Национальный исследовательский университет «Высшая школа экономики»
Факультет компьютерных наук

На правах рукописи

Алексей Александрович Мицюк

Исправление моделей процессов с сохранением их структуры на основе журналов событий

Резюме диссертации

на соискание ученой степени

кандидата компьютерных наук НИУ ВШЭ

Москва — 2019

Диссертационная работа выполнена в Национальном исследовательском университете «Высшая школа экономики».

Научный руководитель:

Ирина Александровна Ломазова
Профессор, доктор физико-математических наук,
Факультет компьютерных наук Национального исследовательского университета «Высшая школа экономики»

Диссертационный комитет:

Александр Константинович Петренко (председатель),
Профессор, доктор физико-математических наук,
Отдел технологий программирования Института системного программирования РАН

Джозеп Кармона (Josep Carmona),
Доцент, Ph.D.,
Департамент компьютерных наук Политехнического университета Каталонии

Марко Монтали (Marco Montali),
Доцент, Ph.D.,
Факультет компьютерных наук Свободного университета Больцано

Валерий Анатольевич Соколов,
Профессор, доктор физико-математических наук,
Кафедра теоретической информатики Ярославского государственного университета

Алессандро Спердути (Alessandro Sperduti),
Профессор, Ph.D.,
Департамент математики Падуанского университета

Тема диссертации

В настоящий момент невозможно представить себе жизнь без информационных систем. Процессы в различных прикладных областях (информационные технологии, банковское дело, здравоохранение, промышленное производство) выполняются при поддержке программных систем, которые хранят и обрабатывают данные, относящиеся к этим процессам. Благодаря этому в последние годы сформировалась концепция процессно-ориентированных информационных систем [1,2]. Системы этого типа строятся на базе моделей, описывающих природу какого-либо прикладного процесса.

Для разработки сложных программных и информационных систем могут применяться подходы программной инженерии, базирующиеся на использовании формальных моделей [3, 4]. В рамках таких подходов при проектировании структуры и процессов системы используется формальное моделирование, а затем тщательно проработанные проектные модели реализуются в виде программного кода. Тем не менее, крайне редко нормативная проектная модель точно реализуется в реальной системе. Более того, процессы подвержены эволюции и эрозии на протяжении жизненного цикла системы. Поэтому структура и поведение реальной системы, как правило, отличаются от её проектной модели.

Для владельца системы полезными являются актуальные, обновляемые модели, описывающие структуру и поведение системы в соответствии с реальным состоянием исполняющихся в ней процессов. Это обуславливает разработку многочисленных методов обратного инжиниринга, предназначенных для анализа информационных систем, восстановления моделей их структуры и поведения.

В частности, можно исследовать реальное поведение программной системы на основе анализа её *журналов событий*. Алгоритмы и методы для анализа такого типа разрабатываются в рамках исследовательской и технологической области под названием «извлечение и анализ процессов» (Process mining) [5]. Модель реальной системы может быть автоматически *синтезирована* по журналу событий. Более того, техники *проверки согласованности* позволяют процессным инженерам диагностировать расхождения в наблюдаемом (журналы событий) и модельном (модели процессов) поведении.

Информация о согласованности может использоваться для *улучшения* или *обогащения* моделей процессов. Например, модель может быть *исправлена* с использованием журнала событий [6]. В этом случае конструируется новая модель процесса, базирующаяся на данной исходной модели, но лучше согласованная с данным журналом событий. Использование техник исправ-

ления моделей процессов позволяет сохранять модель процесса в актуальном состоянии. Данная диссертация посвящена разработке таких алгоритмов.

Цель исправления модели процесса — улучшить её качество[1]. При этом, как правило, имеются дополнительные ограничения. При исправлении в модель должно вноситься как можно меньше изменений, что обеспечивает сохранение её структуры. Наличие этого дополнительного ограничения отличает задачу исправления от задачи автоматического построения модели процесса, когда цель состоит в том, чтобы синтезировать по данному журналу событий модель процесса, удовлетворяющую заданным характеристикам согласованности. Таким образом, методы исправления применяются в тех случаях, когда ценные качества исходной модели процесса могут быть утеряны, если полностью перестроить модель с нуля.

Задача может быть проиллюстрирована с помощью следующего примера. На рисунке 1 показана модель процесса, которая не соответствует[2] наблюдаемому поведению системы. В частности, соответствие данной модели и журнала событий — 0,97 (где 1 — это идеальное соответствие).

