Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

На правах рукописи

Улитин Борис Игоревич

МОДЕЛИ, МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА ТРАНСФОРМАЦИИ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ ЯЗЫКОВ ДЛЯ ИНТЕРФЕЙСОВ ПРОГРАММНЫХ СИСТЕМ ОБЩЕГО НАЗНАЧЕНИЯ

РЕЗЮМЕ

диссертации на соискание ученой степени

кандидата компьютерных наук

Нижний Новгород – 2021

Диссертационная работа выполнена в федеральном государственном автономном образовательном учреждении высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Научный руководитель: Бабкин Эдуард Александрович, кандидат технических наук, PhD, доцент, Нижегородский филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Оглавление

ТЕМА ДИССЕРТАЦИИ

Актуальность темы. В центре внимания этой работы находится понятие предметно-ориентированного языка (ПОЯ) в контексте моделирования структуры интерфейсов программных систем общего назначения. В данном случае под структурой подразумевается совокупность объектов, задействованных в интерфейсе, и взаимосвязей между ними. Алгоритмические ПОЯ, используемые для описания решения некоторой задачи в рамках предметной области, не входят в фокус внимания данной работы. Важно отметить, что используемые в работе подходы на основе ПОЯ применимы для моделирования как программных интерфейсов между различными составляющими сложной программной системы, так и человеко-машинных интерфейсов [14] (т.е. объектов и связей, отображаемых на экране и используемых пользователем). Однако в рамках данного исследования мы демонстрируем приложение и развитие этих подходов только для повышения эффективности разработки человеко-машинных интерфейсов в составе программных систем общего назначения.

С точки зрения компьютерных наук ПОЯ является компьютерным языком (в т.ч. программирования или моделирования) с ограниченными выразительными возможностями, ориентированный на некую конкретную предметную область [17]. С более общей точки зрения (включающей лингвистические аспекты ПОЯ как языка в целом), ПОЯ является искусственно созданным языком, который семантически и синтаксически соответствует некоторой предметной области [2].

Чаще всего ПОЯ разрабатывается как часть некоторой более общей программной системы (которая называется по отношению к ПОЯ основной системой). В этом смысле, ПОЯ связан с языком общего назначения, используемом при создании основной системы и, по отношению к нему, может быть внутренним или внешним [17].

Внутренние ПОЯ написаны на языке основного приложения и встроены в этот язык [17]. С этой точки зрения внутренние ПОЯ представляют собой специфичный способ использования языка общего назначения, используемого при создании основного приложения. Как следствие, для разработки внутренних ПОЯ применимы подходы, характерные для языков общего назначения, и их анализ находится за рамками данного исследования.

В отличие от внутренних ПОЯ, внешние ПОЯ написаны отдельно от основной системы. После создания такие языки встраиваются в основное приложение в качестве компилятора или интерпретатора [17]. Основным преимуществом внешних ПОЯ является то, что они могут наилучшим образом отразить концепты предметной области и требования заказчика [6].

В рамках данной работы мы рассматриваем только внешние ПОЯ, используемые для моделирования интерфейсов программных систем общего назначения. Это эффективно и обоснованно с той точки зрения, что человеко-машинный интерфейс программных систем обладает ограниченной функциональностью (непосредственно связанной с функционалом системы в целом), как и ПОЯ, и может в этом смысле рассматриваться как особый вид ПОЯ [26]. Именно поэтому внедрение элементов разработки и модификации ПОЯ в процессы разработки и модификации программных систем может быть эффективным и обеспечить большую гибкость последних в отношении требований различных категорий пользователей [14].

Интерес к ПОЯ проявляется как среди исследователей, так и среди практиков [26, 31]. В первую очередь это связано с тем, что ПОЯ представляют собой удобный, понятный и достаточно простой с точки зрения пользователя механизм управления той предметной областью, для которой они созданы [17].

Исследованиям проблем разработки ПОЯ посвящены работы многих отечественных ученых, в частности, И.С. Ануреева [1], Б.Н. Гайфуллина и В.Е. Туманова [2], А.О. Сухова [6], В.Г. Федоренкова и П.В. Балакшина [7] и др.

Эта тематика также находит отражение в работах таких зарубежных ученых, как Pablo Gómez-Abajo, Esther Guerra, Juan de Lara [18], Aleksandar Popovic, Ivan Lukovic, Vladimir Dimitrieski, Verislav Djuki [36], Walter Cazzola, Edoardo Vacchi и др.

Кроме того, большое количество работ посвящено исследованиям в смежных областях, в частности: вопросы определения и динамики семантики исследуются в работах А.П. Ершова [4], Л.А. Калиниченко [5, 25], Guizzardi [19, 20], Stoy [39], Strachey [33] и др., определение синтаксических аспектов ПОЯ – в работах Evans [16], Laird [27] и др.

В большинстве современных исследований [17, 26, 34] процесс разработки ПОЯ включает в себя следующие этапы: принятие решения (о необходимости создания ПОЯ, иначе говоря, анализ его применимости), анализ предметной области (для которой создается ПОЯ), проектирование ПОЯ (включающее в себя определение всей структуры ПОЯ и выбор наиболее подходящего вида ПОЯ), реализация ПОЯ и развёртывание ПОЯ.

Также иногда рассматривают как отдельный этап поддержку ПОЯ (включающую в себя возможность эволюции ПОЯ) [13]. Эволюция может происходить как под влиянием изменений самой предметной области [13, 14], так и под влиянием внутренних факторов, таких как изменение поведения пользователей системы [27], их гетерогенность [35].

Однако, как показывает анализ публикаций, в работах не освещен сам механизм отслеживания изменений предметной области и их трансляции на модель ПОЯ. Более того, считается, что к моменту разработки ПОЯ сама модель предметной области уже создана вручную перенесена в модель ПОЯ (как, например, в работах [18, 23]). Справедливо отметить, что ряд исследователей, в их числе Peter Bell [10], Josh G.M. Mengerink, Alexander Serebrenik, Mark van den Brand [29], Ramon R.H.Schiffelers [30] и в частности, Jonathan Sprinkle, Gabor Karsai [38], исследуют вопрос эволюции графических моделей предметной области. Но они не рассматривают последующий перенос проведенных изменений моделей и правил, их обеспечивающих, на

модели ПОЯ [38]. Напротив, эти авторы стараются сохранить структуру ПОЯ неизменной, что не соответствует предположению, что модель ПОЯ идентична модели предметной области, а значит, любое изменение модели предметной области должно выливаться в эквивалентное изменение модели ПОЯ.

Вопрос эволюции ПОЯ является еще более актуальным в случае ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения, поскольку жизненный цикл программных систем предполагает наличие этапа поддержки программной системы, подразумевающего ее развитие (модификацию) под новые требования пользователей [27].

Чтобы подчеркнуть важность этого требования в работе используется понятие динамического контекста, в условиях которого происходит этап поддержки программной системы. Под динамическим контекстом в работе понимается совокупность изменяющихся моделей предметной области и/или набора компетенций пользователей. Наличие динамического контекста обусловливает необходимость адаптации интерфейсов программной системы под новые требования пользователей, а также в соответствии с изменениями моделей предметной области, для которой создана программная система [14].

Как следствие, изменения в системе должны быть отражены в ПОЯ, который также существует в динамическом контексте. Однако, в жизненном цикле ПОЯ этап поддержки (сопровождения) является опциональным, а существующие инструменты разработки ПОЯ не поддерживают функционал по организации полноценной эволюции ПОЯ [26]. Как следствие, могут возникать проблемы эффективности в использовании программной системы на базе ПОЯ [13].

В связи с этим, является актуальной задача предложения подходов, методов и инструментальных средств поддержки эволюции ПОЯ для

моделирования человеко-машинных интерфейсов программных систем общего назначения.

Целью исследования является повышение эффективности процессов поддержки жизненного цикла программных систем общего назначения путем разработки новых моделей и методов эволюции ПОЯ для моделирования интерфейсов.

Для оценки эффективности используются критерии в соответствии с ГОСТ Р ИСО/МЭК 25010-2015 [3], в т.ч. время модификации ПОЯ, количество вносимых вручную строк кода для модификации интерфейсов и др.

Для достижения цели необходимо реализовать следующие ***задачи исследования***:

- проанализировать существующие подходы к разработке и трансформации (эволюции) ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения;
- разработать формальную модель, лежащую в основе предлагаемого подхода к трансформации и эволюции ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения;
- разработать алгоритмы трансформации моделей (как предметной области, так и ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения);
- провести апробацию и тестирование полученных результатов (моделей, алгоритмов и программных прототипов) на ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения в различных предметных областях, обладающих динамическим контекстом.

Диссертация на соискание ученой степени кандидата компьютерных наук – это законченная научно-квалификационная работа, которая соответствует: требованиям пунктов 3.1, 3.3, 3.4, 3.5-3.8, 3.10 Положения о присуждении ученых степеней в Национальном

исследовательском университета «Высшая школа экономики», а также пп. «Инструментальные средства поддержки жизненного цикла программных систем» и «Технологии и инструментальные средства поддержки проектирования, анализа и верификации моделей аппаратных и программно-аппаратных систем» области исследований «Системное программирование» паспорта области науки «Компьютерные науки».

Объектом исследования является процесс поддержки жизненного цикла программных систем общего назначения в части эволюции моделей человеко-машинного интерфейса как примера внешнего ПОЯ в динамических контекстах.

Предметом исследования являются модельно-ориентированные процессы трансформации внешнего ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения в динамическом контексте в случае изменения моделей предметной области или компетенций пользователей ПОЯ.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

Методы исследования. В диссертационной работе использованы такие формальные математические методы как: теория функций, теория графов [37] и графовых грамматик [8], элементы логики предикатов [12]. В качестве средства моделирования предметной области применяются UML-диаграммы и концептуальные (семантические) модели в виде кодифицированных онтологий [19] и объектно-реляционных моделей [8].

В практической части работы применяются методы компиляции, декларативные средства для преобразований на графах [11], методы объектно-ориентированного и событийно-ориентированного программирования.

