathcal{VM}$ дополняет элементы $\mathcal{M}$ возможными параметрами отображений в соответствии с видами статических представлений (элемент метамодели на диаграммах различных видов может иметь различные виды параметров отображений, например, класс на диаграмме классов и диаграмме объектов). Формально определим $\mathcal{VM}$ как $\langle \mathcal{M}, T, DO \rangle$, где $DO \subset \mathcal{M} \times T \times \mathrm{DO}_{\mathcal{M}}$. Определим динамическое представление следующим образом:

$$v = \langle t, M, \Delta, \psi, \beta \rangle,$$

где $t \in T$ (то есть динамическое представление определяется для выборок вида t и таким образом t будем называть видом v), Δ является множеством элементов динамического представления модели M, $\psi: \Delta \longrightarrow E$ является функцией и позволяет определить, какому элементу модели соответствует данный элемент представления, $\beta: \Delta \longrightarrow \mathrm{DO}_{\mathcal{M},t}$ является функцией, действующей из множества элементов представления в множество всех возможных параметров отображения в выборках вида t, при этом верно следующее утверждение: $\forall d \in \Delta \ (\beta(d) \in \mathrm{DO}_{\psi(d),t})$. Таким образом, динамическое представление является выборкой из модели с повторами, и, кроме того, каждый из элементов этой выборки имеет набор параметров отображения. Множество всех возможных динамических представлений вида t для метамодели $\mathcal{M}$ определим следующим образом:

$$V_t = \bigcup_{M \succ \mathcal{M}} \{ v \mid \text{где вид } v \text{ равен } t \}.$$

Диаграмма отличается от динамического представления тем, что всем элементам последнего добавляются *графические свойства* (Graphical Options) – координаты фигуры, цвет и толщина линий фигуры и фона, параметры шрифтов и т.д. Пусть имеется диаграмма вида t, будем обозначать графические свойства элемента диаграммы d как $\mathrm{go}_{d,t}$. Множество всех возможных графических свойств элементов диаграмм вида t определим так:

$$GO_t = \bigcup_{d \in V_t} \{go_{d,t}\}.$$

Рассматривая v2v-трансформации выделим следующие случаи: трансформации, действующие на фиксированном множестве элементов модели/диаграммы (далее — трансформации первого вида) и трансформации, применяемые ко всей диаграмме целиком (далее — трансформация второго вида). Примерами для первого случая могут служить все трансформации, применяемые к одному элементу (таких трансформаций оказывается большинство). Существенно меньше трансформаций, которые можно задать для пары элементов. В качестве примера можно указать на случай, когда требуется определить все возможные пути в графе между двумя элементами (например, двумя состояниями в диаграммах состояний и переходов или для двух классов на диаграмме классов). Для второго случая приведём пример, когда для классов всей диаграммы предложено изображать атрибуты и не изображать методы. При этом неизвестно, сколько именно будет классов. Отметим, что трансформации обоих видов не меняют модель.

Формально определим v2v-трансформацию для первого случая. Определим подмножество модели M, состоящее из всех элементов, соответствующих некоторому типу метамодели θ: $M_\theta = \{e \in M: \varphi(e) = \theta\}$. Будем обозначать декартовое произведение нескольких таких множеств, определённое для одной модели M, следующим образом:

$$\prod_{i=1..n} M_{\theta_i}.$$

Определим функцию ζ_n на отрезке натуральных чисел от 1 до n: $\zeta_n: 1..n \longrightarrow \mathcal{T}$. Для фиксированной метамодели $\mathcal{M}$, $t \in T$ и функции ζ_n определим трансформацию первого вида как функцию следующего вида:

$$tr1_{\mathcal{M},t,\zeta_n} : \prod_{i=1..n} M_{\zeta_n(i)} \longrightarrow V_t,$$

при этом $M \succ \mathcal{M}$ и принимает все возможные значения. Необходимо отметить, что трансформация может быть применена как к элементу модели (и тогда для него создаётся соответствующая выборка вида t, которая потом отображается на отдельной диаграмме), так и к элементу диаграммы. В последнем случае вид выборки не важен, поскольку в качестве аргумента нас не интересуют параметры отображения элементов модели, к которым применяется трансформация: например, если требуется отобразить для класса значение его атрибутов, включая значения видимости и значения по умолчанию, то не имеет значения, как данные атрибуты отображались до этого. Сделаем ещё одно замечание: трансформация действует на элементах модели с повторениями, так как можно, например, запросить построение всех путей из элемента модели в него же.

Теперь определим трансформацию второго вида как функцию следующего вида:

$$\mathrm{tr2}_{\mathcal{M},t_1,t_2} : V_{t_1} \longrightarrow V_{t_2},$$

где $t_1, t_2 \in T$. Таким образом, трансформация второго вида действует на все динамическое представление вида t_1, преобразуя его в представление вида t_2. В большинстве случаев $t_1 = t_2$.

Определим *навигационный сервис* как следующую тройку:

- v2v-трансформация;
- спецификация элементов пользовательского интерфейса для задания параметров сервиса (диалоговое окно) и вызова сервиса: пункт контекстного меню элемента на диаграмме, кнопка на панели инструментов, пункт главного меню и пр.;
- алгоритм раскладки (layout algorithm), который изменяет графические свойства, но не меняет состав динамического представления.

