Анализ непротиворечивости моделей архитектуры предприятия с использованием формальных методов верификации[1]

Э.А. Бабкин
кандидат технических наук, PhD in Computer Science
профессор кафедры информационных систем и технологий
Национальный исследовательский университет «Высшая школа экономики»
Адрес: 603155, г. Нижний Новгород, ул. Большая Печерская, д. 25/20
E-mail: eababkin@hse.ru

Н.О. Пономарев
студент магистерской программы «Бизнес-информатика»
Национальный исследовательский университет «Высшая школа экономики»;
инженер группы разработки программного обеспечения АО «Интел»
Адрес: 603155, г. Нижний Новгород, ул. Большая Печерская, д. 25/20
E-mail: nik4nikita@gmail.com

Аннотация

Разработка архитектуры предприятия является сложным процессом, но в то же время позволяет решить проблему синхронизации возможностей и потребностей бизнеса и информационных технологий (ИТ). Решение данной проблемы достигается за счет уточнения понимания и формализации описания бизнес-процессов и взаимодействия элементов системы путем их формального описания. Наличие большого числа взаимодействующих бизнес-процессов и сущностей архитектуры предприятия объясняет необходимость проверки их корректности. Поэтому необходимо их автоматически верифицировать, на основе формализации требований к архитектуре.

В данной работе предлагается метод обнаружения логических противоречий в моделях архитектуры предприятия, на основе частного случая подхода к проверке моделей, примененного в бизнес-моделировании. В качестве языка описания архитектуры предприятия в работе используется современный открытый и независимый язык ArchiMate. Разрабатываемый The Open Group стандарт предоставляет общую спецификацию для описания построения и функционирования бизнес-процессов, организационных структур, информационных потоков, ИТ-систем и технической инфраструктуры предприятия. В качестве верификатора выбран формальный язык реляционной логики и инструментарий системы MITAlloy Analyzer, позволяющий выполнять анализ ограничений модели в терминах реляционной логики путем автоматической генерации структур, которые удовлетворяют требованиям логической модели.

В данной работе предлагается упростить и автоматизировать процесс спецификации и верификации моделей предметной области архитектуры предприятия с помощью визуального редактора конструирования моделей на языке ArchiMate – Archi. Разработанный авторами плагин к редактору осуществляет трансляцию моделей архитектуры предприятия на язык системы Alloy Analyzer и в качестве основы для построения конкретных моделей предметной области использует метамодель ключевых элементов спецификации ArchiMate. Предложенный метод и программные решения апробированы на примере пользовательского кейса архитектуры предприятия компании ArciSurance.

Ключевые слова: архитектура предприятия, ArchiMate, Alloy Analyzer, верификация, анализ непротиворечивости, формальные методы.

Цитирование: Бабкин Э.А., Пономарев Н.О. Анализ непротиворечивости моделей архитектуры предприятия с использованием формальных методов верификации // Бизнес-информатика. 2017. № 3 (41). С. 30–40. DOI: 10.17323/1998-0663.2017.3.30.40.

[1] Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 16-06-00184 А «Разработка и исследование моделей online-дискуссии на материале обсуждения политических новостей»

Введение

Консалтинговая компания Gartner определяет термин «архитектура предприятия» как дисциплину для инициативного и всестороннего реагирования предприятия на разрушительные силы путем выявления, анализа и выполнения изменений в направлении желаемого видения и результатов бизнеса [1]. Архитектура предприятия помогает руководителям находить оптимальные стратегии поддержки и развития организации и ее изменений относительно информационных систем, от которых она сильно зависит. Однако в процессе трансформации и постепенного усложнения архитектуры серьезной проблемой становится отсутствие практической возможности выполнять ручную проверку модели по заранее сформулированным требованиям. Подобная проверка носит название верификации [2, 3].

В данной работе развивается общий подход формальной верификации с использованием принципов *model checking* (проверка моделей) применительно к архитектуре предприятия. В этом случае требуемые свойства модели выражаются формулами определенного диалекта формальной логики, а проверка согласованности и непротиворечивости сводится к исчерпывающему анализу всего пространства ее состояний [4]. По сравнению с другими подходами данный метод обладает двумя существенными преимуществами. Он может быть полностью автоматизирован, и его применение не требует от бизнес-аналитика специальных знаний в области математической логики и теории доказательств теорем. В работе предлагается новый метод обнаружения логических противоречий в моделях архитектуры предприятия на основе формальных требований. Формальная верификация архитектуры, а также ее бизнес-процессов, должна обеспечить возможность построения более надежной архитектуры.

Объектом исследования является известный кейс Open Group – архитектура страховой компании ArchiSurance [5]. Предметом исследования является логическая непротиворечивость основных элементов архитектуры и бизнес-процессов и сервисов данного предприятия. В качестве языка описания архитектуры предприятия в работе используется современный открытый и независимый язык моделирования архитектуры предприятия на основе стандарта ArchiMate [6]. Этот стандарт предоставляет общую спецификацию для описания, построения и функционирования бизнес-процессов, организационных структур, информационных потоков, ИТ-систем и технической инфраструктуры предприятия.

В качестве инструмента верификации выбран язык и логика системы Alloy Analyzer (*http://alloy.mit.edu/alloy/*) [7]. Инструмент позволяет выполнять анализ ограничений модели в терминах реляционной логики путем автоматической генерации структур, которые удовлетворяют требованиям логической модели.