Научная новизна:

- на основе анализа существующих подходов и методов к разработке ПОЯ представлена обобщенная модельно-ориентированная структура ПОЯ,

представляющая собой унифицированное представление всех уровней структуры ПОЯ;

- формализованы различные виды (модели) эволюции ПОЯ для моделирования человеко-машинного интерфейса для программных систем общего назначения, что позволяет определить согласованную эволюцию СМПО и всех уровней ПОЯ в соответствии с предлагаемым проекционным подходом;
- построен и реализован набор кроссмодельных преобразований (на основе графовых преобразований) для организации горизонтальной и вертикальной эволюции внешнего ПОЯ моделирования интерфейсов программных систем общего назначения;
- на основе выделенного набора кроссмодельных преобразований предложен новый метод (названный проекционным подходом) к разработке внешнего ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения. Предложенный проекционных подход позволяет автоматизировать процесс разработки ПОЯ для моделирования интерфейсов программных систем общего назначения и использовать результаты анализа предметной области (ее представление в виде семантической модели предметной области (СМПО)) при проектировании и реализации ПОЯ, а также организовать автоматизированную эволюцию ПОЯ;
- в соответствии с предложенным методом выполнена алгоритмизация и программная реализация процедуры трансформации человеко-машинного интерфейса как примера внешнего ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения для двух предметных областей.

В результате анализа выполненной программной реализации процедуры трансформации человеко-машинного интерфейса как примера внешнего ПОЯ выявлены повторно-используемые результаты. На этом основании делается вывод, что предложенный подход и модели эволюции ПОЯ применимы для

реализации эволюции ПОЯ в различных предметных областях и являются основой для разработки универсальных программных инструментов поддержки полного жизненного цикла ПОЯ (включая этап эволюции ПОЯ).

Положения, выносимые на защиту:

- формализованная модель эволюции ПОЯ на основе кроссмодельных, графовых преобразований для моделирования интерфейсов программных систем общего назначения в динамических контекстах;
- новый метод (названный проекционным подходом) к разработке внешнего ПОЯ для моделирования интерфейсов программных систем общего назначения, позволяющий автоматизировать процесс разработки ПОЯ для моделирования интерфейсов программных систем общего назначения и использовать результаты анализа предметной области при проектировании, реализации и модификации ПОЯ;
- повторно-используемая программная реализация процедуры трансформации человеко-машинного интерфейса как примера внешнего ПОЯ, использованная в составе двух прототипов программных сред для различных предметных областей.

Личный вклад в положения, выносимые на защиту, включает в себя разработку и реализацию моделей, методов эволюции ПОЯ для моделирования интерфейсов в рамках предлагаемого проекционного подхода с целью повышения эффективности процессов поддержки жизненного цикла программных систем общего назначения. Кроме того, была проведена апробация предлагаемого подхода в процессе реализации программных сред на базе ПОЯ для различных предметных областей.

По итогам апробации были подготовлены тексты научных статей. Соискатель имеет Свидетельство о государственной регистрации программы для ЭВМ №2018616788 «Postgraduate admission office» (v.1.0).

Практическая ценность результатов.

В рамках исследования был разработан новый инженерный метод (проекционный подход) к разработке внешнего ПОЯ для моделирования интерфейсов программных систем общего назначения, позволяющий повысить эффективность и удобство процессов поддержки эволюции ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения. В поддержку предложенного подхода была выполнена алгоритмизация и программная реализация процедуры трансформации человеко-машинного интерфейса как примера внешнего ПОЯ.

Были реализованы прототипы программных сред с применением предлагаемого проекционного подхода на базе ПОЯ (с возможностями эволюции в условиях динамических контекстов) для предметных областей «Программная система Приемной комиссии ВУЗа» и «Программная система распределения ресурсов ж/д станции» (для разработки использовался язык Java и плагины Eclipse, общее число строк кода обоих прототипов – 22483). Разработанные прототипы позволяют не только выполнять сценарии на ПОЯ, но также модифицировать структуру всех уровней ПОЯ в реальном режиме времени без необходимости повторного создания ПОЯ и ИС в целом.

Опыт эксплуатации разработанных программных прототипов показал, что их применение минимизирует количество ошибок и конфликтов при модификации (эволюции) ПОЯ и программной среды в целом, так как все изменения в ПОЯ вносятся в автоматизированном виде, без необходимости ручной модификации. Как следствие, снижается и суммарное время, затрачиваемое на модификацию ПОЯ и программной среды в целом.

Предложенный подход к поддержке эволюции ПОЯ для моделирования интерфейсов программных систем общего назначения применим для реализации программных сред общего назначения различных предметных областей, обладающих динамическим контекстом использования, в частности, в областях с высокой степенью адаптивности, например, в т.н. умных (Smart) системах и кибер-физических системах [26].

ПУБЛИКАЦИИ И АПРОБАЦИЯ РАБОТЫ

Публикации повышенного уровня

1. Ulitin, B. Ontology-based reconfigurable DSL for planning technical services / B. Ulitin, E. Babkin // IFAC-PapersOnLine. – 2019. – v. 52, no. 13. – pp. 1138-1144. – главный соавтор.

Публикации стандартного уровня

2. Ulitin, B. Adapting Domain-Specific Interfaces Using Invariants Mechanisms / B. Ulitin, T. Babkina // Advanced Information Systems Engineering Workshops. CAiSE 2021. Lecture Notes in Business Information Processing, vol 423. – 2021. – pp. 81-92. – главный соавтор.
3. Ulitin, B. Knowledge life cycle management as a key aspect of digitalization / E. Babkin, T. Poletaeva, B. Ulitin // Communications in Computer and Information Science. 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, IC3K 2019, Vienna, Austria, September 17-19, 2019, Revised Selected Papers Issue 1297: Knowledge Discovery, Knowledge Engineering and Knowledge Management. – 2020. – pp. 429-452.
4. Ulitin, B. Automated formal verification of model transformations using the invariants mechanism / B. Ulitin, E. Babkin, T. Babkina, A. Vizgunov // Perspectives in Business Informatics Research. 18th International Conference, BIR 2019, Katowice, Poland, September 23-25, 2019 Proceedings. Lecture Notes in Business Information Processing. – 2019. – i. 365. – pp. 59-73. – главный соавтор.
5. Ulitin, B. Digitalization: A meeting point of knowledge management and enterprise engineering / E. Babkin, T. Poletaeva, B. Ulitin // IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management. – 2019. – v. 3. – pp. 22-36.
6. Ulitin, B. A Projection-Based Approach for Development of Domain-Specific Languages / B. Ulitin, E. Babkin, T. Babkina // Perspectives in Business

Informatics Research. 17th International Conference, BIR 2018, Stockholm, Sweden, September 24-26, 2018 Proceedings. Lecture Notes in Business Information Processing. – 2018. – i. 330. – pp. 219-234. – главный соавтор.

7. Улитин, Б.И. Моделирование динамики онлайн-дискуссий в сети Интернет с использованием многоагентных систем / Э.А. Бабкин, Т.С. Бабкина, Б.И. Улитин // Бизнес-информатика. – 2018. – № 2 (44) – с. 17-29.
8. Ulitin, B. Ontology and DSL co-evolution using graph transformations methods / B. Ulitin, E. Babkin // Perspectives in Business Informatics Research. 16th International Conference, BIR 2017, Copenhagen, Denmark, August 28-30, 2017 Proceedings. Lecture Notes in Business Information Processing. – 2017. – 2017. – i. 295. – pp. 233-247. – главный соавтор.
9. Ulitin, B. Combination of DSL and DCSP for decision support in dynamic contexts / B. Ulitin, E. Babkin, T. Babkina // Perspectives in Business Informatics Research. 15th International Conference, BIR 2016, Prague, Czech Republic, September 15-16, 2016 Proceedings. Lecture Notes in Business Information Processing. – 2016. – i. 261. – pp. 159-173. – главный соавтор.

Прочие публикации

10. Ulitin, B. Providing Models of DSL Evolution Using Model-to-Model Transformations and Invariants Mechanisms / B. Ulitin, E. Babkin // Digital Transformation and New Challenges, Lecture Notes in Information Systems and Organisation. – 2020. – v. 40. – pp. 37-48. – главный соавтор.
11. Ulitin, B. An Object-Oriented Model for Smart Devices in Internet of Things / B. Ulitin, E. Babkin // Proceedings of the 22st Conference of Open Innovations Association FRUCT, Jyvaskyla, Finland. – 2018. – pp. 263-271.

    – главный соавтор.
12. Ulitin, B. Ontology-based DSL development using graph transformations methods / B. Ulitin, E. Babkin, T. Babkina // Journal of Systems Integration. – 2018. – v.9 (2). – pp. 37-51. – главный соавтор.

Свидетельства о регистрации программы для ЭВМ и баз данных:

13. Свидетельство о государственной регистрации программы для ЭВМ №2018616788 «Postgraduate admission office» (v.1.0).