Для задания v2v-трансформаций предложено использовать язык трансформаций ATL. В работе представлена реализация подхода с помощью DSM-платформы GMF. Для выполнения трансформаций использована технология ATL, для выполнения раскладок — технология KIELLER. С помощью получившейся технологии был реализован редактор классов и 12 основных трансформаций диаграмм классов. Были проведены измерения эффективности этих трансформаций по следующим критериям: производительность, объем кода, сложность кода. Производительность измерялась для диаграммы размером в 100 классов и 90 связей и составила 2–3 секунды для каждого навигационного сервиса. Основная работа уходила на подготовку и загрузку/выгрузку данных, ATL-трансформация исполнялась меньше секунды. Объем кода для каждой трансформации составил не более 50 строк на ATL. В случае разработки навигационных сервисов «вручную» потребовались бы значительные усилия на интеграцию такого кода с кодом сгенерированного редактора, и, кроме того, реализация содержала бы много однотипного кода по работе с коллекциями, который плохо поддаётся повторному использованию с помощью стандартных средств. Сложность разработки трансформаций измерялась с помощью метрики, имеющей следующие значения: использование только базовых знаний ATL (легко, Л), необходимость знания стандартных конструкций ATL (средне, Ср), необходимость глубокого знания ATL (сложно, С). Из 12 созданных трансформаций 2 имели сложность Л, 6 — сложность Ср, 4 — сложность Сл.

Алгоритм слияния и-карт. Данный алгоритм является составной частью стратегии версионного контроля, необходимость которого для DSM-решений обоснована в Методологии. В данной работе использовался известный 3DM-алгоритм слияния XML-файлов. Алгоритм был модернизирован следующим образом: (i) было реализовано

разрешение конфликтов слияния по умолчанию; (ii) конфликт Delete/Move автоматически разрешался в пользу сохранения информации (Move); (iii) сливаемые версии были неравнозначны (были выделены серверная и локальные версии), при разрешении конфликтов в автоматическом режиме предпочтение отдавалось серверной версии; (iv) информация о конфликтах сохранялась и использовалась в расширенном режиме (пользователь может исправить результат автоматического разрешения конфликтов); (v) edit-скрипт преобразовывал серверную версию (а не исходную) в целевую; (vi) была изменена процедура идентификации 3DM-алгоритма: использовалась метрика близости для элементов, имеющих общее происхождение (использовалось Q-расстояние между строками). Предложенный алгоритм был реализован для системы Comapping. Для него был также спроектирован и реализован пользовательский интерфейс работы с конфликтами.

Реализованный алгоритм был применён для сбора информации и начальной формализации требований в трёх индустриальных проектах. Измерялись следующие показатели (в скобках после каждого показателя приводятся цифровые данные для каждого из трёх исследованных проектов): количество узлов и-карты (439, 119, 206), максимальная глубина дерева (11, 5, 7), общий объем текста в и-карте (количество слов — 3903, 328, 811), количество участников процесса групповой работы (4, 3, 2), количество выполнений процедуры слияния (12, 5, 8). Несмотря на то что предложенный алгоритм слияния запускался относительно нечасто, его наличие, по мнению разработчиков анализируемых проектов, обеспечивало надёжность процесса и сохранность собираемых данных.

Метод контроля качества визуальных спецификаций. Известно, что предметно-ориентированный язык, реализованный с помощью стандартного инструмента моделирования (UML-пакет, ARIS, Mega пр.), имеет много ограничений, которые инструмент контролировать не в состоянии. С другой стороны, если предметно-ориен-тированный язык задаётся с помощью метамодели и далее реализуется с помощью DSM-платформ типа MetaEdit и GMF, генерирующих целевой графический редактор по метамодели, то с помощью метамодели затруднительно задать все особенности языка. В качестве примера можно привести спецификацию самого UML, в которой метамодель сопровождается большим количеством дополнительных ограничений. Для их спецификации целесообразно использовать декларативный язык OCL. Если говорить о промышленных решениях, то для обоих обозначенных выше случаев общепринятой индустриальной практикой является создание проверочных скриптов, которые будут автоматически контролировать эти ограничения для реальных моделей. Автором предложен метод разработки таких скриптов и создания итогового валидатора, имеющего единый интерфейс запуска и работающего в режиме средств Build Management. Предложена также методика и архитектура средств для поддержки разработки ограничений корректности на языке OCL с последующей генерацией кода целевого

валидатора. Автором также предложена идея включать в область проверки не только модели, но и способы их представления, в частности, проверять соблюдение правил размещения элементов модели и диаграмм по различным папкам репозитория, а также состав диаграмм на соблюдение некоторых правил, не контролируемых инструментом моделирования. Предложенная архитектура была реализована с применением пакета Dresden OLC Toolkit. С помощью созданной технологии было разработано два целевых валидатора для двух промышленных проектов. Первый проект использовал диаграммы конечных автоматов для спецификации алгоритмов семейства телекоммуникационных систем с автоматической генерацией программного кода по диаграммам (использован инструмент моделирования Real). Второй проект использовал предметно-ориентированное моделирование для описания ИТ-архитектуры крупной нефтегазовой корпорации (использован инструмент моделирования ARIS). В диссертационной работе представлена расширенная метамодель для этого примера, а также соответствующие OCL-ограничения. Чтобы обосновать эффективность предложенного метода, для последней апробации вместе со сгенерированным валидатором была выполнена «ручная» реализация соответствующих проверочных скриптов, которую автор сравнил с автоматически сгенерированным валидатором. Измерялись следующие параметры: сложность разработки, объем кода и время работы. Сложность разработки OCL-ограничений оценивалась по следующей шкале: лёгкая (Л) — использование простых базовых конструкций OCL; средняя (Ср) — добавляются итераторы и процедуры; сложная (С) — создание больших спецификаций (по 20–50 строк) и использование OCL в полном объёме. Из 11 рассмотренных ограничений 6 имели сложность Л, 5 — С, сложных не оказалось. Время работы сгенерированного валидатора на репозитории в 27 мегабайт (рабочий репозиторий КИТ-проекта, содержащий полное описание модели ИТ-архитектуры корпорации) составило 35 минут. С учётом того что эксперименты проводились на виртуальной машине, в то время как целевой валидатор должен работать на высокопроизводительном сервере, данная производительность оказывается приемлемой. С дальнейшим обсуждением данного вопроса можно ознакомиться в тексте диссертационной работы.

Четвёртая глава описывает созданные автором средства анализа, проектирования и разработки специализированных классов ПО, а также отдельных рабочих продуктов процесса разработки ПО, основанных на предметно-ориентированном моделировании — см. соответствующие виды DSM-решений, определённые в Методологии. Рассмотрим эти результаты.