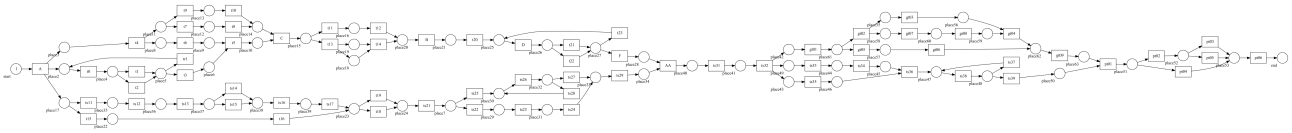

Рисунок 1: Заданная модель процесса

Можно применить один из алгоритмов автоматического построения и синтезировать новую модель с нуля так, чтобы она идеально соответствовала этому журналу событий. Это можно сделать, к примеру, с использованием индуктивного алгоритма синтеза (`Inductive miner`), если настроить алгоритм на нулевой уровень шума в журнале событий. На рисунке 2 показана модель, которая была синтезирована с применением этого алгоритма.

Эта модель идеально соответствует журналу событий. Она моделирует тот же самый процесс с тем же множеством действий, а потому содержит переходы с пометками из того же множества, что и исходная модель. Тем не менее, структура моделей сильно отличается. Заметим, что исходная модель почти идеально соответствовала журналу событий. Это означает, что степень несовместимости модели и журнала нельзя назвать серьёзной. В действительности модель, изображенная на рисунке 1, может быть исправлена

[1]Качество модели оценивается в соответствии с выбранным критерием качества. Например, модель должна быть согласована с заданным журналом событий.

[2]*Соответствие* — один из многих применяемых критериев согласованности. Модель идеально соответствует журналу событий, если она способна воспроизвести все варианты поведения, записанные в данном журнале. Уровень соответствия показывает, какая доля поведения из журнала событий может быть воспроизведена моделью.

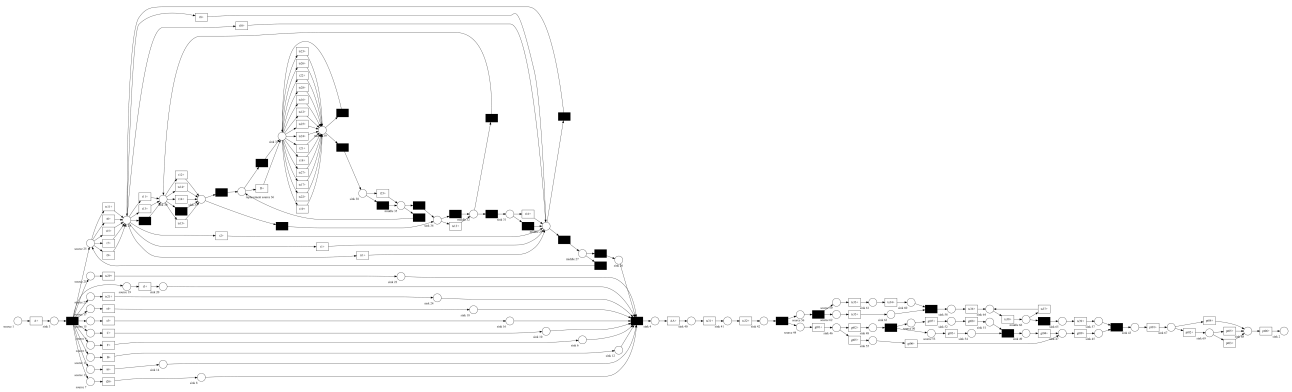

Рисунок 2: Модель, синтезированная с нуля индуктивным алгоритмом синтеза

путём перестановки всего двух её элементов (переходов). Подобные несовместимости далеко не всегда требуют полного перестроения модели. Нередко они могут быть исправлены без внесения существенных изменений в модель процесса.

Именно в таких случаях актуальным является применение алгоритмов, сохраняющих структуру модели. Алгоритмы исправления моделей такого типа довольно сложны, так как от них требуется сохранение баланса между необходимостью получить модель, более согласованную с журналом событий, и устремлением сохранить структуру исходной модели неизменной. Цель данной диссертации как раз и состоит в том, чтобы предложить новые сохраняющие структуру методы исправления моделей процессов.