Доклады на конференциях и семинарах

Результаты диссертационного исследования были представлены на следующих международных конференциях и семинарах:

1. 15th International Conference on Perspectives in Business Informatics Research, Prague, Czech Republic, September 15-16, 2016. Доклад: «Combination of DSL and DCSP for decision support in dynamic contexts» («Комбинация предметно-ориентированных языков и распределенных методов удовлетворения ограничений для поддержки принятия решений в динамических контекстах»);
2. 16th International Conference on Perspectives in Business Informatics Research, Copenhagen, Denmark, August 28-30, 2017. Доклад: «Ontology and DSL co-evolution using graph transformations methods» («Организация коэволюции онтологии и предметно-ориентированных языков с использованием механизмов графовых трансформаций»);
3. 17th International Conference on Perspectives in Business Informatics Research, Stockholm, Sweden, September 24-26, 2018. Доклад: «A Projection-Based Approach for Development of Domain-Specific Languages» («Проекционный подход к разработке предметно-ориентированных языков»);
4. 22st Conference of Open Innovations Association FRUCT, Jyvaskyla, Finland, May 15-18, 2018. Доклад: «An Object-Oriented Model for Smart Devices in Internet of Things» («Объектно-ориентированная модель интеллектуальных устройств в рамках концепции Интернета вещей»);
5. 18th International Conference on Perspectives in Business Informatics Research, Katowice, Poland, September 23-25, 2019. Доклад: «Automated formal verification of model transformations using the invariants mechanism»

(«Автоматическая формальная верификация преобразований моделей с использованием механизма инвариантов»);

6. 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, Vienna, Austria, September 17-19, 2019. Доклад: «Digitalization: A meeting point of knowledge management and enterprise engineering» («Цифровизация: место встречи управления знаниями и корпоративной инженерии»);
7. 15th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS в рамках CAiSE 2019), Rome, Italy, June 3-4, 2019. Доклад: «Providing Models of DSL Evolution Using Model-to-Model Transformations and Invariants Mechanisms» («Реализация моделей эволюции предметно-ориентированных языков с использованием кроссмодельных преобразований и механизмов инвариантов»);
8. 9th IFAC/IFIP/IFORS/IISE/INFORMS Conference Manufacturing Modelling, Management and Control (MIM), Berlin, Germany, August 28-30, 2019. Доклад: «Ontology-based reconfigurable DSL for planning technical services» («Реконфигурируемый предметно-ориентированных язык на основе онтологий для планирования технического обслуживания»);
9. 1st International Workshop on Model-driven Organizational and Business Agility (MOBA-2021) в рамках 33rd International Conference on Advanced Information Systems Engineering, June 28-29, 2021. Доклад: «Adapting Domain-Specific Interfaces Using Invariants Mechanisms» («Адаптация предметно-ориентированных интерфейсов с использованием механизмов инвариантов»).

СОДЕРЖАНИЕ РАБОТЫ

Диссертационная работа состоит из введения, 4 глав, заключения, списка литературы, включающего 116 источников (в т.ч. 17 отечественных),

глоссария и 7 приложений. Работа изложена на 162 листах машинописного текста, содержит 45 рисунков, 4 таблицы.

Во введении представлены актуальность работы, цель и задачи диссертационного исследования, научная новизна, теоретическая и практическая значимость исследования, а также приводится краткое содержание работы.

Глава 1 посвящена анализу существующей классической методологии проектирования и реализации ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения в контексте жизненного цикла программных систем общего назначения. Для этого в главе приводится определение ПОЯ и его места в программных системах общего назначения. Представлено описание основных этапов классического жизненного цикла ПОЯ и программных систем. Выделены недостатки существующих методов управления жизненным циклом ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения.

В частности, жизненный цикл ПОЯ и программных систем не полностью соответствуют друг другу, поскольку в случае программных систем этап поддержки и сопровождения с возможностью модификации является обязательным, в то время как в случае ПОЯ данный этап жизненного цикла является опциональным и не поддерживается большинством существующих платформ разработки ПОЯ.

Кроме того, в процессе разработки ПОЯ по классическому подходу имеет место дублирование процессов: так, на этапе анализа предметной области формируется семантическая модель предметной области, которая в дальнейшем в формальном виде не используется при определении семантики ПОЯ, осуществляемом вручную. Как следствие, при внесении изменений в семантическую модель предметной области необходимо также вручную корректировать и семантическую модель ПОЯ, а в последующем и синтаксис ПОЯ.

Данные недостатки классического подхода делают невозможной автоматизацию основных этапов жизненного цикла ПОЯ, а также не позволяют организовать его эволюцию, поскольку любое изменение на всех уровнях структуры ПОЯ проводится вручную и приводит к созданию нового ПОЯ, не согласованного с предыдущим ПОЯ. Данный недостаток становится критическим в случае ПОЯ для программных систем общего назначения на этапе их сопровождения и модификации в соответствии с требованиями пользователей.

Для устранения недостатков классического подхода предлагается использовать подход, основанный на модельно-ориентированном представлении всех уровней структуры ПОЯ с организацией процесса разработки ПОЯ посредством кроссмодельных преобразований между различными моделями: семантической моделью предметной области, семантической моделью ПОЯ, метамоделью ПОЯ.

Глава 2 посвящена анализу существующих методов и формализмов, используемых при описании структуры ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения. В частности, рассматриваются формализации таких артефактов, как: семантическая модель предметной области (СМПО), составляющие структуры ПОЯ: семантика, абстрактный (метамодель) и конкретный синтаксис.

В частности, выделено, что структура ПОЯ включает в себя семантику, абстрактный и конкретный синтаксис (Рис. 1). При этом семантика ПОЯ является проекцией СМПО и может быть получена путем кроссмодельных преобразований (Рис. 2) из формализованного представления СМПО в виде

$$DSM = (\mathcal{H}_C, \mathcal{H}_R, O, R, A, M, D) \quad (1)$$

где

- $\mathcal{H}_C$ и $\mathcal{H}_R$ представляют собой множества классов и отношений между ними. Классы содержат набор атрибутов, тип каждого из которых также

является классом. В обоих множествах $\mathcal{H}_C$ и $\mathcal{H}_R$ определены частичные порядки, позволяющие представлять концепты и таксономии отношений соответственно;

- O и R представляют собой множества объектов классов и отношений между ними, также называемых объектами и кортежами соответственно;
- A представляет собой множество аксиом, описанных специальными правилами, выражающими ограничения предметной области;
- M представляет собой набор модулей вывода, которые представляют собой логические программы, состоящие из набора (дизъюнктивных) правил, которые позволяют рассуждать о представленных и сохраненных знаниях, а также выводить новые, не объявленные явно ранее, знания;
- D представляет собой набор дескрипторов (т. е. правил вывода в двумерной объектно-ориентированной грамматике атрибутов), позволяющий распознавать экземпляры класса (концепции), содержащиеся в O, а также их аннотации.

Учитывая, что любая модель представляет собой комбинацию (E, R) некоторого множества сущностей и отношений между ними, метамодель ПОЯ может быть формализована в модельно-ориентированном виде в следующей форме.

- ***Множество сущностей метамодели*** $E = \{e_i\},\ i \in \mathbb{N},\ i < \infty$, где каждая сущность $e_i = \{SName_i, SICount_i, Attr_i, Opp_i, SRest_i\}$ характеризуется своим названием ($SName_i$, уникальное в рамках конкретной модели), допустимым количеством экземпляров сущности ($SICount_i \in \mathbb{N},\ SICount_i \geq 0$), множеством атрибутов ($Attr_i = \{attr_{j_i}\},\ j_i \in \mathbb{N},\ j_i < \infty$), множеством операций над экземплярами

сущности ($Opp_i = \{opp_{j_i}\},\ j_i \in \mathbb{N},\ j_i < \infty$) и множеством ограничений ($SRest_i = \{srest_{j_i}\},\ j_i \in \mathbb{N},\ j_i < \infty$).

- ***Множество отношений между сущностями*** $R = \{r_i\},\ i \in \mathbb{N},\ i < \infty$, где каждое отношение $r_i = \{RName_i, RType_i, RMult_i, RRest_i\}$ определяется своим названием ($RName_i$, уникальное в рамках конкретной модели), типом ($RType_i \in \mathbb{N},\ RType_i \geq 0$), определяемым природой отношения, множественностью (или кардинальностью в отношении) ($RMult_i \in \mathbb{N},\ RMult_i \geq 0$), которая определяет, сколько экземпляров сущностей, участвующих в отношении, могут быть использованы, и множеством ограничений ($RRest_i = \{rrest_{j_i}\},\ j_i \in \mathbb{N},\ j_i < \infty$).

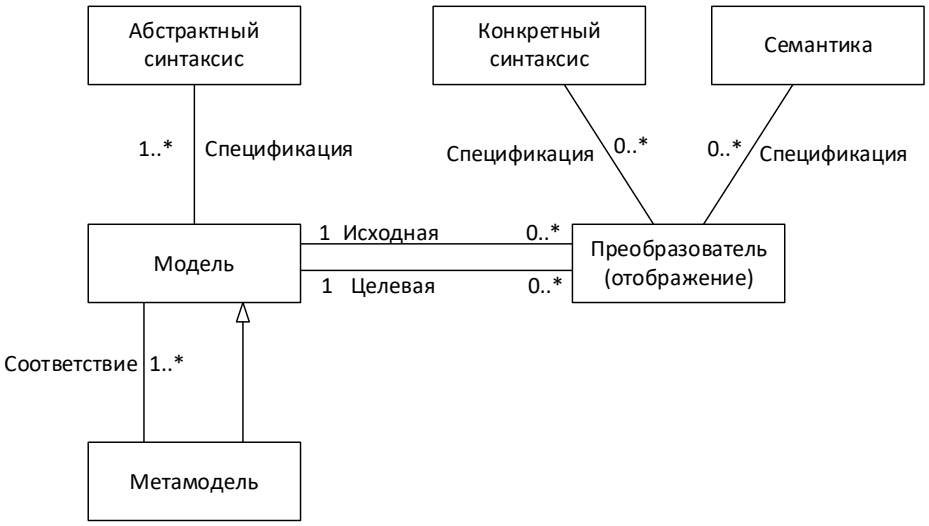

Рис. 1. Определение структуры ПОЯ в модельно-ориентированном виде [13]

Основываясь на данных положениях, структура метамодели ПОЯ может быть определена как $(E, R, Rest, Opp)$, где первые два элемента E и Rel представляют собой объектный уровень, а остальные $Rest = \bigcup_{i=1}^{|E|} SRest_i \bigcup_{i=1}^{|E|} RRest_i$, Opp представляют собой функциональные аспекты работы с ПОЯ. Интерпретируя множество E как множество сущностей предметной области, R как множество отношений между ними и $Opp, Rest$ как множество операций над сущностями и ограничений на них (соответственно),

можно утверждать, что структура СМПО (представленная в Разделе 2.1) и структура метамодели ПОЯ соответствуют друг другу.