Авторы считают, что наибольший эффект применение формальных методов дает в случае тесной интеграции с программной средой для моделирования архитектуры предприятия. Разработанное интегрированное программное решение в этом случае позволяет автоматизировать процесс формализации модели архитектуры и ее верификации. Следуя этому принципу, в рамках данного исследования необходимо разработать метамодель спецификации ArchiMate, на основе которой будет строиться пользовательская модель в определенной среде моделирования.

В данной статье результаты представлены следующим образом. В разделе 1 изложены основные проблемы построения архитектуры предприятия и специфика языка ArchiMate. В разделе 2 показан принцип верификации моделей. Раздел 3 посвящен анализу доступных средств верификации, здесь же более детально раскрыты сведения об используемом языке и логике системы MIT Alloy Analyzer. В разделе 4 представлены основные результаты создания метамодели ArhiMate, а раздел 5 посвящен описанию предлагаемого общего алгоритма преобразования модели из ArchiMate в конструкции языка MIT Alloy Analyzer. Раздел 6 описывает процесс создания формальной модели рассматриваемого примера. Раздел 7 содержит результаты верификации примера представления архитектуры предприятия на основе предложенного алгоритма. Наконец, раздел 8 посвящен деталям интеграции представленного подхода в инструмент моделирования Archi. В Заключении подводятся итоги и определяются пути дальнейших исследований.

1. Архитектура предприятия и проблемы ее построения

На практике архитектура предприятия обычно разрабатывается по причине того, что у некото-

рых «заинтересованных сторон» (стейкхолдеров) есть определенные сомнения по поводу функционирования бизнеса и ИТ-систем. Роль архитектора предприятия состоит в том, чтобы устранить эти проблемы, определив и уточнив требования стейкхолдеров, разработать представления архитектуры, показывающие, как будут решаться проблемы с учетом требований и компромиссов, которые необходимо согласовывать с потенциально противоречивыми интересами сторон и в итоге синхронизовать возможности и потребности бизнеса и ИТ [6, 8]. Решение данной проблемы достигается за счет уточнения понимания и формализации описания бизнес-процессов и взаимодействия элементов системы путем формального их описания.

В качестве языка описания архитектуры предприятия эта работа использует современный, открытый и независимый язык ArchiMate 2.1. Его популярность и известность подтверждается большим количеством сертифицированных организаций в мире (4314) (*http://www.togaf.info/archimate-visualmap.html*), а число участников ежегодного ArchiMate Forum достигает 121 (*http://reports.opengroup.org/membership_report_archimate_forum.pdf*). Среди них такие известные организации, как Boeing, Dell, IBM, Philips и многие другие. Данный язык позволяет представить архитектуру в виде набора представлений, которые, в зависимости от потребностей, могут включать только элементы на одном уровне или могут показывать вертикальные отношения между уровнями.

К числу уровней относятся:

✦ бизнес-уровень, который предлагает продукты и услуги внешним клиентам;

✦ уровень приложения, который поддерживает бизнес-уровень с помощью сервисов приложений, реализуемых программными приложениями;

✦ технологический уровень, который предоставляет инфраструктурные услуги, необходимые для поддержки программных систем.

Аспекты:

✧ аспектом активной структуры являются различные компоненты, отображающие фактическое поведение, т. е. «субъекты» деятельности;

✧ аспект поведения представляет процессы, функции, события, выполняемые субъектами;

✧ аспект пассивной структуры представляет объекты (физические или информационные), на которых выполняется поведение.

2. Построение и верификация моделей

Наличие большого числа взаимодействующих бизнес-процессов и сущностей архитектуры предприятия в составе моделей ArchiMate ставит задачу проверки их корректности. Данная задача может быть выполнена с помощью метода проверки моделей. Проверка моделей – это метод проверки того, что на заданной формальной модели системы заданная логическая формула выполняется, то есть принимает истинное значение [9].

Метод включает несколько этапов: моделирование, спецификацию и верификацию. Первая задача состоит в том, чтобы привести проектируемую модель архитектуры к некому формальному виду, приемлемому для инструментальных средств верификации моделей программ. На этапе спецификации необходимо сформулировать свойства, которыми должна обладать проектируемая модель архитектуры предприятия, на языке формальной логики.

Верификация модели может показать, соответствует ли проектируемая система заданной формальной спецификации, но определить, охватывает ли данная спецификация все свойства системы не представляется возможным. Этап верификации, в идеале, должен проводиться автоматически, однако на практике чаще всего необходимо вмешательство человека. В случае отрицательного результата верификации будет сгенерирован контрпример, который позволит пользователю отследить места возникновения ошибки и исправить ее. Также возможно, что формальный вид системы или требования к ней были описаны некорректно. Результат верификации также должен помочь выявить эти проблемы [10].

3. Сравнительный анализ инструментария верификации

В ходе работы были рассмотрены наиболее популярные языки моделирования и анализа абстракций: B [11], OCL [12], VDM [13], Z [14] и Alloy [7]. Все они способны описать любую сложную структуру в краткой и абстрактной форме, и у каждого есть активное сообщество пользователей и исследователей.

Z. Язык Z был впервые разработан в 1977 году Жан-Раймондом Абриалем в Оксфордском университете и основан на логике и теории множеств. Одним из преимуществ Z является то, что он имеет богатую математическую нотацию, делая его выразительным языком. Четкий стиль вычислительной нотации дает ему возможность поддерживать

множество разных идиом. Однако автоматическое доказательство теорем в Z ограничено. Оно автоматизировано только до определенной степени, сложным доказательствам часто требуется руководство от опытного пользователя.