FSS-метод. Метод предназначен для анализа предметной области в контексте разработки ПО, реализующего государственные/публичные сервисы в некоторой предметной области (например, в контексте русско-финского приграничного сотрудничества или жилищно-коммунальное хозяйство Санкт-Петербурга). Необходимость специального метода обусловлена большим количеством информации, которая должна

быть учтена при разработке такого ПО. Схема метода представлена на рис. 3. Отметим, что в рамках FSS-метода целевая Web-система может получаться как результат генерации программного кода по моделям. При генерации кода главным является использование *метафор Web-визуализации*. Такая метафора представляет собой схему фрагмента пользовательского интерфейса, которая поддержана соответствующим универсальным кодом, который, в свою очередь, параметризуется данными из предметной области. Создание итогового кода производится специальным генератором. Однако программная поддержка метафор в рамках FSS-метода не входит в рамки данного исследования.

На рис. 4 приведён пример модели документов. В этом примере сокращённо изображено дерево документов для выдачи загранпаспорта. Начальным (центральным) узлом диаграммы является Input_Documents. Непосредственно с этим узлом связаны документы «Анкета», «Старый загранпаспорт», «Фотография для анкеты», «Квитанция об оплате госпошлины», которые обязательны для всех заявителей. Далее имеется два групповых узла – «До 18 лет» и «От 18 лет», поскольку дальнейшие документы, которые необходимо подать на загранпаспорт, зависят от этого параметра заявителя.

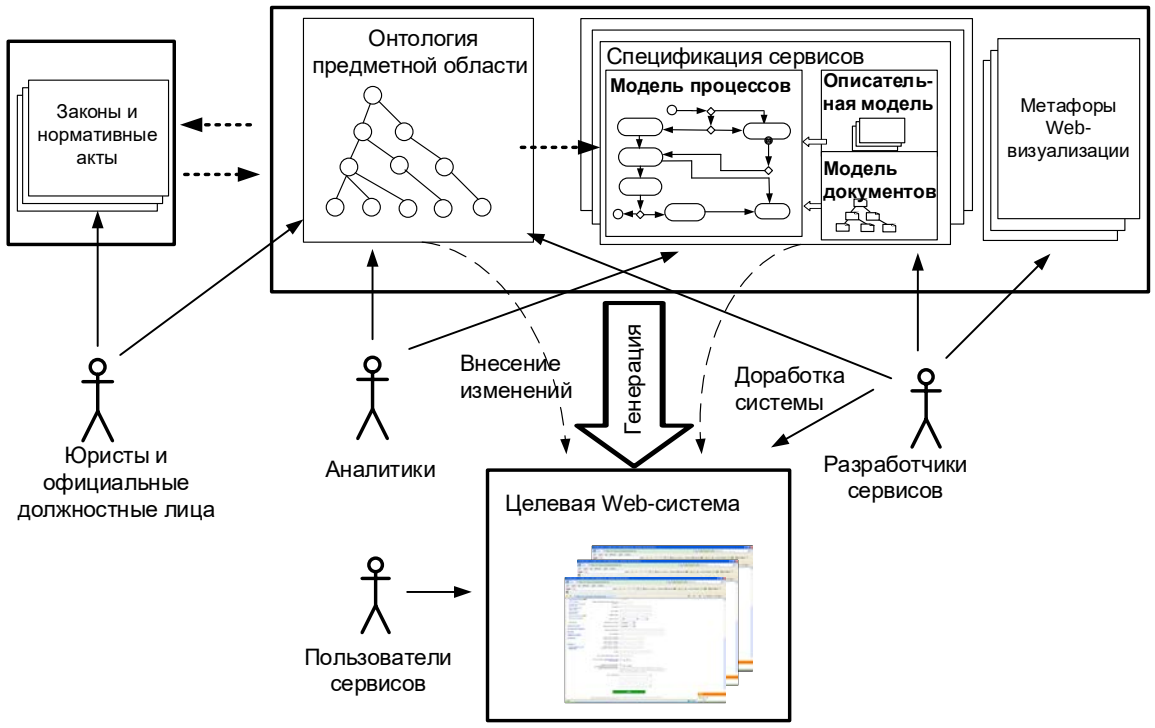

Рис. 3. Схема FSS-метода

С узлом «От 18 лет» связаны документы «Дополнительная анкета», «Дополнительная фотография», «Паспорт гражданина РФ» и др., обязательные для заявителей старше 18-ти лет. У этих заявителей имеется группа документов, зависящая от отношения к воинской обязанности. Здесь возможны следующие ситуации – «Отслуживший», «Военнослужащий», «Призывник». С узлами дерева могут быть также связаны

инфоблоки (InfoBlocks), с помощью которых задаётся дополнительная информация о ситуации или о документе.

Модель средств разработки семейств программно-аппаратных систем, создаваемых на основе компоновки аппаратной части. Опишем целевой класс приложений, на который ориентирована модель: (i) может быть создана общая база знаний оборудования продуктов семейства; (ii) может быть создан программный конструктор семейства, и целевые продукты семейства строятся на основе его компонент; (iii) программное обеспечение целевых систем в значительной степени определяется конфигурацией их аппаратной части.

Для разработки таких систем предлагается следующая модель. Создаётся программный конструктор ПО семейства. Далее выполняется описание всех программных компонент семейства и аппаратуры семейства, а также отображение аппаратуры на программные компоненты (XML-описание). Спецификация оборудования конкретного продукта производится с помощью DSM-языка THCL на основе базы знаний оборудования семейства. По этой спецификации автоматически строится и конфигурируется ПО продукта.

Основными конструкциями языка THCL являются *блок* (аппаратный функциональный узел системы, например, микросхема или плата), *разъем* (для соединителя элементов оборудования кабелями), *кабель*, *переходник* (для перехода с одного разъёма на другой) и *разветвитель* (кабель, перераспределяющий сигналы из одного разъёма по нескольким другим). На рис. 5 представлено главное окно THCL-редактора и фрагмент соответствующей спецификации.