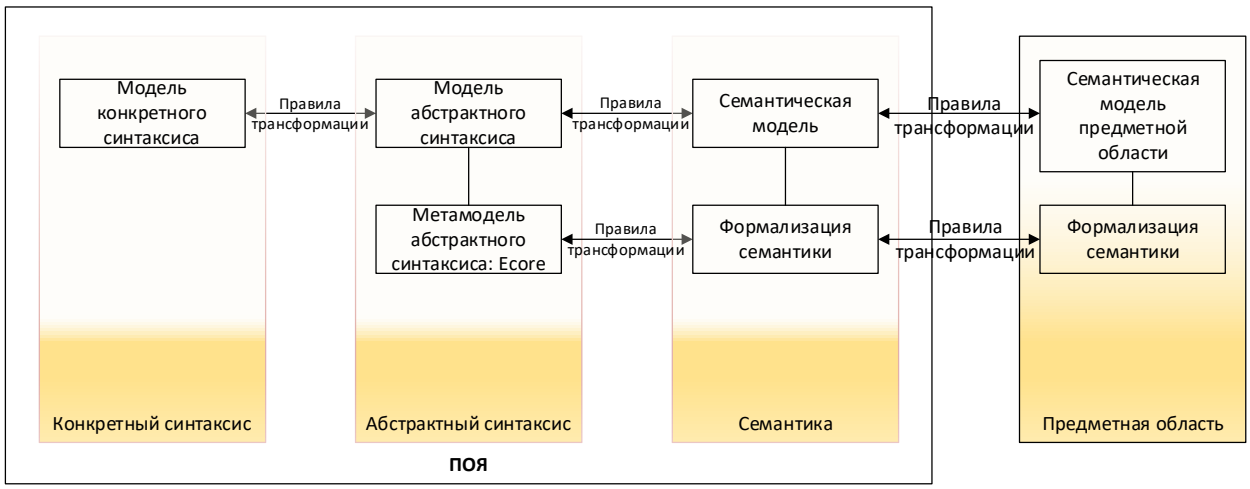

Рис. 2. Определение структуры ПОЯ посредством механизмов кроссмодельных преобразований

Синтаксический же уровень ПОЯ может быть формализован в виде тройки $(O_{syntax}, R_{syntax}, Rule_{syntax})$, где $O_{syntax} \subseteq E$ и $R_{syntax} \subseteq R$ являются подмножествами множеств объектов и отношений между ними метамодели ПОЯ, а $Rule_{syntax}$ представляет собой множество правил, описывающих отображения между метамоделью и конкретным синтаксисом ПОЯ.

Такое определение конкретного синтаксиса ПОЯ на основе его метамодели не зависит от типа конкретного синтаксиса ПОЯ (например, текстовый или визуальный).

В этих условиях мы можем сказать о полном модельно-ориентированном представлении структуры синтаксиса ПОЯ. Структура позволяет не только описывать оба уровня синтаксиса ПОЯ структурированным и унифицированным образом, но и оптимизировать процесс разработки и дальнейшего развития ПОЯ путем введения нескольких синтаксических диалектов ПОЯ на одной неизменяемой метамодели. Более того, управление эволюцией ПОЯ может быть обеспечено аналогичным образом как на мета-уровне, так и на уровне конкретного синтаксиса, без

необходимости воссоздавать всю структуру ПОЯ каждый раз, когда требуются изменения. Это важно, поскольку ПОЯ может иметь несколько конкретных синтаксисов, объединенных единой метамоделью.

Для переходов между различными моделями (СМПО, модели различных уровней структуры ПОЯ) используются кроссмодельные преобразования. При этом в процессе преобразований сохраняются существенные характеристики (свойства) моделей и их объектов (например, свойство принадлежности объекта к определенному классу).

В отличие от существующих вариантов описания структуры ПОЯ (в частности, текстовых [6, 21] и визуальных [18] ПОЯ), отражающих либо абстрактный синтаксис ПОЯ, либо конкретный синтаксис ПОЯ, без отражения семантических аспектов, представленная обобщенная структура позволяет описывать любые ПОЯ, независимо от их вида, и позволяет устранить недостатки классического подхода к разработке ПОЯ, рассмотренного в Главе 1, в частности, обеспечив возможность переноса результатов этапа анализа предметной области на этап непосредственной разработки ПОЯ путем применения кроссмодельных преобразований к СМПО. В дальнейшем аналогичные преобразования можно использовать для построения всех уровней синтаксиса ПОЯ. Это позволяет организовать согласованные изменения в СМПО и структуре ПОЯ и напрямую использовать в процессе разработки ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения представление предметной области в виде СМПО, автоматизируя процессы определения структуры ПОЯ, тем самым устраняя необходимость повторного создания ПОЯ в случае модификации СМПО в процессе эволюции предметной области.

Глава 3 содержит подробное описание предлагаемого нового проекционного подхода к разработке ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения,

базирующегося на модельно-ориентированном представлении структуры ПОЯ, сформулированном в предыдущей главе.

В таком случае семантика ПОЯ полностью основывается на семантической модели предметной области и может быть описана с помощью соответствующей СМПО. С этой точки зрения семантика ПОЯ может быть описана как тройка $Sem = (O, R, A)$, где O – множество объектов предметной области, R – отношения между ними, A – множество ограничений предметной области (как на объекты, так и на связи между ними).

Метамодель (абстрактный синтаксис) ПОЯ может быть представлен в виде множества $MM = (E, Rel, Rest, Opp)$, где первые два элемента E и Rel представляют собой объектный уровень, а остальные $Rest = \bigcup_{i=1}^{|E|} SRest_i \bigcup_{i=1}^{|E|} RRest_i$, Opp представляют собой функциональные аспекты работы с ПОЯ. В отличие от других существующих способов представления метамодели ПОЯ (например, в соответствии с идеями J. Luoma из [28]), в данном случае на уровне метамодели мы получаем полное объектно-ориентированное представление, с возможностью преобразования его на уровень конкретного синтаксиса.

Конкретный синтаксис может быть формализован в виде тройки $CS = (O_{syntax}, R_{syntax})$, где $O_{syntax} \subseteq E$ и $R_{syntax} \subseteq R$ являются подмножествами множеств объектов и отношений между ними метамодели ПОЯ.

Как следствие, между СМПО и моделями, описывающими семантический уровень ПОЯ, уровень абстрактного синтаксиса и уровень конкретного синтаксиса можно организовать переходы, используя кроссмодельные преобразования, описанные в Разделе 2.4. Обозначив $Rule_{sem}$, $Rule_{MM}$, $Rule_{syntax}$ соответственно правила перехода от СМПО к модели семантики ПОЯ, от модели семантики ПОЯ к модели абстрактного синтаксиса (метамодели) ПОЯ и от метамодели ПОЯ к модели конкретного синтаксиса, можем утверждать, что структура ПОЯ в модельно-

ориентированном виде представляет собой множество: $(Sem, Rule_{sem}, MM, Rule_{MM}, CS, Rule_{syntax})$.

В данном случае результат применения правил трансформации к СМПО для получения семантической модели ПОЯ называется проекцией. Метамодель ПОЯ, получаемая из каждой отдельной проекции посредством соответствующих правил трансформации составляет синтаксис языка. Наконец, модели конкретного синтаксиса, получаемые из метамодели путем применения соответствующих правил трансформации, являются различными диалектами ПОЯ, поскольку основаны на единой метамодели и, как следствие, содержат разные представления для одних и тех же объектов и операций над ними.

Исходя из этого, предлагается следующая новая иерархия процесса построения ПОЯ в соответствии с проекционным подходом (Рис. 3).

В нашем случае эта иерархия разделена на четыре уровня в соответствии с этапами разработки ПОЯ. Каждый нижний уровень основан на модельных артефактах верхнего уровня [40].

Единая мета-метамодель (англ. kernel meta-meta-model, KM3 [22]) определяет общие основания для всех метамоделей и моделей нижних уровней.

Структура семантической иерархии определяет соответствующий процесс создания внешних ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения. Он начинается с определения СМПО, содержащей все ключевые объекты целевой предметной области и отношения между ними. Когда СМПО создана, мы можем построить семантическую модель ПОЯ с помощью операции семантической проекции. Любая семантическая проекция выполняет определенное M2M-преобразование СМПО в какой-то ее фрагмент. Таким образом, семантическая проекция полностью определяет семантическую модель того или иного

диалекта ПОЯ. В этом случае семантическая модель становится объектно-временной структурой, поскольку ее следует адаптировать в соответствии с изменениями в СМПО с течением времени, тем самым определяя новое заполнение объекта ПОЯ.