OCL. Объектный язык ограничений (OCL) – это язык ограничений UML, разработанный в IBM и ObjecTime Limited и добавленный в UML в 1997 году, который изначально разработан как язык аннотации для диаграмм классов UML. OCL основан на логике предикатов первого порядка, но использует синтаксис, подобный языкам программирования, и очень тесно связан с синтаксисом UML. OCL позволяет смешивать декларативные элементы и элементы операций. Однако данный язык слишком ориентирован на реализацию и поэтому не подходит для концептуального моделирования.

VDM. VDM (Vienna Development Method) представляет собой набор методов для разработки компьютерных систем. Он возник из лаборатории IBM в Вене в середине 1970-х годов и был разработан Клиффом Джонсом и Динсом Бьеорнером. Однако, все существующие инструменты не обеспечивают полностью автоматический анализ в стиле проверки модели. VDM поддерживает объектно-ориентированную парадигму и параллельность. Хотя это один из первых формальных методов разработки ИТ-систем, он был усовершенствован, стандартизирован и по-прежнему широко используется в промышленности.

B. Язык B был разработан Жаном-Раймондом Абриалом, одним из создателей Z. Он включает в себя язык и способ получения реализаций из абстрактных моделей путем поэтапной обработки. Язык спецификации, Abstract Machine Notation (AMN), отражает свою суть в названии: система для языка рассматривается (как в VDM и Z) как конечный автомат с операциями над глобальным состоянием. Начиная с очень абстрактного автомата, детали добавляются по одному слою за раз, пока не будет получен автомат, который может быть переведен непосредственно в код. Однако, по сравнению с тем же Z, язык B более низкоуровневый, похож на абстрактный язык программирования и сильнее сфокусирован на уточнении кода, а не на спецификации систем.

Alloy. Инструментарий MIT Alloy Analyzer, разработанный в Массачусетском технологическом институте под руководством Даниэля Джексона, позволяет обозначить и обнаружить противоречия в проектируемых моделях систем.

Alloy – это язык структурного моделирования, основанный на логике первого порядка, для выражения сложных структурных ограничений и поведения. Язык Alloy берет свое начало от языка спецификаций Z и реляционных исчислений Тарского, рассматривая отношения как основную единицу анализа, и использует реляционную композицию в качестве мощного оператора для объединения различных структурированных данных.

MIT Alloy Analyzer – это инструмент для анализа моделей, написанных на Alloy. Он поддерживает два вида автоматического анализа: поиска сущности, удовлетворяющей ограничению, и поиск контрпримера для заданных суждений модели.

Для ограничения размеров поиска используется функционал охвата (scope), который фиксирует количество сущностей и контрпримеров, анализируемых данной командой. Alloy анализирует ограничения модели и подбирает структуры, которые им удовлетворяют. Структуры отображаются графически, и их внешний вид может быть настроен вручную.

В ходе оценки был выбран язык Alloy, являющийся полноценным структурированным декларативным языком моделирования, который способен выразить всевозможные сложные структурные ограничения, отразить логику поведения моделей и проводить автоматический анализ.

4. Мета-моделирование

Для нахождения противоречий в моделях архитектуры предприятия с использованием инструментария MIT Alloy Analyzer необходимо создать метамодель спецификации ArchiMate, на основе которой в последствии будет осуществляться построение конкретных моделей предметной области. Метамодель – это некая модель языка, которая фиксирует его основные свойства, языковые концепции и семантику [4].

Основу разработанной метамодели составляют несколько ключевых сущностей спецификации ArchiMate 2.1, общих для всех уровней представлений. От них наследуются и расширяют их свойства все остальные элементы на каждом из уровней языка. Ключевое слово «abstract» языка Alloy подчеркивает свойство сигнатуры, что данная сущность не включает элементов за пределами ее расширений. В предложенной архитектуре метамодели все сигнатуры формальной модели системы расширяют сущность Element.

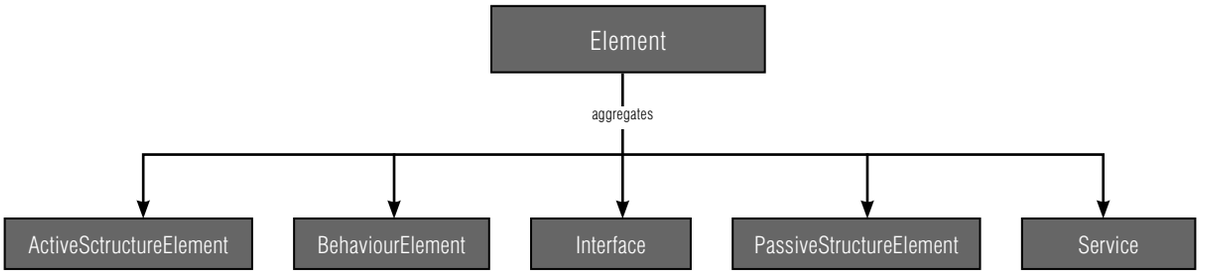

Рис. 1. Графическое представление метамодели верхнего уровня AcrhiMate

В графическом виде соответствующий фрагмент метамодели верхнего уровня ArchiMate представлен на *рисунке 1.*

Определив все необходимые базовые сущности, расширим иерархию метамодели в сторону бизнес-уровня, который отражает все сущности и связи из спецификации ArchiMate 2.1[2].

Далее по тому же принципу задаем метамодель уровня приложений на языке Alloy из спецификации ArchiMate 2.1[3]. В данном случае функционал некоторых типов схож, например, ApplicationFunction и ApplicationInteraction. Чтобы не дублировать связи и код модели, выделим базовый класс со всеми общими типами связи и дополним его функционал для каждого из типов.