Предложенный метод и языка THCL были апробированы при разработке семейства телевещательных систем. Модель использовалась для создания трёх продуктов семейства; общее количество блоков, разъёмов и кабелей составило от 230 до 300 элементов. Каждый элемент состоял из нескольких фигур Visio, так что суммарное количество фигур в модели одного продукта составило приблизительно 2000. При этом было обнаружено существенное замедление при перерисовке элементов на диаграммах, что, по мнению автора, является ограничением продукта Microsoft Visio при работе с большими диаграммами и сложными нотациями.

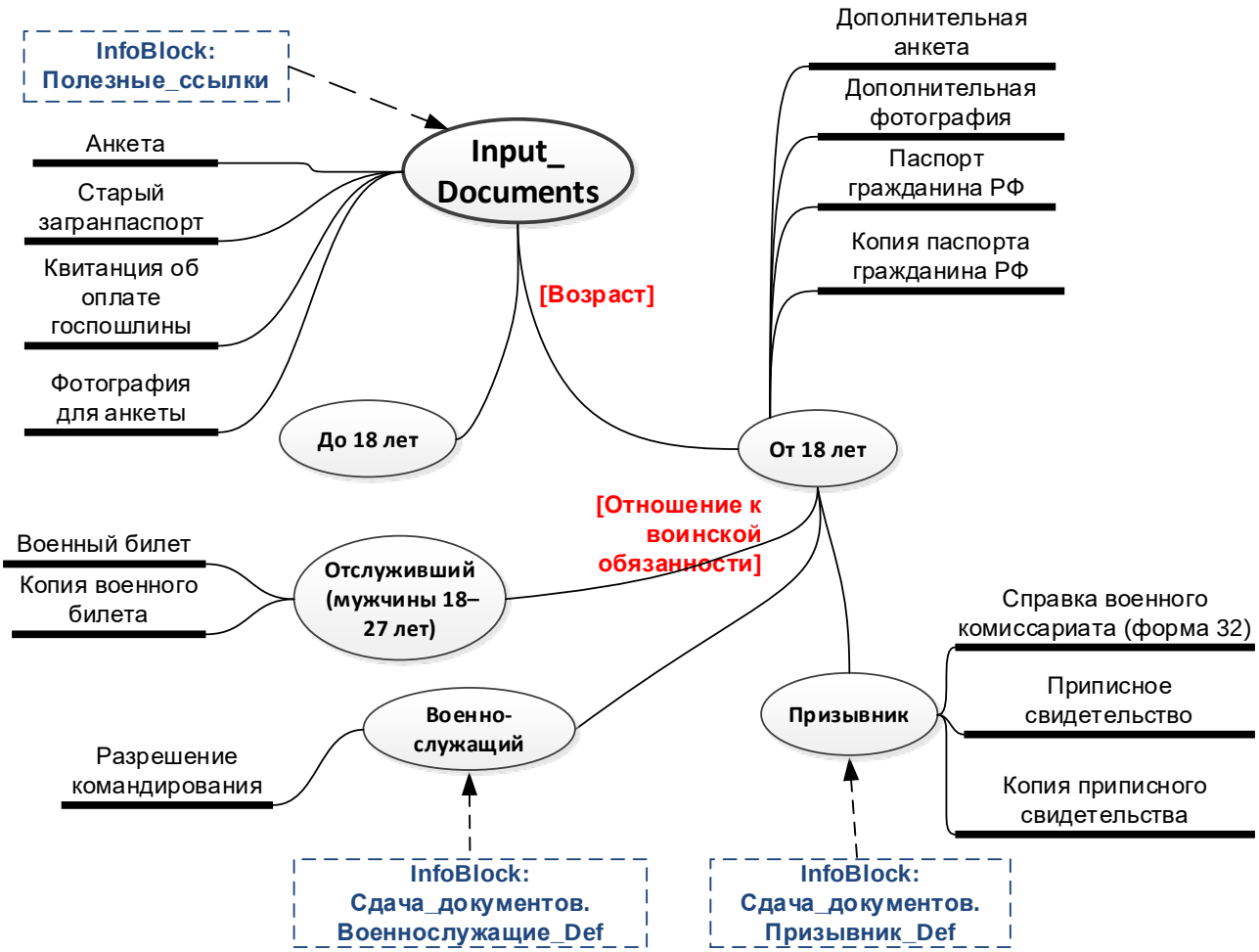

Рис. 4. Пример модели документации FSS-метода

Метод DocLine, язык DRL/GR и алгоритм поиска повторов и рефакторинга документации. DocLine создан под руководством автора диссертационной работы для разработки и сопровождения документации ПО на основе повторного использования. DocLine рассматривает документацию в XML-форматах, поскольку именно такой способ всё чаще применяется для разработки объёмной технической документации. DocLine интегрируется с известным XML-стандартом DocBook (стандарт де-факто в Linux/Unix сообществе), являясь надстройкой последнего, и использует его для задания структуры и полиграфической разметки документов (главы, параграфы, перечисления и пр.).

Язык DRL/GR является частью языка DRL, созданного в рамках проекта DocLine для спецификации повторного использования — «крупноблочного», ориентированного на вариативность включения глав, разделов и других больших частей документации, и «мелкозернистого» — для гибкой конфигурации повторного использования средних и небольших фрагментов документации. DRL имеет две формы — графическую (DRL/GR) и текстовую (DRL/PR). DRL/GR предназначен для проектирования

пакетов документации линейки продуктов и спецификации «крупноблочного» повторного использования, DRL/PR — для задания «мелкозернистого» повторного использования и итоговой спецификации всех документов.