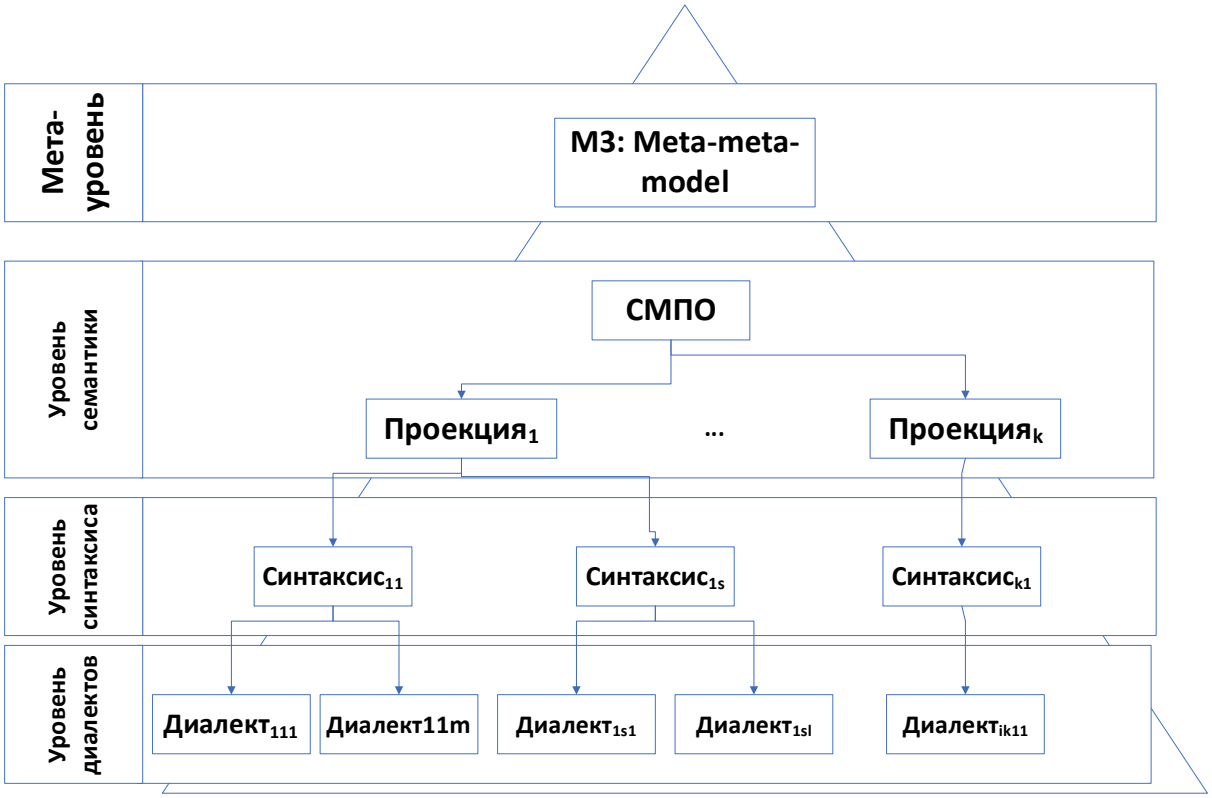

Рис. 3. Семантическая иерархия процесса построения ПОЯ в проекционном подходе

После того, как результат семантической проекции успешно получен (тем самым, определена семантическая модель ПОЯ), синтаксический уровень ПОЯ может быть получен путем кроссмодельного (M2M) преобразования результата соответствующей семантической проекции. Получаемые в данном случае синтаксические модели ПОЯ независимы друг от друга и определяются конечными пользователями в соответствии с адаптацией семантической проекции к их собственным задачам.

Наконец, созданные модели синтаксиса используются конечными пользователями ПОЯ, которые определяют набор диалектов ПОЯ в рамках одной конкретной синтаксической модели.

Таким образом, в соответствии с предлагаемым подходом, ПОЯ создается с учетом знаний предметной области и требований пользователей. Первый факт дает возможность организовать согласованные изменения в СМПО и семантической модели ПОЯ, а второй гарантирует возможность вносить изменения в синтаксис ПОЯ без необходимости переопределять семантический уровень ПОЯ.

Предлагаемый подход существенно отличается от классического подхода к построению ПОЯ (Рис. 4), в котором переход между различными уровнями структуры ПОЯ реализуется вручную, а результаты анализа предметной области, зафиксированные в виде СМПО, используются только неформально. В данном случае изменения на различных уровнях структуры ПОЯ не могут быть реализованы независимо друг от друга, поскольку изменения в синтаксис ПОЯ требуют первоначальной реализации соответствующих изменений на уровне семантики ПОЯ и его метамодели. Как следствие, каждое изменение в целевой предметной области приводит к необходимости ручного переопределения всех уровней структуры ПОЯ. Аналогичный процесс повторяется в случае, когда изменения в ПОЯ вызваны конечными пользователями. В результате, на выходе мы получаем множество несовместимых диалектов ПОЯ, которые не могут быть сопоставлены между собой из-за различий на всех уровнях структуры ПОЯ.

По сравнению с традиционными подходами, предлагаемый проекционный подход (Рис. 4, внизу) к разработке ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения организован в строгом соответствии с целевой предметной областью и жизненным циклом программных систем.

В данном случае результаты этапов жизненного цикла не только фиксируются в виде артефактов, но также используются при реализации последующих этапов жизненного цикла. Так, например, результаты этапа анализа предметной области фиксируются в виде семантической модели

предметной области (представленной, например, в виде онтологии), затем данная модель с помощью кроссмодельных преобразований преобразуется в семантическую модель ПОЯ, которая в данном случае может использовать не полную семантическую модель предметной области, но ее отдельные компоненты. Тем самым достигается возможность на одной семантической модели предметной области построить несколько семантических моделей для различных ПОЯ. Затем с помощью кроссмодельных преобразований на базе семантической модели ПОЯ строится метамодель (абстрактный синтаксис) ПОЯ, на базе которого в дальнейшем определяется конкретный синтаксис языка.

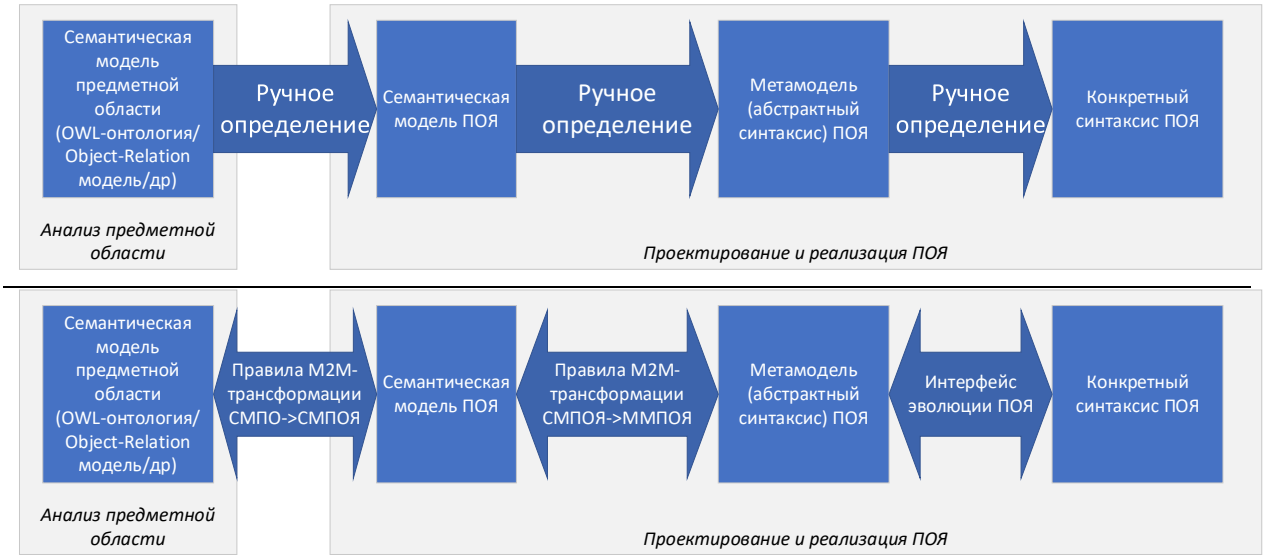

Рис. 4. Схема различий традиционного (наверху) и проекционного (внизу) подходов к построению ПОЯ

При этом, поскольку кроссмодельные преобразования являются двунаправленными, изменения в одной из моделей могут быть автоматизировано перенесены на другие модели, тем самым сохраняя согласованность всех моделей между собой. Единственным этапом, на котором требуется участие пользователя, остается этап определения конкретного синтаксиса языка. Однако, даже данный этап частично автоматизирован, поскольку пользователь использует объекты метамодели

ПОЯ, определяя лишь набор команд по работе с ними через человеко-машинный интерфейс эволюции ПОЯ, встроенный в систему.

В результате мы можем определить несколько синтаксических диалектов ПОЯ в рамках одной конкретной семантической модели ПОЯ, которые будут согласованными и между которыми пользователи могут осуществлять взаимные переходы без переопределения семантических моделей ПОЯ.

В предлагаемом проекционном подходе переход между различными составляющими структуры ПОЯ происходит посредством применения набора правил кроссмодельных преобразований, базирующегося на сформулированных в главе формализациях горизонтальной и вертикальной эволюции ПОЯ.

Вертикальная эволюция означает изменение уровня концептуализации (перспективы) модели. В процессе данного вида эволюции в модель добавляются новые или удаляются существующие сущности (с соответствующим последующим изменением набора отношений – поскольку в рамках СМПО сущность не может являться «висячей» (не связанной с другими сущностями)). С формальной точки зрения вертикальную эволюцию можно определить следующим образом: (E^1, R^1) является результатом вертикальной эволюции модели (E, R), если $|E| \neq |E^1|$ и $|R| \neq |R^1|$. Обозначив через F_v процедуру вертикальной эволюции, можно переписать данное выше определение следующим образом: $F_v\big((E, R)\big) = (E^1, R^1) \Longrightarrow \{|E| \neq |E^1|\text{^}|R| \neq |R^1|\}$.

В отличие от вертикальной, в случае ***горизонтальной эволюции*** модели сохраняется уровень ее концептуализации, но изменяется наборов атрибутов для некоторых сущностей и/или набор отношений между сущностями. С формальной точки зрения горизонтальную эволюцию можно определить следующим образом: (E^1, R^1) является результатом горизонтальной эволюции модели (E, R), если $|\mathrm{E}| = |\mathrm{E}^1|$ и $\exists\, \mathrm{e_i} \in \mathrm{E}, \mathrm{e_i^1} \in \mathrm{E}^1 \colon \mathrm{Attr_i} \cap \mathrm{Attr_i^1} =$

$\mathrm{Attr_i} \wedge |\mathrm{Attr_i}| \neq |\mathrm{Attr_i^1}|$ и $\exists\, \mathrm{r_i} \in \mathrm{R},\ \mathrm{r_j} \in \mathrm{R^1}\colon\ \mathrm{r_i} \notin \mathrm{R^1} \vee\ \mathrm{r_j} \notin \mathrm{R}$. Обозначив через F_h процедуру горизонтальной эволюции, можно переписать данное выше определение следующим образом: $F_h\big((E,R)\big) = (E^1,R^1) \Longrightarrow \{|\mathrm{E}| = |\mathrm{E^1}|\text{^}\exists\, \mathrm{e_i} \in \mathrm{E}, \mathrm{e_i^1} \in \mathrm{E^1}\colon \mathrm{Attr_i} \cap \mathrm{Attr_i^1} = \mathrm{Attr_i} \wedge |\mathrm{Attr_i}| \neq |\mathrm{Attr_i^1}|\text{^}\exists\, \mathrm{r_i} \in \mathrm{R},\ \mathrm{r_j} \in \mathrm{R^1}\colon\ \mathrm{r_i} \notin \mathrm{R^1} \vee\ \mathrm{r_j} \notin \mathrm{R}\}$.