Последний уровень метамодели – уровень технологий. Он описывается по аналогии с предыдущими уровнями в соответствии со спецификацией ArchiMate 2.1 на языке Alloy[4]. Таким образом, архитектура метамодели имеет иерархичную структуру: есть верхний общий слой, от которого наследуются три модуля уровней ArchiMate, которые, в свою очередь, будут использованы сущностями пользовательского кейса.

5. Алгоритм преобразования модели

Моделирование сущностей предметной области полностью строится на основе сущностей созданной метамодели [4, 10]. Поэтому перед описанием формальной модели архитектуры необходимо импортировать модуль разработанной метамодели ArchiMate, команды Alloy "*openArchimateMeta-mode*".

Алгоритм преобразования из ArchiMate в Alloy предлагается реализовать следующим образом. Каждая сущность модели представления преобразуется в сигнатуру с расширением, соответствующим типу сущности. В имени сигнатуры пробелы заменяются на нижний слеш, а символы верхнего регистра – на соответствующие символы нижнего регистра. Таким образом, создается единое уникальное имя для сигнатуры. Связи, которые исходят из сущности, преобразуются в ограничения (факты) сигнатуры. Если же сущность не имеет каких-либо связей, которые описаны в сигнатуре метамодели, от которой она наследуется, то ограничение для этой связи задается как "*none*", то есть запрещает создавать контрпример с данной связью при верификации (*таблица 1*).

Таблица 1.

Отображение сущностей ArchiMate в Alloy

ArchiMate	Alloy
Элементы модели	Сигнатура (sig "name") с расширением (extends) типа сущности (ex. BusinessFunction)
Связи	Список ограничений (fact) для конкретной сигнатуры

6. Создание формальной модели предметной области

В качестве кейса по верификации рассмотрим классический пример, разработанный для иллюстрации использования языка моделирования ArchiMate в контексте структуры TOGAF. В нем описывается базовая архитектура компании, а также ряд сценариев изменений [5].

[2] В графическом виде диаграмма метамодели доступна по ссылке: *https://goo.gl/ZLr4Qv*
[3] В графическом виде диаграмма метамодели доступна по ссылке: *https://goo.gl/J9TTHX*
[4] В графическом виде диаграмма метамодели доступна по ссылке: *https://goo.gl/Yo44Dw*

Кейс касается страховой компании ArchiSurance, которая была образована в результате слияния трех ранее независимых компаний:

✦ Home & Away (страхование домовладельцев и путешествий);

✦ PRO-FIT (автострахование);

✦ Legally Yours (страхование юридических расходов).

В настоящее время компания состоит из трех подразделений с такими же названиями и штаб-квартирами, как и у их независимых предшественников.

ArchiSurance создавалась с целью использования эффекта синергии между тремя организациями, чтобы контролировать свои затраты, поддерживать удовлетворенность своих клиентов и инвестировать в новые технологии. Новая компания предлагает все страховые продукты трех компаний, которые были объединены, а ее организационная структура выглядит следующим образом (*рисунок 2*).

Полностью охватить на одной диаграмме все аспекты архитектуры предприятия – задача непростая и объемная. Чаще всего пользователей интересует лишь определенная сторона архитектуры, конкретный ее аспект. Поэтому в ArchiMate присутствует понятие «представление». Это некая точка зрения, которая позволяет довольно гибко работать в общей архитектуре, акцентируя внимание на важных аспектах, как отдельных, так в связке разных уровней.

В данной работе будет рассмотрено многослойное и наиболее общее представление архитектуры предприятия ArchiSurance, разделенное на три уровня по спецификации ArchiMate[5].

В соответствии с описанным ранее алгоритмом преобразования, получаем модель на языке Alloy для последующего формального анализа. Модель состоит из 67 сигнатур и, соответственно, из такого же количества фактов. Каждый факт имеет от нуля до шести ограничений по связям.

7. Определение формальных требований предметной области и верификация в MIT Alloy Analyzer

На этапе конструирования архитектуры сформулированы свойства, которыми должна обладать проектируемая модель архитектуры предприятия. Одним из таких свойств является следующее высказывание: «Есть ли такие компоненты приложений или их функционал, которые используют данные, но не имеют доступа к технологическим сервисам их изъятия?».

Чтобы провести верификацию, необходимо описать данное высказывание на языке формальной логики Alloy, а также определить охват поиска. Необходимо, чтобы поиск решения осуществлялся при наличии всех сущностей представления, поэтому запрос дополнен списком всех элементов модели с ключевым словом «exactly».

При запуске поиска контрпримера для заданного суждения модели существенную роль играет сколемизация. Часто квантифицированные формулы могут быть сведены к эквивалентным формулам без использования кванторов. Это сокращение называется сколемизацией и основано на введении одной или нескольких констант Сколема или функций, которые фиксируют ограничение количественной формулы по их значениям.