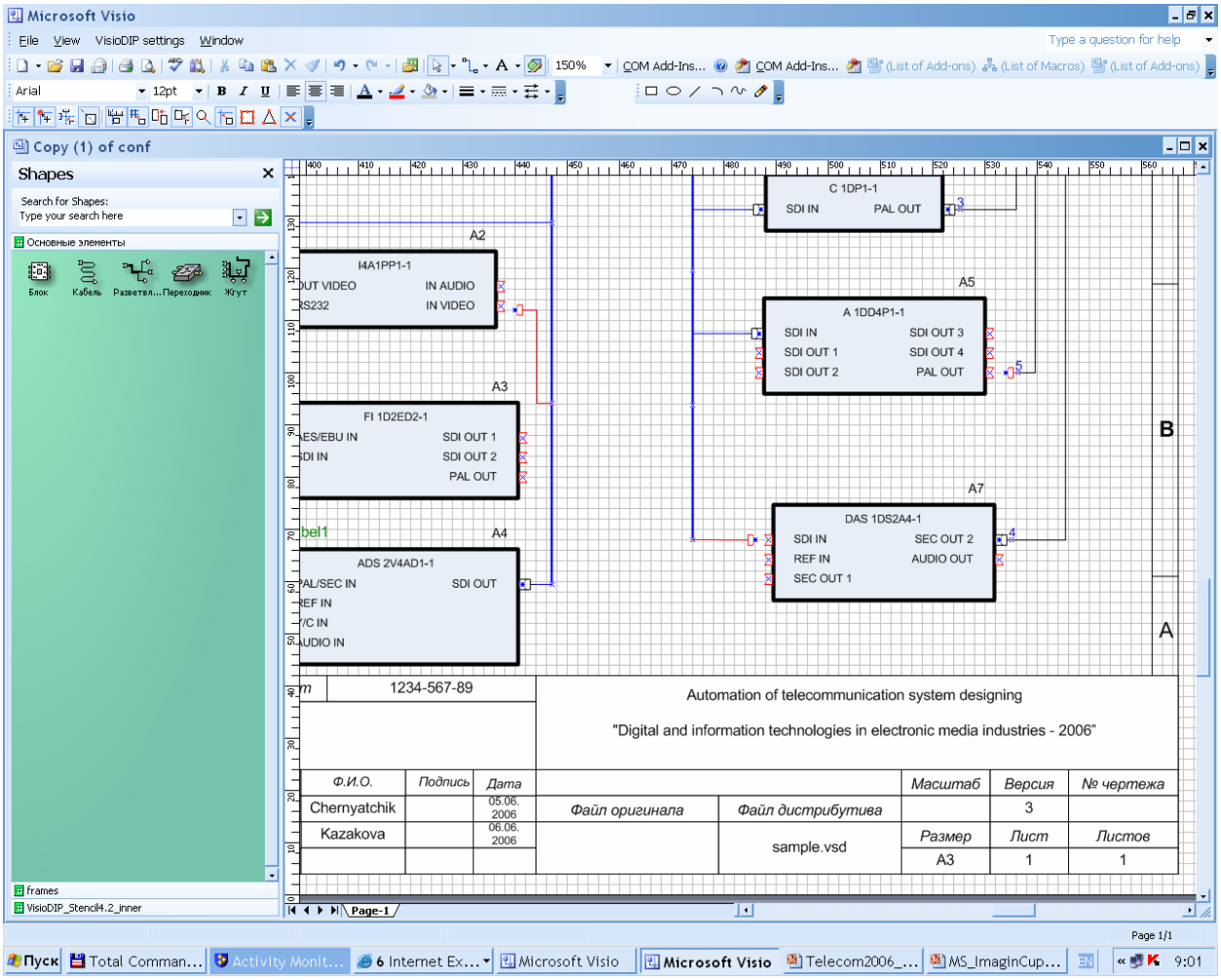

Рис. 5. Главное окно THCL-редактора и фрагмент модели

Вся информация, задаваемая с помощью языка DRL/GR, переходит в спецификацию на DRL/PR, то есть графическая информация имеет текстовый аналог. Документация целевого продукта семейства получается после выполнения препроцессора DocLine, «разрешающего» все виды вариативности и порождающего конечный DocBook-формат, из которого средствами DocBook генерируются итоговые документы. На рис. 6 представлен пример диаграмм вариативности.

Алгоритм поиска повторов и рефакторинга документации предназначен для автоматизации задачи сопровождения документов (см. рис. 7). На вход алгоритма можно подать DRL/DocBook/ASCI/Unicode файлы. Далее в этом документе с помощью инструмента поиска клонов CloneMiner находятся группы клонов, которые затем фильтруются (устраняются клоны, состоящие только из XML-разметки, клоны, состоящие из фраз типа «that is» и пр.).

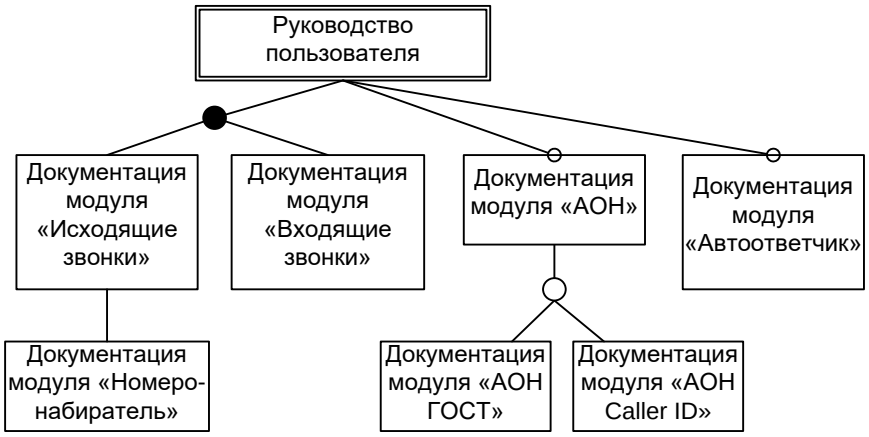

Рис. 6. Пример диаграммы вариативности языка DRL/GR

После этого технический писатель выбирает те группы клонов, которые он хочет превратить в повторно используемые активы в документации, и выполняет для них операции рефакторинга DocLine. В итоге формируется новый целевой документ. В работе приводятся результаты работы алгоритма для документации ядра операционной системы Linux (870 Кб/ 25000 строк).