В главе также показано, что для реализации любого вида эволюции достаточно следующего набора правил:

- создание (добавление) сущности;
- удаление сущности;
- создание (добавление) отношения между сущностями;
- удаление отношения между сущностями;
- создание (добавление) атрибута;
- удаление атрибута.

Важно отметить, что данный набор правил (преобразований) является универсальным и может быть применен для реализации любого вида эволюции любых моделей. Следовательно, он применим и для реализации согласованной эволюции СМПО и моделей всех уровней ПОЯ.

Тогда, можно утверждать, что операции эволюции могут быть формализованы следующим образом: $F_v = f(Rule_{CE}, Rule_{DE}, Rule_{CR}, Rule_{DR})$ и $F_h = g(Rule_{CA}, Rule_{DA}, Rule_{CR}, Rule_{DR})$, где f и g – некоторые функции (последовательности преобразований), формализующие кроссмодельные преобразования, $Rule_{CE}$ и $Rule_{DE}$ – правила (кроссмодельные преобразования) добавления и удаления сущностей модели соответственно, $Rule_{CR}$ и $Rule_{DR}$ – правила (кроссмодельные преобразования) добавления и удаления отношений между сущностями модели соответственно, $Rule_{CA}$ и $Rule_{DA}$ – правила (кроссмодельные преобразования) добавления и удаления атрибутов сущностей модели соответственно.

При этом все определенные выше правила $Rule_{CE}, Rule_{DE}, Rule_{CR}, Rule_{DR}, Rule_{CA}, Rule_{DA}$ не зависят от того, какие сущности/отношения/атрибуты в них используются. Данные правила имеют двойную природу, поскольку применяются к набору некоторых объектов модели и требуют также дополнительного параметра, уточняющего, над каким из объектов набора необходимо провести преобразование.

Учитывая, что F_v и F_h являются суперпозицией указанных выше правил кроссмодельных преобразований, можно также утверждать, что в данном случае процедуры эволюции являются результатом последовательного применения различных правил кроссмодельных преобразований и, следовательно, при их применении также сохраняются обозначенные выше существенные характеристики моделей.

Таким образом, получается набор кроссмодельных преобразований моделей, не зависящий от того, в каком виде представлена СМПО и модели всех уровней ПОЯ. Как следствие, данный набор правил является повторно-используемым и может быть адаптирован для организации эволюции любых видов ПОЯ.

На основе данного набора правил кроссмодельных преобразований, в главе представлены детали алгоритмизации и программной реализации процедуры эволюции человеко-машинного интерфейса программных систем общего назначения. В этом случае алгоритмизация процедур эволюции ПОЯ может быть определена следующим образом:

1. Определяются основные объекты метауровня исходной модели (классы и пр.).
2. Определяются основные объекты метауровня целевой модели.
3. Устанавливается соответствие между объектами метауровня исходной и целевой модели путем определения правил трансформации ATL между ними.

4. Осуществляется применение заданных правил трансформации посредством программной среды выполнения трансформаций (в нашем случае используется Eclipse Modeling Project [15]).

Разработанные программные алгоритмы использованы для реализации программных систем в двух предметных областях, описанных в следующей главе.

Глава 4 содержит описание и анализ программной реализации процедуры эволюции человеко-машинного интерфейса программных систем общего назначения как примера внешнего ПОЯ в двух предметных областях: «Программная система Приемной комиссии ВУЗа» и «Программная система распределения ресурсов ж/д станции». Представлено определение правил кроссмодельных преобразований на языке ATL [9] для реализации горизонтальной и вертикальной эволюции ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения. Разработаны инструментальные средства (модули) для поддержки эволюции ПОЯ для моделирования человеко-машинных интерфейсов программных систем общего назначения.

В частности, показано, что каждая из реализованных программных сред содержит отдельный модуль для организации любых видов эволюции ПОЯ, причем как визуальных (в случае Приемной комиссии), так и текстовых (в случае ж/д станции). При этом внесение изменений в ПОЯ происходит через интерфейс и не требует ручного изменения структуры ПОЯ (Рис. 5, Рис. 6). Внесенные изменения применяются в реальном режиме времени, устраняя необходимость ручного редактирования ПОЯ и ИС в целом.

В случае же ж/д станции представлены два варианта сценариев модификации ПОЯ: для вертикальной и горизонтальной эволюции. В случае горизонтальной эволюции пользователь определяет новую команду в ПОЯ. В последующем для реализации вертикальной эволюции в ПОЯ внедряется

новая сущность, которая также используется к созданной команде в реальном режиме времени.

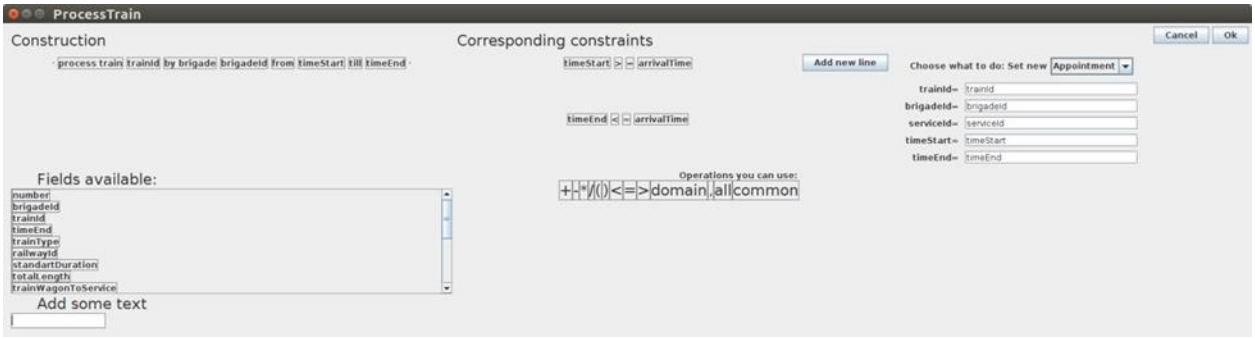

Рис. 5. Интерфейс создания (редактирования) команды ПОЯ

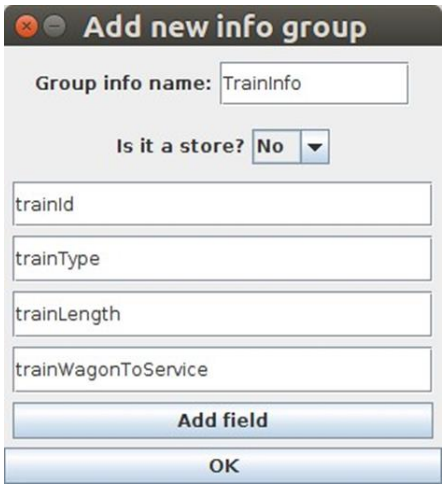

Рис. 6. Интерфейс добавления новой сущности в метамодель ПОЯ

В данном случае при создании новой сущности без рассмотренного подхода, пользователь, должен был бы изменить: грамматику метамодели ПОЯ (описав как сущность в целом, так и все ее атрибуты вместе с доменами), фрагменты синтаксического анализатора для распознавания соответствующих команд, связанных с созданной сущностью, фрагменты ограничений, связанных с создаваемой сущностью – для решателя. Наконец, синтаксические конструкции уровня конкретного синтаксиса ПОЯ. В общей сложности, данные изменения (в рассмотренном выше примере) составили бы порядка 20 0 0 строк исходного кода во всех компонентах системы. В предлагаемой же системе, благодаря внедренной системе правил для реализации согласованной эволюции СМПО и всех уровней ПОЯ, данные изменения вносятся

автоматически и у пользователя отсутствует необходимость изменять исходный код различных компонентов системы.

Более того, пользователь может в целом не обладать глубокими знаниями в области грамматик языка и программирования, поскольку все изменения формулируются в понятной ему (согласованной с предметной областью) форме через интерфейс системы и применяются (транслируются) на все уровни ПОЯ автоматически.

Как следствие, значительно снижается вероятность ошибки при внесении изменений в исходный код приложения, что также является преимуществом предлагаемой системы и подхода в целом.

В главе также приведены результаты оценки характеристик существующих и предложенных программных систем в соответствии с ГОСТ Р ИСО/МЭК 25010-2015 по следующим критериям: функциональные возможности, надежность, практичность, эффективность, мобильность, сопровождаемость и переносимость.

С точки зрения функциональных возможностей можно выделить следующее соответствие критерию качеству:

- *Пригодность* – продемонстрирована на различных кейсах, реализованных с использованием обеих программных сред; проведена апробация представленных решений;
- *Правильность* – результаты, получаемые в процессе работы программных сред, проверялись и оценивались экспертами предметной области. Кроме того, проверка правильности работы программных сред проверялась на архивных данных, после чего происходила оценка полученных результатов с архивным эталонным значением. В результате таких экспериментов в случае Приемной комиссии (сравнение происходило по 183 показателям) – результаты совпали в 100% случаев, в случае ж/д станции распределение составов (сравнением происходило в процессе воспроизведения 54 экспериментов, в каждом случае в среднем рассматривалось 60

составов, прибывающих на станцию) – отклонение от эталонного значения обнаружено в 3 экспериментах, однако, данное отклонение вызвано тем, что предлагаемая система не смогла найти решение за время, которое отводилось на поиск решения (данное время совпадает с временем, за которое решение было найдено существующей системой АРМ «ОСА»). Однако, в целом, найденное программой решение (найденное после истечения отведенного времени) совпадало с эталонным.

- *Способность к взаимодействию* – в данном случае мы рассматриваем только способность к взаимодействию с пользователем, которое построенные программные среды полностью обеспечивают, поскольку взаимодействие осуществляется через графические пользовательские интерфейсы.
- *Согласованность* – построенные программные среды разрабатывались в полном соответствии с актуальными Правилами приема (в случае Приемной комиссии) и Стандартами и Руководствами по грузоперевозкам и управлению грузовыми комплексами ж/д станций (в случае ж/д станции), что позволяет утверждать согласованность построенных решений. Кроме того, реализованные модули эволюции в обеих программных средах позволят в дальнейшем также обеспечивать изменение программных сред в соответствии с изменениями в указанных документах, сохраняя соблюдение требований по согласованности.
- *Защищенность* – вопросы обеспечения защищенности построенных программных сред не входят в данное исследование, поэтому в данном случае мы можем утверждать лишь защищенность построенных программных сред, обеспечивающуюся стандартными средствами: разделение данных и интерфейсов доступа к ним, защита используемых данных на уровне сервера с базой данных и пр. Таким образом,

защищенность построенных программных сред является удовлетворительной.