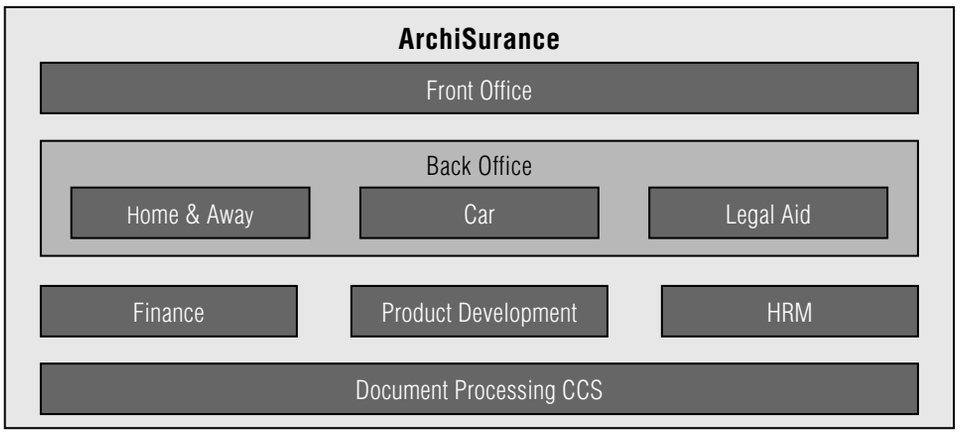

Рис. 2. Организационная структура компании ArciSurance

[5] В графическом виде диаграмма доступна по ссылке: *https://goo.gl/XdvNLy*

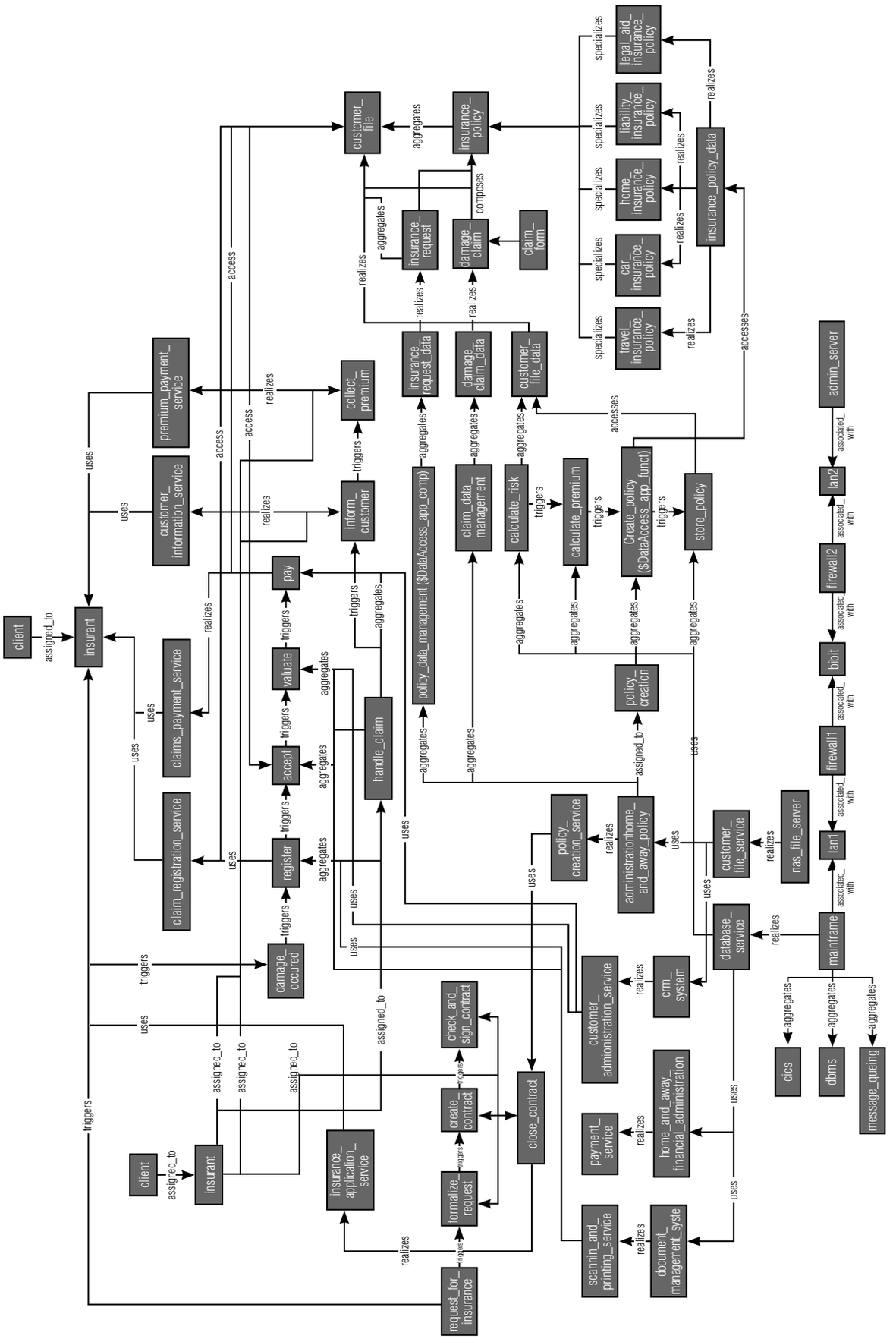

Рис. 3. Кейс ArchiSurance: результат верификации представления

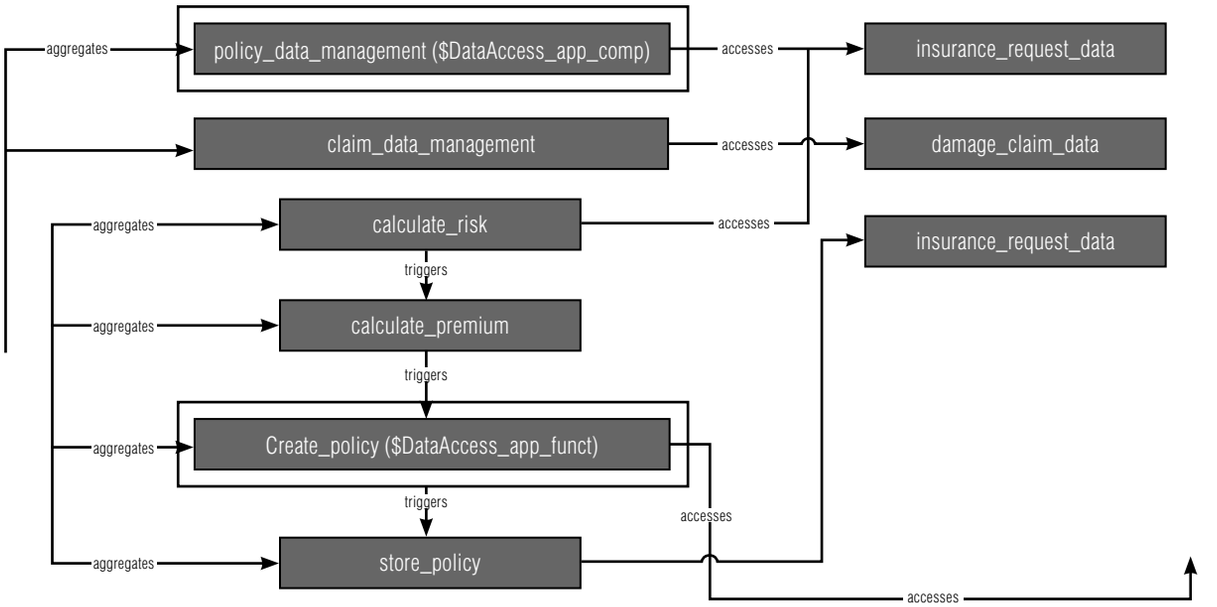

Рис. 4. Кейс ArchiSurance. Найденные сущности, противоречащие заданному условию (в рамке)

В рассматриваемом примере MIT Alloy Analyzer находит контрпример для заданного суждения и присваивает отношению Сколема *app_comp* типа ApplicationComponent имя «$DataAccess_app_comp» для существующей сущности «PolicyDa-taManagement», а отношению *app_funct* типа ApplicationFunction имя «$DataAccess_app_funct» для существующей сущности «CreatePolicy». Таким образом, в нашем примере найдены две сущности, не соответствующие заданным формальным требованиям (*рисунки 3* и *4*).

После отрицательного результата верификации возможны два метода решения проблемы: изменение текущей архитектуры или формальных требований. Если же формальные требования определены стейкхолдерами и, следовательно, не подлежат корректировке, то архитектура предприятия нуждается в реконструкции до тех пор, пока верификация не даст положительный результат.

8. Интеграция подхода в визуальный редактор Archi