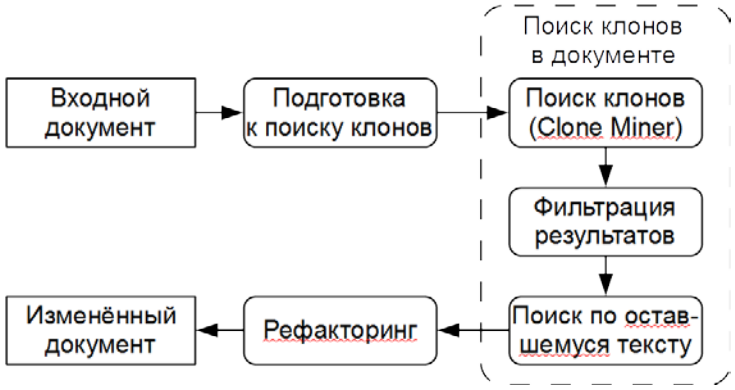

Рис. 7. Схема алгоритма поиска клонов и рефакторинга документации

Модель КИТ-решения. Корпоративная ИТ-архитектура — это концепции, принципы, стандарты и технологии, которыми крупная организация пользуется при разработке и внедрении ИТ-технологий. В данной диссертационной работе предлагается рассматривать проекты по разработке ИТ-архитектуры как специальные ИТ-проекты (КИТ-проекты; КИТ — Корпоративная ИТ-архитектура). Результаты таких проектов предлагается рассматривать как специфические ИТ-решения (КИТ-решения). Основой КИТ-решения является стандартный EAM-инструмент (ARIS, Mega и т.д.). В работе предлагается типовой состав КИТ-решения, анализируются реальные КИТ-проекты, в которых принимал участие автор.

В **пятой главе** представлены практические DSM-решения, при разработке которых была применена предложенная Методология предметно-ориентированного моделирования. Эти решения могут послужить в роли «клише» или прототипов для дальнейшего тиражирования. Фактически, это банк стандартных решений.

1. Модернизация EAM-инструмента ОРГ-Мастер. Данный проект был выполнен под руководством и при непосредственном участии автора диссертационной работы. В задачу проекта входило создание новой реализации программного продукта (предыдущая была выполнена в начале 2000-х годов с помощью технологий, к настоящему времени устаревших). Продукт был перепроектирован и реализован на современных технологиях, его функциональность — реструктурирована. Был модернизирован язык моделирования и создана архитектура дополнительных средств предметно-ориентированного моделирования. На основе данной архитектуры были реализованы программные средства на базе Microsoft Visio (редактор бизнес-процессов, редактор стратегий, редактор «процесс-результат»), разработан и реализован алгоритм слияния моделей, реализующий вычислимый способ хранения базовой версии сливаемых моделей.
2. DSM-решение РУП (РУсско-финское Приграничное сотрудничество) было создано на базе пакета Microsoft Visio и предназначалось для проектирования контента при разработке Web-портала для российско-финских путешественников. Идея решения РУП принадлежит автору диссертационной работы, он также выполнил разработку предметно-ориентированного языка и проектирование программных средств и осуществлял руководство разработкой и использованием средств. Решение РУП было создано в контексте русско-финского исследовательского проекта «Improving Social Services» (проект проходил в 2011–2013 годах в рамках международной программы ENPI Finnish Russian Cross-Border Cooperation). Решение поддерживает три вида моделей: доменная модель, модель типов, модель предметной области. Для этих видов моделей реализована поддержка целостности (Microsoft Visio не предоставляет репозитория). Был также реализован генератор в ОРГ-Мастер — онтология, схема которой создавалась в РУП и заполнялась данными в ОРГ-Мастере. Схема онтологии составила 15 диаграмм формата А4. Использование данного решения позволило реализовать проект качественно и в срок.
3. Средства диаграммной Web-визуализации деятельности правительства города Москвы. В 2011 году правительство города Москвы инициировало Системный Проект, посвящённый описанию и оптимизации управленческих процессов города Москвы. Моделирование управленческих процессов вы-

полнялось в ОРГ-Мастере, на основе этих моделей был создан аналитический портал. В рамках данного проекта под руководством автора диссертационной работы было создано DSM-решение, предназначенное для автоматической генерации диаграммных Web-отчётов по моделям ОРГ-Мастера. Было спроектировано и реализовано девять видов графических нотаций и создано программное решение для автоматизированного построения соответствующих диаграмм по исходной модели. Для спецификации нотаций использовались упрощённые метамодели, предложенные автором и совмещающие в себе информацию об абстрактном и конкретном синтаксисе, включая правила раскладки элементов на диаграмме. Решение поддерживало синхронизацию исходной модели в ОРГ-Мастере и диаграмм с максимальным сохранением выполненной «ручной доводки» раскладки диаграмм. Для этого соответствующие виды диаграмм генерировались сначала в Microsoft Visio, а после этого — в целевой формат SVG. С помощью решения было сгенерировано и размещено на портале несколько сотен диаграмм.