Как видим, с точки зрения функциональных возможностей построенные программные решения отвечают большинству представленных критериев: пригодность, правильность, способность к взаимодействию, согласованность. Как следствие, можно сделать вывод, что методы, используемые при разработке представленных программных сред, позволяют строить программные решения, для которых важно обеспечение высокой степени согласованности с существующими стандартами и ограничениями предметной области, обеспечивающих взаимодействие с конечным пользователем. Однако, они не применимы в случае, когда речь идет о разработке т.н. ответственных систем, для которых критическим является показатель безопасности.

С точки зрения надежности можно выделить следующее соответствие критерию качеству:

- *Стабильность* – обе рассмотренные программные среды предусматривают возможность проверки пользовательских сценариев, как на этапе их ввода (проверка используемых конструкций), так и на этапе выполнения (поиск конфликтов). Однако, модуль организации эволюции ПОЯ обеих сред требуют ручного контроля со стороны пользователя (в частности, на соответствие предметной области в случае добавления новых сущностей). Кроме того, в случае ж/д станции система имеет ограничения на количество пользователей, одновременно выполняющих сценарии – не более 15 пользователей. В случае, когда количество одновременно работающих пользователей превышало данное ограничение, в системе возникали зависания, а часть сценариев не успевали обработаться. Таким образом, можно утверждать удовлетворительный уровень соответствия программных сред данному критерию качества.

- *Устойчивость к ошибке* – в обеих построенных программных средах имеются встроенные механизмы проверки пользовательских сценариев, как на этапе их ввода (проверка используемых конструкций), так и на этапе выполнения (поиск конфликтов). Как следствие, можно утверждать, что построенные системы устойчивы к ошибкам.

Таким образом, построенные системы обеспечивают достаточное соответствие критериям надежности, являясь устойчивыми к ошибкам. Однако, этого недостаточно для использования представленных подходов в случае высоконагруженных и высокопроизводительных систем, для которых критичным является организация одновременного доступа большого числа пользователей (более 50) и обработки большого числа запросов в реальном режиме времени.

Аналогичные выводы можно сделать, анализируя соответствие представленных программных сред критериям эффективности:

- *Характер изменения во времени* и *Характер изменения ресурсов* – с точки зрения того, что любая модификация конструкций языка и его метамодели ведет к усложнению обработки данных конструкций, предложенные методы обеспечивают лишь удовлетворительное соответствие данному критерию качества. Однако, как было показано выше, это становится критическим только в случае, когда с системой работают более 15 пользователей и часть сценариев остаются не обработанными за выделенное время. В случае же, когда количество пользователей меньше, даже после модификации конструкций языка и добавления новых сущностей, время обработки операций в среднем изменялось на 1,3мс (в случае кейса ж/д станции).

С токи зрения же практичности разработанные программные среды обладают высоким качеством, поскольку удовлетворяют следующим критериям:

- *Понятность* и *Обучаемость* – разработанные программные среды полностью основаны на терминологии и семантической модели

предметной области, как следствие, являются понятными конечным пользователям. Важно отметить, что в случае использования обеих программных сред не понадобилось проводить отдельных мероприятий по обучению, что подтверждает высокую степень соответствия критериям понятности и обучаемости.

- *Простота использования* – сценарии в случае ПОЯ ж/д станции формулируются с использованием команд, применяемых в предметной области, тем самым обеспечивая простоту создания данных сценариев. В случае же Приемной комиссии все взаимодействие происходит посредством графических интерфейсов, что также значительно упрощает возможность генерации различных отчетов по приемной кампании и ввод сведений по абитуриентам.

С точки зрения сопровождаемости представленные программные среды обеспечивают возможность модификации в реальном режиме времени и удовлетворяют следующим критериям:

- *Изменяемость* и *Тестируемость* – обе среды содержат отдельные модули организации эволюции ПОЯ, обеспечивающие возможность модификации его конструкций через интерфейс в реальном режиме времени, без необходимости внесения ручных изменений. Так, в случае модификации основной структуры модуля Приемной комиссии – Абитуриента (добавление дополнительного атрибута), общее число добавленных строк кода составило 479. Без предлагаемого подхода все эти строки необходимо было бы внести вручную, в случае же предлагаемой системы все они генерируются автоматически без вмешательства конечного пользователя. Внесенные изменения сразу же доступны конечному пользователю, поэтому их можно сразу проверить на корректность, что говорит о простоте тестирования внесенных изменений.

Таким образом, представленные подходы могут применяться для разработки систем (и ПОЯ для таких систем), для которых важно обеспечить

поддержку возможности простого и быстрого внесения изменений, без необходимости ручного изменения на уровне исходного кода.

Наконец, с точки зрения мобильности и переносимости построенные системы удовлетворяют следующих критериям:

- *Адаптируемость* и *Соответствие* – аналогично соответствию изменяемости, построенные программные среды обеспечивают высокую степень адаптивности построенных решений в реальном режиме времени, а также обеспечивают сохранение соответствия различным Положениям и ограничениям предметной области. В этой связи также важно отметить кроссплатформенность (и, как следствие, переносимость) построенных решений, поскольку при их разработке использовался язык Java.

Таким образом, построенные программные среды обеспечивают высокую степень адаптивности. Как следствие, подходы, применяемые при их разработке, могут применяться для построения других систем, для которых необходимо обеспечение простоты внесения изменений (адаптивности) и сохранения соответствия существующим ограничениям (и положениям) предметной области.

Подводя итог представленному анализу качества построенных программных сред, можно утверждать, что применяемые при из разработке подходы, методы и алгоритмы могут быть также применены в случае построения адаптивных систем общего назначения. Однако, не рекомендуется их использование для высоконагруженных, высокопроизводительных, ответственных систем, в которых важно обеспечение высокого уровня безопасности и быстродействия системы в случае большого числа одновременных пользователей и запросов.

В заключении работы содержатся перечисление основных результатов исследования, оценка уровня достижения поставленной цели, а также предложения по дальнейшему развитию и практическому применению полученных результатов в различных предметных областях.

ВЫВОДЫ

Таким образом, в рамках данной работы были рассмотрены вопросы разработки и эволюции предметно-ориентированных языков моделирования интерфейсов для программных систем общего назначения в условиях динамических контекстов. Целью исследования является повышение эффективности процессов поддержки жизненного цикла программных систем общего назначения путем разработки новых моделей и методов эволюции ПОЯ для моделирования интерфейсов. Предлагаемый в этой работе проекционный подход позволяет изменять структуру всех уровней ПОЯ в реальном режиме времени без необходимости пересоздания приложения в целом. В этом заключается основное отличие собственного решения от существующих подходов [10, 13, 14, 24], основная идея которых заключается в отказе от реализации полноценной эволюции ПОЯ в пользу пересоздания структуры языка в случае необходимости его модификации.

Научная новизна проведенного исследования заключается в следующем:

- на основе анализа существующих подходов и методов к разработке ПОЯ представлена обобщенная модельно-ориентированная структура ПОЯ (Раздел 2.3), представляющая собой унифицированное представление всех уровней структуры ПОЯ;
- формализованы различные виды (модели) эволюции ПОЯ для моделирования человеко-машинного интерфейса для программных систем общего назначения (Раздел 3.2), что позволяет определить согласованную эволюцию СМПО и всех уровней ПОЯ в соответствии с предлагаемым проекционным подходом;
- построен и реализован набор кроссмодельных преобразований (на основе графовых преобразований – представлена в Разделе 3.3) для организации горизонтальной и вертикальной эволюции внешнего ПОЯ моделирования интерфейсов программных систем общего назначения;

- на основе выделенной системы кроссмодельных преобразований предложен новый метод (названный проекционным подходом) к разработке внешнего ПОЯ (Раздел 3.1), позволяющий автоматизировать процесс разработки ПОЯ для моделирования интерфейсов программных систем общего назначения и использовать результаты анализа предметной области при проектировании и реализации ПОЯ, а также организовать автоматизированную эволюцию ПОЯ;
- в соответствии с предложенным методом выполнена алгоритмизация (Разделы 3.2 и 3.3) и программная реализация (Разделы 4.2 и 4.3) процедуры трансформации человеко-машинного интерфейса как примера внешнего ПОЯ для двух предметных областей.

В рамках практической части работы реализованные прототипы информационных сред для двух предметных областей: «Программная система Приемной комиссии ВУЗа» и «Программная система распределения ресурсов ж/д станции». Построенные прототипы основаны на предлагаемом проекционном подходе, предоставляя пользователю не только функционал для решения задач предметной области, но также и для организации согласованной эволюции всех уровней ПОЯ и СМПО.

В отличие от существующих предметно-ориентированных решений для различных предметных областей [18, 26, 34], представленные программные прототипы содержат отдельные модули для организации эволюции ПОЯ без необходимости внесения ручных изменений в различные уровни структуры ПОЯ. При этом создаваемые диалекты ПОЯ сохраняют совместимость с ранее определенными, поскольку изменения вносятся не только на уровне конкретного синтаксиса ПОЯ, но переносятся, с использованием предложенных авторами правил кроссмодельных преобразований, на уровни абстрактного синтаксиса ПОЯ и СМПО. Тем самым, обеспечивается полная согласованность эволюции на всех уровнях ПОЯ и СМПО.

Для достижения данных результатов были использованы такие формальные математические методы, как элементы теории функций (Раздел 3.3) – для формализации связи между МКЕ и МКК и реализации кроссмодельных преобразований, элементы теории графов (Раздел 2.4, Разделы 3.2 и 3.3) – для формализации преобразований между моделями СМПО и ПОЯ.