Одной из основных целей данной работы является разработка программного инструмента, который позволил бы автоматизировать часть процесса верификации моделей ArchiMate и сделать подход проверки моделей в области бизнес-моделирования более доступным. В качестве программной платформы выбран популярный редактор Archi. Это бесплатный кросплатформенный инструмент с открытым исходным кодом для визуального моделирования и дизайна моделей ArchiMate, разработанный на платформе Eclipse EMF и расширяемый с помощью пользовательских плагинов.

В данной работе разработан плагин к визуальному редактору Archi, поддерживающий автоматическое преобразование представления архитектуры предприятия в язык Alloy (*рисунок 5*). Плагин

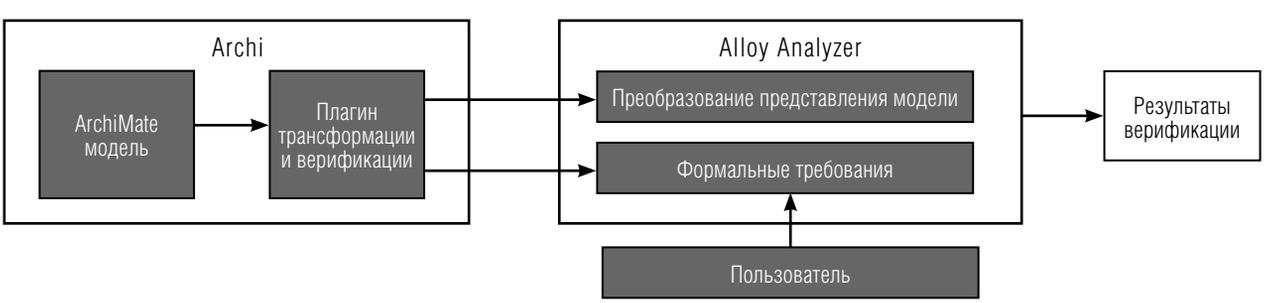

Рис. 5. Схема инструментария верификации

написан на языке Java. Дизайн решения содержит три основных класса: логика синтаксического анализа и преобразования модели, реализация пользовательского интерфейса и вспомогательные методы преобразования строк. Логика преобразования инкапсулирована в классе AlloyExporter, а публичный метод transformModelпринимает объект типа IArchimateModel и формирует временный файл CaseExample.als, в который записывается преобразованная модель. Интерфейс ввода Alloy-команд написан с использованием библиотеки Swing. Объем кода за счет повторного использования библиотек системы Alloy составляет 546 строк (*https://github.com/nik-ponomarev/archimate2alloy*).

После проектирования многоуровневой архитектуры предприятия ArchiSurance вызов диалогового окна верификации в Archi осуществляется посредством выбора «File → Export → Model to Alloy Format» в контекстном меню.