4. КИТ-проект (серия проектов) по моделированию ИТ-архитектуры крупной компании. Крупная нефтегазовая российская компания использует несколько сотен информационных систем, более половины из которых были специально созданы для этой компании. При этом использовалось около трёхсот различных платформ разработки и функционирования информационных систем — Oracle, SAP, 1C, различные Java-технологии и др. В связи с этим компания организовала серию КИТ-проектов. В рамках этих проектов автор принимал активное участие в разработке предметно-ориентированного языка моделирования, а также разрабатывал дополнительное программное обеспечение и участвовал в пилотном моделировании SAP-систем.
5. Система управления знаниями на основе и-карт (Mind Maps) и метод управления разработкой текстов дипломных работ выпускающей кафедрой университета на основе предметно-ориентированного моделирования. Автор принимал активное участие в проектировании системы Comapping, ему принадлежит разработка алгоритма синхронизации двух версий и-карт. Он также занимался разработкой функциональности системы и её применением для различных задач управления знаниями в образовании. В частности, им был разработан метод управления разработкой текстов дипломных работ выпускающей кафедрой университета. Система поддерживает древовидную нотацию, средства раскладки диаграмм и детализации больших диаграмм, а также средства слияния и-карт после обрыва и восстановления Интернет-соединения. Предложен также метод, основанный на и-картах и Comapping, для управления разработкой дипломных текстов со стороны выпускающей

кафедры. Метод применяется в рамках специального семестрового семинара, проводимого для выпускников кафедры системного программирования СПбГУ в последнем семестре, непосредственно перед выпуском студентов. Метод предназначен для помощи и контроля студентов в разработке текстов дипломов. С этой целью студентам предлагается создавать и-карты с планами дипломных текстов, согласовывая их с научными руководителями и ведущим семинара. Эти планы создаются итеративно, далее предлагается создать несколько итераций текстов в соответствии с этими планами (дальнейшую работу над текстами студенты продолжают самостоятельно). Использование предметно-ориентированного решения помогает ведущему семинара эффективно управлять процессом разработки сразу целого множества дипломных текстов, не будучи знакомым с деталями выполняемых работ.

6. Проекты Real и Real-IT — предметно-ориентированный инструментарий для специальных видов ПО. Инструмент моделирования Real предназначен для разработки телекоммуникационных систем, имеющих компоненты информационных систем. Real-IT является предметно-ориентированным решением на базе Real для разработки линейки информационных систем, интенсивно работающих с данными.

Делаются выводы об эффективности предложенной Методологии.

В **заключении** формулируются результаты, достигнутые в ходе выполнения данного исследования.

1. Предложена методология предметно-ориентированного моделирования, предназначенная для разработки инструментов анализа и проектирования программного обеспечения на основе визуальных моделей и предоставляющая средства для спецификации итоговой поставки DSM-проекта, описывающая дополнительные функциональные компоненты, не реализованные существующими техническими средствами, включающая средства для создания процесса разработки и сопровождения DSM-решения, а также для анализа рисков.
2. Разработан алгоритм слияния двух версий и-карт (Mind Maps) в контексте групповой разработки требований к ПО (первичный сбор и анализ требований) средствами Интернет после обрыва и восстановления сетевого соединения.
3. Создана модель v2v-трансформаций для автоматизированной разработки диаграммных сервисов с целью решения проблем навигации и сопровождения больших моделей.
4. Предложен метод контроля качества (корректности) предметно-ориентированных спецификаций, включающий средства для

автоматизированного создания валидаторов спецификаций на основе OCL-ограничений.

5. Предложена модель для проектирования и разработки продуктов семейств программно-аппаратных систем, основанных на программных конструкторах и конфигурировании ПО целевых продуктов с помощью компоновки аппаратной части. Модель включает предметно-ориентированный язык THCL (Telecommunication Hardware Configuration Language) и архитектуру технологии графического проектирования.
6. Создан метод FSS (Formal Services Specification), предназначенный для анализа и проектирования программных систем, реализующих электронный доступ к государственным и публичным услугам.
7. Предложена модель КИТ-решения (Корпоративная ИТ-архитектура), предназначенная для разработки и сопровождения ИТ-архитектур крупных компаний.
8. Создан метод DocLine для разработки и сопровождения документации ПО на основе повторного использования. Предложен предметно-ориентированный язык DRL/GR (Documentation Reuse Language/Graphical Representation) для проектирования документации линеек программных продуктов. Создан алгоритм обнаружения повторов и рефакторинга документов на основе техники поиска клонов в ПО.

В качестве рекомендаций для использования полученных результатов в промышленности, образовании и научных исследованиях указывается, что предложенная Методология предметно-ориентированного моделирования является полноценным руководством для создания DSM-решений различной степени сложности — ИТ-решений, настроенных стандартных продуктов моделирования, а также методик, предназначенных для совсем небольших применений предметно-ориентированных языков.

Сформулированы также перспективы дальнейшей разработки представленной в работе тематики.

СПИСОК ОСНОВНЫХ ПУБЛИКАЦИЙ ПО ТЕМЕ ДИССЕРТАЦИИ

Монография

1. Кознов Д. В. Визуальное моделирование информационных е-сервисов в публичной сфере / Д. В. Кознов. — СПб.: Изд-во СПбГУ. 2014. — 144 с.

Статьи из «Перечня российских рецензируемых научных журналов, в которых должны быть опубликованы основные научные результаты

диссертаций на соискание учёных степеней доктора и кандидата наук», рекомендованного ВАК