В данной диссертации решается **задача** *исправления модели процесса*, которая определяется следующим образом. Исходная модель процесса N описывает поведение системы. Журнал событий L этой системы не согласован[3] с моделью процесса N в соответствии с некоторыми наперед заданными критериями согласованности. Другими словами, модель N не в полной мере отражает наблюдаемое поведение системы. Обобщенная задача исправления модели процесса состоит в том, чтобы найти модель N^r (*исправленная модель процесса*) такую, чтобы она была согласована с журналом событий L в соответствии с упомянутыми выше критериями, и, кроме того, модели N^r и N схожи настолько, насколько это возможно. Для выполнения последнего условия процедура исправления не должна существенным образом изменять модель процесса.

[3]Заметим, что в данной диссертации применяются две формы записи с одинаковым значением: *модель согласована с журналом событий* и *журнал событий согласован с моделью*.

Основные результаты диссертации

Основные результаты данной диссертации, полученные её автором и выносимые на защиту:

I. Предложен *новый* модульный подход (схема) исправления моделей процессов на базе журналов событий. Подход основан на применении декомпозиции моделей процессов. В качестве моделей рассматриваются сети потоков работ. Модульная схема может включать разнообразные алгоритмы декомпозиции модели и исправления под-сетей. Сформулированы достаточные условия эффективности подхода.

II. Предложены *новые* алгоритмы для выполнения локального и нелокального исправления моделей процессов на базе журналов событий. Данные алгоритмы реализуют обобщённую модульную схему и используют принцип *разделяй и властвуй*.

III. Предложены *новые* алгоритмы генерации журналов событий в результате симуляции моделей процессов.

IV. Прототипы алгоритмов исправления моделей процессов экспериментально оценены с использованием журналов событий, которые были подготовлены в результате применения алгоритмов симуляции моделей процессов.

Публикации и апробация работы

Результаты данной диссертации были опубликованы в международных рецензируемых журналах и трудах конференций. Для каждой публикации приводится её формальный статус и вклад автора диссертации.

Публикации

Основные результаты данной диссертации опубликованы в следующий статьях, которые приводятся с разбиением на три группы на основе из формального статуса в соответствии с правилами Диссертационного совета по компьютерным наукам НИУ ВШЭ: «*Публикациями повышенного уровня* являются статьи в изданиях, входящих в системы цитирования Web of Science (квартили Q1, Q2) или Scopus (квартили Q1, Q2), а также в рецензируемых трудах конференций, входящих в рейтинг CORE (ранги A и A*). *Публикациями стандартного уровня* являются статьи в изданиях, входящих в список

рекомендованных журналов НИУ ВШЭ или в системы цитирования Web of Science (квартили Q3, Q4) или Scopus (квартили Q3, Q4), а также в рецензируемых трудах конференций, входящих в рейтинг CORE (ранг B)»[4].

Публикации повышенного уровня

1. *Mitsyuk A. A.*, Shugurov I. S., Kalenkova A. A., van der Aalst W. M. P. Generating Event Logs for High-Level Process Models // Simulation Modelling Practice and Theory. 2017. Vol. 74. P. 1–16. *(Статья в журнале, проиндексированная SCOPUS и WoS, Q2; вклад автора:* 0,5*)* [7]

Публикации стандартного уровня

1. *Mitsyuk A. A.*, Lomazova I. A., van der Aalst W. M. P. Using Event Logs for Local Correction of Process Models // Automatic Control and Computer Sciences. 2017. Vol. 51. No. 7. P. 709–723. *(Статья в журнале, проиндексированная SCOPUS и WoS, Q3; вклад автора:* 0,9*)* [8]

   Перевод с русского статьи: *Мицюк А. А.*, Ломазова И. А., ван дер Аалст В. М. П. Использование журналов событий для локальной корректировки моделей процессов // Моделирование и анализ информационных систем. 2017. Т. 24. № 4. С. 459–480. *(Статья в журнале из рекомендованного списка НИУ ВШЭ; вклад автора:* 0,9*)* [9]

2. Shugurov I. S., *Mitsyuk A. A.* Iskra: A Tool for Process Model Repair // Proceedings of the Institute for System Programming. 2015. Vol. 27. No. 3. P. 237–254. *(Статья в журнале из рекомендованного списка НИУ ВШЭ; вклад автора:* 0,6*)* [10]