Предлагаемый в данной работе проекционный подход к реализации ПОЯ существенно отличается от классического подхода [35, 37], в котором СМПО рассматривается лишь как базис для разработки ПОЯ и никак не используется в качестве артефакта в процессе реализации ПОЯ. В случае же предлагаемого подхода разработка ПОЯ ведется последовательно, начиная от определения СМПО с последующим преобразованием через кроссмодельные преобразования в семантическую модель (метамодель) ПОЯ и трансформацией до уровня конкретного синтаксиса с последующим определением отдельных команд в случае текстового ПОЯ. Основываясь на идеях кроссмодельных преобразований и объектном представлении используемых моделей, подход гарантирует согласованность всех уровней ПОЯ с СМПО, поскольку в основе формальных механизмов трансформаций используются графовые преобразования с сохранением существенных характеристик (свойств) моделей.

Эффективность и гибкость предлагаемого подхода продемонстрирована на примере использования в процессе разработки двух различных программных систем общего назначения. Важно отметить, что подход является универсальным и применяется как к визуальным, так и к текстовым видам ПОЯ для программных систем, что продемонстрировано в рассматриваемых ситуациях.

Предложенный в работе подход, модели и методы могут применяться для построения адаптивных систем общего назначения. Однако, не рекомендуется их использование для высоконагруженных,

высокопроизводительных, ответственных систем, в которых важно обеспечение высокого уровня безопасности и быстродействия системы в случае большого числа одновременных пользователей и запросов.

В результате реализации программного обеспечения для двух разных предметных областей выделены повторно-используемые структуры данных и алгоритмы кроссмодельных преобразований для поддержки эволюции ПОЯ. Таким образом сформирована основа для создания инструментальных средств общего назначения.

В дальнейшем возможным расширением подхода может стать реализация полноценной универсальной среды для разработки ПОЯ (по примеру MetaEdit+ [32]), позволяющей не только модифицировать набор созданных конструкций ПОЯ, но также поддерживать организацию эволюции любого ПОЯ, создаваемого инструментами среды.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Ануреев, И.С. Предметно-ориентированные системы переходов: объектная модель и язык // Системная информатика. – 2013. – №. 1. – с. 1-34.
2. Гайфуллин, Б.Н., Туманов, В.Е. Предметно-ориентированные системы научной осведомленности в науке и образовании // Современные информационные технологии и ИТ-образование. – 2012. – №. 8. – с. 741-750.
3. ГОСТ Р ИСО/МЭК 25010-2015: СИСТЕМНАЯ И ПРОГРАММНАЯ ИНЖЕНЕРИЯ Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов // Москва: Стандартинформ. – 2015.
4. Ершов, А.П. Денотационный подход к описанию трансформационной семантики: Слайды доклада на семинаре проекта CIP в Мюнхенском техническом университете. – 1982.

5. Конструирование канонических информационных моделей для интегрированных информационных систем / В. Н. Захаров, Л. А. Калиниченко, И. А. Соколов, С. А. Ступников // Информатика и ее применения. — 2007. — Т. 1, № 2. — с. 15–38.
6. Сухов, А.О. Классификация предметно-ориентированных языков и языковых инструментариев // Математика программных систем. – 2012. – с. 74-83.
7. Федоренков, В.Г., Балакшин, П.В. Особенности применения предметно-ориентированных языков для тестирования веб-приложений // Программные продукты и системы. – 2019. – №4. – с. 601-606.
8. Akehurst, D., Kent, S. A relational approach to defining transformations in a metamodel // J.-M. Jézéquel, H. Hussmann, and S. Cook (eds.), Proc. Fifth International Conference on the Unified Modeling Language – The Language and its Applications, LNCS. – 2002. – vol. 2460. – pp. 243–258.
9. ATL Transformation Language [Электронный ресурс] – URL: http://www.eclipse.org/atl/.
10. Bell, P. Automated Transformation of Statements within Evolving Domain Specific Languages // Computer Science and Information System Reports. – 2007. – pp. 172–177.
11. Bergmann, G. et al. Change-driven model transformations / G. Bergmann, I. Ráth, G. Varró, D. Varró // Software & Systems Modeling. – 2021. – v. 11(3). – pp. 431-461.
12. Cabot, J. et al. Verification and validation of declarative model-to-model transformations through invariants / J. Cabot, R. Clarisó, E. Guerra, J. de Lara // J Syst Softw. – 2010. – v. 83(2). – pp. 283–302.
13. Cleenewerck, T. Component-Based DSL Development // Software Language Engineering. – 2003. – pp. 245-264.
14. Cleenewerck, T. Evolution and Reuse of Language Specifications for DSLs (ERLS) / T. Cleenewerck, K. Czarnecki, J. Striegnitz, M. Volter // Object-

Oriented Technology. ECOOP 2004 Workshop Reader. – 2004. – pp. 187-201.

15. Eclipse Graphical Modeling Project (GMP) [Электронный ресурс] – URL: http://www.eclipse.org/modeling/gmp/.
16. Evans, E. Domain-Driven Design: Tackling Complexity in the Heart of Software. – Addison-Wesley, 2013. – 560 p. – ISBN 978-0-321-12521-7.
17. Fowler, M. Domain specific languages / M. Fowler, R. Parsons. – Addison Wesley, 2010. – 413 p. – ISBN: 978-0-321-71294-3.
18. Gómez-Abajo, P. A domain-specific language for model mutation and its application to the automated generation of exercises / Pablo Gómez-Abajo, Esther Guerra, Juan de Lara // Computer Languages, Systems & Structures. – 2016. – v. 49. – pp. 152-173.
19. Guizzardi, G. Ontological foundations for structural conceptual models. – Enschede, The Netherlands, 2005. – 418 p. – ISBN 90-75176-81-3.
20. Guizzardi, G., Halpin, T. Ontological Foundations for Conceptual Modeling // Applied Ontology. – 2008. – v. 3. – pp. 91-110.
21. Haav, H.-M. et al. Ontology-Based Integration of Software Artefacts for DSL Development / H.-M. Haav, A. Ojamaa, P. Grigorenko, V. Kotkas // On the Move to Meaningful Internet Systems: OTM 2015 Workshops. Lecture Notes in Computer Science. – 2015. – v. 9416. – pp. 309-318.
22. Hausmann, J.H., Heckel, R., Sauer, S. Extended model relations with graphical consistency conditions // UML 2002 Workshop on Consistency Problems in UML-based Software Development. – 2002. – pp. 61–74.
23. Kelly, S., Tolvanen J.-P. Domain-specific modeling: enabling full code generation. – Hoboken, New Jersey, USA: Wiley-IEEE ComputerSociety Press, 2008. – 444 p. – ISBN 978-0-470-03666-2.
24. Kessentini, W., Sahraoui, H., Wimmer, M. Automated metamodel/model co-evolution: A search-based approach // Information and Software Technology. – 2019. – pp. 49-67.

25. Kogalovsky M. R., Kalinichenko L. A. Conceptual and ontological modeling in information systems // Programming. – 2009. – v. 35 (5). – pp. 241-256.
26. Kosar, T., Bohra, B., Mernik, M.: Domain-Specific Languages: A Systematic Mapping Study // Information and Software Technology. – 2016. – pp.77-90.
27. Laird, P., Barrett, S. Towards Dynamic Evolution of Domain Specific Languages // Software Language Engineering. – 2010. – pp. 144-153.
28. Luoma, J., Kelly, S., Tolvanen, J.-P. Defining domain-specific modeling languages: Collected experiences // Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04). – 2004.
29. Mengerink, J.G.M. et al. Udapt Edapt Extensions for Industrial Application / J.G.M. Mengerink, A. Serebrenik, M. van den Brand, R.R.H. Schiffelers // ITSLE 2016 Industry Track for Software Language Engineering October 31, 2016, Amsterdam, Netherlands. – 2016. – pp. 21-22.
30. Mengerink, J.G.M. et al. A Complete Operator Library for DSL Evolution Specification / J.G.M. Mengerink, A. Serebrenik, R.R.H. Schiffelers, M.G.J. van den Brand // MDSE 32nd International Conference on Software Maintenance and Evolution, 2016. – 2016. – pp. 144-154.
31. Mernik, M., Heering, J., Sloane, A. When and how to develop domain-specific languages // ACM computing surveys (CSUR). – 2005. – v. 37, no. 4. – pp. 316-344.
32. MetaCase+ [Электронный ресурс] – URL: http://www.metacase.com/
33. Milne, C., Strachey, R. A Theory of Programming Language Semantics (2 Vol). – Chapman & Hall, 1976. – 868 p. – ISBN 978-0-412142-60-4.
34. Parr, T. (2012). Language Implementation Patterns: Create Your Own Domain-Specific and General Programming Languages. – Pragmatic Bookshelf, 2012. – 389 p. – ISBN 978-1-934356-45-6.
35. Pereira, M.J.V. et al. Ontological approach for DSL development / M.J.V. Pereira, J. Fonseca, P.R. Henriques // Computer Languages, Systems & Structures. – 2016. – v. 45. – pp. 35-52.

36. Popovic, A.A. et al. DSL for modeling application-specific functionalities of business applications / A. Popovic, I. Lukovic, V. Dimitrieski, V. Djuki // Computer Languages, Systems & Structures. – 2015. – pp. 69-95.
37. Schürr, A. Graph-Transformation-Driven Correct-by-Construction Development of Communication System Topology Adaptation Algorithms // I. Schaefer, D. Karagiannis, A. Vogelsang, D. Méndez, C. Seidl (Eds.): Modellierung, LNI. – 2018. – pp. 15–29.
38. Sprinkle, J. A domain-specific visual language for domain model evolution // Journal of Visual Languages & Computing. – 2004. – pp. 291–307.
39. Stoy, J.E. Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory. – MIT Press, 1985. – 450 p. – ISBN: 978-0-262-69076-8.
40. Ulitin, B., Babkin, E., Babkina, T. A Projection-Based Approach for Development of Domain-Specific Languages // Lecture Notes in Business Information Processing Issue 330: Perspectives in Business Informatics Research. – 2018. – pp. 219-234.