2. **Кознов, Д. В. Трансформация динамических представлений в предметно-ориентированном визуальном моделировании / Д. В. Кознов, Е. В. Ларчик, А.Н. Терехов // Программирование. — 2015. — №4. — С. 1–10.**
3. **Кознов, Д. В. Программная инженерия и визуальное моделирование: воспитание культуры работы с информацией / Д. В. Кознов // Програм-мная инженерия. — 2015. — №10. — С. 3–11.**
4. **Кознов, Д. В. Особенности проектов в области разработки корпоративной архитектуры предприятий / Д.В. Кознов, М. Ю. Арзуманян, Ю. В. Орлов, М. А. Деревянко, К. Ю. Романовский, А. А. Сидорина // Бизнес-информатика —2015. — № 4. — С. 15–26.**
5. **Кознов, Д. Метод поиска повторяющихся фрагментов текста в технической документации / Д. Луцив, Д. Кознов, Х. Басит, О. Ли, М. Смирнов, К. Романовский // Научно-технический вестник информационных технологий, механики и оптики. — 2014. — № 4. — С. 106–114.**
6. **Кознов, Д. В. Метод проектирования дипломных работ по программной инженерии / Д. В. Кознов, Д. М. Николаева // Университетский научный журнал. — 2014. — № 8. — С. 131–143.**
7. **Кознов, Д. Архитектура средств графического бизнес-моделирования в технологии ОРГ-Мастер / Д. Кознов, Д. Кудрявцев, Л. Григорьев, Р. Гагарский // Программная инженерия. — 2014. — № 2. — С. 3–11.**
8. **Кознов, Д. В. Предметно-ориентированное визуальное решение для сбора и упорядочивания информации при разработке информационной Web-системы / Д. В. Кознов // Компьютерные инструменты в образовании. — 2013. — № 5. — С. 3–16.**
9. **Кознов, Д.В. Инструменты для управления вариативностью — готовность промышленному применению / Д. В. Кознов, И. А. Новицкий, М. Н. Смирнов // Труды СПИИРАН. — 2013. — № 3. — С. 297–331.**
10. **Кознов, Д. В. Средства типизации и прагматика языка моделирования ОРГ-Мастер / Д. В. Кудрявцев, Д. В. Кознов, Л. Ю. Григорьев // Научно-технический вестник информационных технологий, механики и оптики. — 2013. — № 6. — С. 79–85.**
11. **Кознов, Д. В. О проектировании текстов дипломных работ с помощью ментальных карт / Д. В. Кознов // Компьютерные инструменты в образовании. — 2013. — № 6. — С. 41–52.**
12. **Кознов, Д. В. Модельно-ориентированный метод спецификации государственных услуг / Д. В. Кознов, А. В. Азарсков, А. В. Самочадин, Ю. А. Шевцова, К. Ю. Романовский // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2012. — № 4. — С. 102–116.**
13. **Кознов, Д. В. Поиск клонов при рефакторинге технической документации / Д. В. Кознов, А.В. Шутак, М. Н. Смирнов, М. А. Смажевский // Компьютерные инструменты в образовании. — 2012. — № 4. — С. 30–40.**

14. Кознов, Д. В. О спецификации диаграммных преобразований в графических редакторах / Д. В. Кознов // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2011. — № 3. — С. 100–111.

15. Кознов, Д. В. О задаче слияния карт памяти (Mind Maps) при коллективной разработке / Д. В. Кознов, Е. В. Ларчик, М. М. Плискин, Н. И. Артамонов // Программирование. — 2011. — № 6. — С. 56–66.

16. Кознов, Д. В. WebMLDoc: подход к автоматизированному отслеживанию изменений в пользовательской документации Web-приложений / Д.В. Кознов, М. Н. Смирнов, В. А. Дорохов, К. Ю. Романовский // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2011. — № 3. — С. 112–126.

17. Кознов, Д.В. Моделирование полнофункциональных интерфейсов Web-приложений, интенсивно работающих с данными / А. Н. Иванов, Д. В. Кознов, М. Г. Тыжгеев // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2009. — № 3. — С. 189–204.

18. Кознов, Д. В. DocLine: метод разработки документации семейства программных продуктов / Д. В. Кознов, К. Ю. Романовский // Программирование. — 2008. — № 4. — С. 41–53.

19. Кознов, Д. В. Язык DRL для проектирования и разработки документации семейства программных продуктов / К. Ю. Романовский, Д. В. Кознов // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2007. — № 4. — С. 110–122.

20. Кознов, Д. В. Комплекс средств разработки проблемно-ориентированных визуальных языков / А. А. Павлинов, Д. В. Кознов, А. Ф. Перегудов, Д. Ю. Бугайченко, А.С. Казакова, Р. И. Чернятчик Т. А. Фесенко, А. Н. Иванов // Вестник Санкт-Петербургского университета. Серия 10: Прикладная математика. Информатика. Процессы управления. — 2007. — № 2. — С. 86–96.

21. Кознов, Д. В. Метод автоматической валидации UML-спецификаций на основе OCL / Л. Б. Ольхович, Д. В. Кознов // Программирование. — 2003. — № 6. — С. 44–50.

22. Кознов, Д. В. Real: методология и CASE-средство для разработки систем реального времени и информационных систем / А. Н. Терехов, К. Ю. Романовский, Д. В. Кознов, П. С. Долгов, А. Н. Иванов // Програм-мирование. — 1999. — № 5. — С. 44–51.

Статьи в изданиях, входящих в базы цитирования Web of Science

23. Koznov, D. Computer-supported collaborative learning with mind-maps / D. Koznov, M. Pliskin // Communications in Computer and Information Science. — 2011. — Vol. 17. — P. 478–489.

24. Koznov, D. A knowledge management approach for industrial model-based testing / V. Malinov, E. Sokhransky, M. Novikova // KMIS 2009 — 1st International Conference on Knowledge Management and Information Sharing, Proceedings. — 2009. — P. 200–205.
25. Koznov, D. On project-specific languages and their application in reengineering / D. Boulychev, D. Koznov, A. Terekhov // Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR). — 2002. — P. 177–185.

Статьи в изданиях, входящих в базы цитирования SCOPUS

26. Koznov, D.V. Teaching to write software engineering documents with focus on document design by means of mind maps / D.V. Koznov // Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education (CATE). — 2012. — P. 112–118.
27. Koznov, D.V. Towards e-government services in Russia / D.Koznov, A. Samochadin, A. Azarskov, J. Chevzova // KMIS 2011 — Proceedings of the International Conference on Knowledge Management and Information Sharing. — 2011. — P. 294–301.
28. Koznov, D. Process model of DSM solution development and evolution for small and medium-sized software companies / D. Koznov // Proceedings of IEEE International Enterprise Distributed Object Computing Workshop, EDOC. — 2011. — P. 85–92.
29. Koznov, D. Refactoring the documentation of software product lines / K. Romanovsky, D. Koznov, L. Minchin // Lecture Notes in Computer Science. — 2011. — Vol. 4980. — P. 158–170.