3. *Mitsyuk A. A.*, Shugurov I. S. On Process Model Synthesis Based on Event Logs with Noise // Automatic Control and Computer Sciences. 2016. Vol. 50. No. 7. P. 460–470. *(Статья в журнале, проиндексированная SCOPUS и WoS, Q4; вклад автора:* 0,8*)* [11]

   Перевод с русского статьи: *Мицюк А. А.*, Шугуров И. С. Синтез моделей процессов по журналам событий с шумом // Моделирование и анализ информационных систем. 2014. Т. 21. № 4. С. 181–198. *(Статья в журнале из рекомендованного списка НИУ ВШЭ; вклад автора:* 0,8*)* [12]

[4] `https://www.hse.ru/en/science/disscoun/council_computerscience/`

Прочие публикации

1. *Mitsyuk A. A.* Non-Local Correction of Process Models Using Event Logs, in: Proceedings of the 2017 Ivannikov ISPRAS Open Conference. Los Alamitos : IEEE Computer Society , 2018. Ch. 2. P. 6–11. *(Статья в сборнике трудов конференции, проиндексированная SCOPUS; вклад автора:* 1*)* [13]

2. *Mitsyuk A. A.*, Lomazova I. A., Shugurov I. S., van der Aalst W. M. P. Process Model Repair by Detecting Unfitting Fragments, in: Supplementary Proceedings of the 6th International Conference on Analysis of Images, Social Networks and Texts (AIST-SUP 2017), Moscow, Russia, July 27-29, 2017. CEUR-WS.org Vol. 1975 Aachen, 2017. Ch. 32. P. 301–313. *(Статья в сборнике трудов семинара, проиндексированная SCOPUS; вклад автора:* 0,7*)* [14]

3. Shugurov I. S., *Mitsyuk A. A.* Generation of a Set of Event Logs with Noise, in: Proceedings of the 8th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2014). Moscow: Institute for System Programming RAS, 2014. P. 88–95. *(Статья в сборнике трудов конференции; вклад автора:* 0,7*)* [15]

Другие публикации автора диссертации

Список публикаций автора диссертации не ограничивается статьями, перечисленными выше. Ниже приводятся статьи автора по теме анализа и автоматического синтеза моделей процессов, но не содержащие результатов данной диссертации. Как и выше, все статьи разбиты на три категории.

Публикации повышенного уровня

1. Rubin V., *Mitsyuk A. A.*, Lomazova I. A., van der Aalst W. M. P. Process Mining Can Be Applied to Software Too!, in: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. NY: ACM, 2014. Ch. 57. P. 1–8. [16]

Публикации стандартного уровня

1. Nesterov R. A., *Mitsyuk A. A.*, Lomazova I. A. Simulating Behavior of Multi-Agent Systems with Acyclic Interactions of Agents // Proceedings of the Institute for System Programming. 2018. Vol. 30. No. 3. P. 285–302. [17]

2. Shugurov I. S., *Mitsyuk A. A.* Applying MapReduce to Conformance Checking // Proceedings of the Institute for System Programming. 2016. Vol. 28. No. 3. P. 103–122. [18]

3. Nikitina N., *Mitsyuk A. A.* Carassius: A Simple Process Model Editor // Proceedings of the Institute for System Programming. 2015. Vol. 27. No. 3. P. 219–236. [19]

4. *Mitsyuk A. A.*, Kalenkova A. A., Shershakov S. A., van der Aalst W. M. P. Using process mining for the analysis of an e-trade system: A case study // Business Informatics. 2014. Vol. 29. No. 3. P. 15–27. [20]

Прочие публикации

1. *Mitsyuk A. A.*, Kotylev Y. V. Layered Layouts for Software Systems Visualization Using Nested Petri Nets, in: Tools and Methods of Program Analysis: 4th International Conference, TMPA 2017, Moscow, Russia, March 3-4, 2017, Revised Selected Papers. Communications in Computer and Information Science Vol. 779. Springer International Publishing, 2018. Ch. 11. P. 127–138. [21]

Конференции и семинары

Результаты данной диссертации были **представлены и обсуждены** в рамках следующих конференций и семинаров:

1. Открытая конференция Института системного программирования РАН им В. П. Иванникова — 2017. Москва, РАН, 30.11.2017. Доклад: Нелокальная корректировка моделей процессов с использованием журналов событий (Non-Local Correction of Process Models Using Event Logs).