Далее появляется интерфейс ввода формальных требований к архитектуре на языке Alloy, а также команды запуска верификации, где нужно определить охват поиска. На данном этапе реализации возможна проверка лишь одного высказывания за один запуск.

Опция «*Use full scope*» говорит о том, что в верификации должны будут участвовать все сущности модели, каждый в одном экземпляре. Если же пользователю необходимо задать свой тип охвата, то для этого есть опция «*Custom scope*». Кнопка «*Find solution*» запускает механизм верификации.

В случае отрицательного результата верификации, при наличии сколемизированных структур, сущности, не удовлетворяющие условию запроса, будут окрашены в красный цвет. При отсутствии данных переменных в красный цвет будут окрашены все элементы модели, которые будут сгенерированы Alloy Analyzer в качестве контрпримера. Если же не найдено никаких контрпримеров, то исходная модель удовлетворяет формальным требованиям. В этом случае будет выведено соответствующее окно с сообщением «*No example/counter-example found*».

В нашем примере на заданный вопрос плагин нашел две компоненты уровня приложений, неудовлетворяющие условию ограничения, и окрасил их в красный цвет. Это компоненты PolicyDataManagement и CreatePolicy, которые в текущей архитектуре пользуются данными клиентов, однако не имеют доступа к их получению из сервисов баз данных технологического уровня. В связи с этим необходимо пересмотреть логику текущей архитектуры.

Предложенный в работе подход к проверке моделей позволяет определить ошибки, не просто связанные с неправильным отображением спецификации, а именно ошибки на уровне логики проектирования архитектуры предприятия и поведения бизнес-процессов. Следует отметить, что с ростом размера моделей проводить подобную верификацию вручную не представляется возможным, что еще раз подтверждает практическую ценность данного инструмента.

Заключение

В настоящей работе исследуется проблема автоматической верификации моделей архитектуры предприятия на языке ArhiMate. В качестве инструментария метода проверки моделей (model checking) использовался инструмент реляционной логики и система моделирования MIT Alloy Analyzer.

Идея работы состоит в разработке решения, тесно интегрированного с активно используемым на практике инструментом моделирования Archi, которое позволяло бы извлекать все элементы из представления архитектуры, а затем полностью автоматическим преобразованием создавать формальную модель на языке Alloy и проводить верификацию посредством MIT Alloy Analyzer. Верификация проводится на основе спроектированной метамодели спецификации ArchiMate на языке Alloy, которая описывает основные сущности, связи между ними и другие ограничения языка. Спецификация формальных требований в виде фактов сигнатур (fact), импликаций на проверку (assertion), предикатов (predicate) и функций (function) может вводиться пользователем в отдельном меню, указав опции ограничения размеров поиска. В работе также обсуждаются методы преобразования модели и правила описания требований, применяемые в реализованном программном обеспечении.

Наконец, прикладной характер метода представлен на примере кейса компании ArchiSurance – верификации его обобщенного многослойного представления архитектуры предприятия. Также в работе описан механизм отображения результатов верификации в инструменте моделирования Archi, что позволяет лучше отобразить принципы функционирования предприятия.

На основании проделанной работы можно сделать вывод о применимости механизма логической проверки для моделей архитектуры предприятия ArchiMate. Наиболее важным этот метод представляется для моделей с большим количеством эле-

ментов и связей между ними, верификация которых вручную не представляется возможной. Внедрение формального описания архитектуры предприятия, ее бизнес-процессов, и требований позволит построить системы управления качеством компании, решить проблему построения эффективной структуры управления, оптимизировать деятельность на основе ключевых показателей.

В перспективе, данная работа будет совершенствоваться в направлении реализации проверки на моде-лях с помощью LTL/CTL-логик. Развитие предложенных в этой работе подходов позволит использовать формальные методы анализа для широкого класса моделей, в том числе и моделей коммуникации автономных агентов в рамках методологии DEMO. ■

Литература

1. James G.A., Handler R.A., Lapkin A., Gall N. Gartner enterprise architecture framework: Evolution 2005. October 25, 2005. Gartner ID: G00130855.
2. Кларк Э.М., Грамберг О., Пелед Д. Верификация моделей программ: Model checking / Пер. с англ. – М.: Изд. Московского центра непрерывного математического образования, 2002.
3. Коротков А. Архитектура предприятия. Как заставить ИТ работать на вашу компанию. 2013. [Электронный ресурс]: http://andrey-korotkov.ru/wp-content/uploads/2013/02/andrey-korotkov.ru_Enterprise_architecture.pdf (дата обращения 16.03.2017).
4. Clark T., Sammut P., Willans J. Applied meta-modeling: A foundation for language driven development. Ceteva, 2008.
5. Jonkers H., Band I., Quartel D. The ArchiSurance case study / White paper. The Open Group, 2012.
6. Archimate 2.1 Specificattion. Open Group Standard. Zaltbommel: Van Haren Publishing, 2013. [Электронный ресурс]: https://www.vanharen.net/Samplefiles/9789401800037SMPL.pdf (дата обращения 16.03.2017).
7. Jackson D. Software abstractions: Logic, language and analysis. MIT Press, 2006.
8. Кудрявцев Д.В., Арзуманян М.Ю., Григорьев Л.Ю. Технологии бизнес-инжиниринга. – СПб: Изд. Политехнического университета, 2014.
9. Карпов Ю.Г. Model checking. Верификация параллельных и распределенных программных систем. СПб: БХВ-Петербург, 2009.
10. Szwed P. Verification of ArchiMate behavioral elements by model checking // 14th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM 2015). Warsaw, Poland, 24-26 September 2015. P. 132–144.
11. Lano K. The B language and method: A guide to practical formal development. Springer Science & Business Media, 2012.
12. Warmer J.B., Kleppe A.G. The object constraint language: Precise modeling with UML. Addison-Wesley, 1998.
13. Fitzgerald J.S., Larsen P.G., Verhoef M. Vienna development method / Wiley encyclopedia of computer science and engineering. John Wiley & Sons, 2008.
14. Spivey J.M., Abrial J.R. The Z notation. Hemel Hempstead: Prentice Hall, 1992.