2. Семинар московской секции ACM SIGMOD. Москва, ВМК МГУ, 26.10.2017. Доклад: Корректировка моделей процессов по логам событий (Process Model Correction based on Event Logs).

3. Семинар лаборатории ПОИС. Москва, ФКН НИУ ВШЭ, 09.10.2017. Доклад: Использование журналов событий для локальной корректировки моделей процессов (Using Event Logs for Local Correction of Process Models).

4. 6-я международная конференция «Анализ изображений, социальных сетей и текстов» (AIST 2017). Москва, Политехнический университет, 27-29.07.2017. Постерный доклад: Исправление моделей процессов путём

выявления фрагментов с несоответствиями (Process Model Repair by Detecting Unfitting Fragments).

5. Семинар лаборатории ПОИС. Москва, ФКН НИУ ВШЭ, 17.10.2016. Доклад: Модульное исправление моделей процессов (Modular Process Model Repair).

6. Весенне-летний коллоквиум молодых исследователей по программной инженерии (SYRCoSE 2015). Самара, ПГУТИ, 28-30.05.2015. Доклад: Iskra: инструмент исправления моделей процессов (Iskra: A Tool for Process Model Repair).

7. Семинар «Процессно-ориентированные информационные системы». Московская область, Вороново, 28-29.11.2015. Доклад: Об исправлении моделей процессов (On Process Model Repair).

8. Семинар лаборатории ПОИС. Москва, ФКН НИУ ВШЭ, 12.10.2015. Доклад: Плохие и хорошие модели бизнес-процессов (Bad and Good Business Process Models).

9. Весенне-летний коллоквиум молодых исследователей по программной инженерии (SYRCoSE 2014). Санкт-Петербург, СПбГПУ им Петра Великого, 29-31.05.2014. Доклад: Генерация набора журналов событий с шумом (Generation of a Set of Event Logs with Noise).

Содержание диссертации

Данная диссертация состоит из четырёх основных частей, введения и заключения.

Во **введении** обсуждается актуальность темы диссертации, неформально описывается главная задача, решаемая в работе. Кроме того, введение вмещает необходимые формальные секции, а также краткое содержание всей диссертации.

В **первой части** содержится вводная информация, приводятся необходимые далее базовые определения и обзор литературы.

Во-первых, определяются сети Петри. Этот язык используется для моделирования процессов в данной диссертации. Сеть Петри — это направленный двудольный граф с узлами двух типов. Переходы моделируют действия процесса, а с помощью позиций задаётся порядок выполнения действий. В

частности, сети Петри позволяют моделировать основные конструкции процесса: последовательность действий, выбор, параллелизм, цикл. Узлы в сети Петри соединяются между собой дугами.

Текущее состояние сети Петри задаётся так называемой разметкой. Разметка — это распределение маркеров по позициям сети. Переход в сети Петри может сработать, если имеется достаточное количество маркеров в каждой из его входных позиций. При срабатывании переход потребляет маркеры в своих входных позициях и производит маркеры в своих выходных позициях. Таким образом, срабатывание перехода изменяет разметку сети.

В данной диссертации используются сети потоков работ, которые являются особым подклассом сетей Петри. Сеть потоков работ отличается тем, что имеет выделенные начальную (исток) и конечную (сток) позиции. Начальная разметка сети потоков работ содержит единственный маркер в *истоке*, тогда как в конечной разметке все позиции пусты, кроме позиции-*стока*, которая содержит единственный маркер. Все возможные исполнения такой сети начинаются с начальной разметки и заканчиваются в конечной разметке. Пример сети потоков работ показан на рисунке 3. Начальная разметка показана в виде чёрного маркера в позиции-истоке (source).

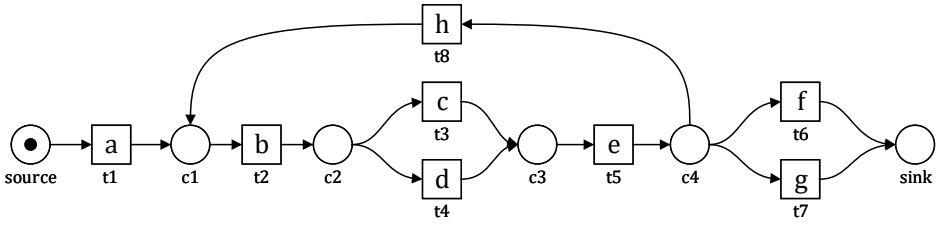

Рисунок 3: Простая сеть потоков работ

Во-вторых, определяются журналы событий.