Analysis of the consistency of enterprise architecture models using formal verification methods

Eduard A. Babkin
Professor, Department of Information Systems and Technology
National Research University Higher School of Economics
Address: 25/12, Bolshaya Pecherskaya Street, Nizhny Novgorod, 603155, Russian Federation
E-mail: eababkin@hse.ru

Nikita O. Ponomarev
Student, Business Informatics MSc Program
National Research University Higher School of Economics;
Software Engineer, Intel Corporation
Address: 25/12, Bolshaya Pecherskaya Street, Nizhny Novgorod, 603155, Russian Federation
E-mail: nik4nikita@gmail.com

[6] The research was carried out with financial support of Russian Fund of Basic Research No. 16-06-00184 A "Development and investigation models of online-discussion based on materials of political news"

Abstract

Enterprise architecture design is a complex process which makes it possible to synchronize the capabilities and needs of business and information technologies (IT). It can be achieved by clarifying the understanding and formalization of the business processes and the interaction of the elements of the system through their formal description. The large number of interacting business processes and enterprise architecture entities raises the question of verifying their correctness. Therefore, it is necessary to formalize the requirements for architecture and be able to automatically verify them.

In this paper, we propose a method for detecting logical contradictions in enterprise architecture models based on a model checking approach adopted in the context of business modeling. As an enterprise architecture description language, we use the modern open and independent ArchiMate standard. Developed by The Open Group, the standard provides a general specification for business processes, organizational structures, information flows, IT-systems and the technical infrastructure description of the enterprise. As a verifier, the language and tools of the MIT Alloy Analyzer system were chosen; they facilitate analysis of model constraints in terms of relational logic by automatically generating structures that satisfy the requirements of a logical model.

In this paper, we propose to simplify and automate the process of specification and verification of enterprise architecture domain models using Archi - the visual editor for ArchiMate models. We have developed the editor plug-in which translates the enterprise architecture models into the language of the MIT Alloy Analyzer system and uses the meta-model of the ArchiMate specification as the basis for constructing specific domain models. The proposed method and software solutions have been tested using the ArciSurance case and their enterprise architecture model.

Key words: enterprise architecture, ArchiMate, Alloy Analyzer, verification, consistency analysis, formal methods.

Citation: Babkin E.A., Ponomarev N.O. (2017) Analysis of the consistency of enterprise architecture models using formal verification methods. *Business Informatics*, no. 3 (41), pp. 30–40.
DOI: 10.17323/1998-0663.2017.3.30.40.

References

1. James G.A., Handler R.A., Lapkin A., Gall N. (2005) *Gartner enterprise architecture framework: Evolution 2005.* October 25, 2005. Gartner ID: G00130855.
2. Clarke E., Grumberg O., Peled D. (1999) *Model checking.* MIT Press.
3. Korotkov A. (2013) *Arkhitektura predpriyatiya. Kak zastavit' IT rabotat' na vashu kompaniyu* [Enterprise architecture. How to force IT to work in favor of your company]. Available at: http://andrey-korotkov.ru/wp-content/uploads/2013/02/andrey-korotkov.ru_Enterprise_architecture.pdf (accessed 16 March 2017) (in Russian).
4. Clark T., Sammut P., Willans J. (2008) *Applied meta-modeling: A foundation for language driven development.* Ceteva.
5. Jonkers H., Band I., Quartel D. (2012) *The ArchiSurance case study. White paper.* The Open Group.
6. The Open Group (2013) *Archimate 2.1 Specificattion. Open Group Standard.* Zaltbommel: Van Haren Publishing. Available at: https://www.vanharen.net/Samplefiles/9789401800037SMPL.pdf (accessed 16 March 2017).
7. Jackson D. (2006) *Software abstractions: Logic, language and analysis.* MIT Press.
8. Kudryavtsev D.V., Arzumanyan M.Y., Grigoriev L.Y. (2014) *Tekhnologii biznes-inzhiniringa* [Technologies of business engineering]. St. Petersburg, Polytechnic University (in Russian).
9. Karpov Y.G. (2009) *Model checking. Verifikatsiya parallel'nykh i raspredelennykh programmnykh sistem* [Model checking. Verification of parallel and distributed program systems]. St. Petersburg, BHV-Petersburg (in Russian).
10. Szwed P. (2015) Verification of ArchiMate behavioral elements by model checking. Proceedings of the *14th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM 2015)*. Warsaw, Poland, 24-26 September 2015, pp. 132–144.
11. Lano K. (2012) *The B language and method: A guide to practical formal development.* Springer Science & Business Media.
12. Warmer J.B., Kleppe A.G. (1998) *The object constraint language: Precise modeling with UML.* Addison-Wesley.
13. Fitzgerald J.S., Larsen P.G., Verhoef M. (2008) Vienna development method. *Wiley encyclopedia of computer science and engineering.* John Wiley & Sons.
14. Spivey J.M., Abrial J.R. (1992) *The Z notation.* Hemel Hempstead: Prentice Hall.