Пусть $A \subseteq \mathcal{U}_A$ — это множество *действий* процесса. Они используются для задания меток переходов в модели. Будем предполагать, что *событие* в журнале — это имя действия, то есть никаких дополнительных атрибутов события не имеют. Таким образом, *событие* представляет *действие* процесса в *трассе журнала событий*.

Определим *трассу* σ как конечную последовательность действий из A, то есть $\sigma \in A^*$. *Журналом событий* (или просто *журналом*) L будем называть конечное мультимножество трасс, то есть $L \in \mathcal{B}(A^*)$. Например, $L = \left[\langle a,b,c,d,e\rangle^3, \langle a,b,a,d,e\rangle^5, \langle a,d,c,b,e\rangle\right]$ — это журнал событий с действиями из того же множества, что и модель, изображённая на рисунке 3.

В-третьих, описываются три составляющие дисциплины анализа процессов. *Анализ и извлечение моделей процессов* (*Process mining*) — это область исследований на стыке формальных методов и наук о данных [5]. В рамках данной дисциплины разрабатываются методы анализа процессов в информационных системах на базе журналов событий, записываемых в ходе жизненного цикла системы. Рассматривают алгоритмы трёх основных типов: для автоматического синтеза модели процесса по журналу событий (process discovery), для проверки согласованности модели и журнала событий (conformance checking), для усовершенствования или улучшения модели процесса (process enhancement). Алгоритмы всех трёх типов используются в данной диссертации.

Заметим, что среди алгоритмов третьего типа имеются как предназначенные для усовершенствования моделей процессов, так и процессов как таковых. Последние предназначаются для улучшения реальных процессов в соответствии с заданными критериями эффективности/производительности. Как правило, такие методы менее формальны и в большей степени ориентированы на применение в коммерческой деятельности. Методы и техники усовершенствования моделей процессов обычно более формальны. Алгоритмы *исправления моделей процессов* (или просто *исправления моделей*) относятся к этой последней категории методов [5].

Далее в данной части рассмотрены четыре основных численных критерия качества модели процесса: соответствие (fitness), точность (precision), обобщённость (generalization), простота (simplicity). Эти критерии используются повсеместно в ходе анализа моделей процессов [5].

Первые два применяются для проверки согласованности модели и журнала событий. *Соответствие* показывает в какой степени трассы журнала событий могут быть воспроизведены моделью. *Точность* определяет способна ли модель производить варианты поведения, не представленные в журнале событий. Обобщённость и простота характеризуют модель саму по себе. Обобщённость показывает уровень абстрактности модели, тогда как простота отражает её компактность.

В данной диссертации рассматривается задача исправления модели процесса с точки зрения её соответствия журналу событий. При оценке исправленных моделей также вычисляется насколько точно они отражают соответствующие журналы событий. В данной части работы детально описываются конкретные методы измерения соответствия и точности, которые используются далее. В диссертации для подсчёта согласованности модели и лога применяется подход, основанный на применении так называемых выравни-

ваний [22], который является общепринятым среди специалистов по анализу моделей процессов в настоящий момент.

Кроме того, в данной части описываются различные алгоритмы автоматического синтеза модели процесса [5]. Для использования в данной работе отобраны два таких алгоритма: индуктивный алгоритм `Inductive miner` [23] и алгоритм `ILP miner` [24], сводящий задачу синтеза к задаче целочисленного линейного программирования. Выбор этих двух алгоритмов обуславливается тем, что они оба, если используются с верно заданными настройками, обеспечивают синтез моделей, идеально соответствующих журналам событий, использованным в качестве исходных для синтеза.

В-четвёртых, в данной части приводится обзор методов исправления моделей процессов, предложенных ранее. Рассматриваются конкретные постановки задач, способы решения и их особенности.

Предлагаемый в данной диссертации подход также имеет ряд специфических свойств. Наиболее схожим из известных является способ, предложенный Д. Фаландом (D. Fahland) и В. ван дер Аалстом (W. van der Aalst) [25]. Тем не менее, отличием является тот факт, что в данной диссертации применяются техники декомпозиции модели, что позволяет затем заменить фрагменты модели с несоответствиями. Таким образом, исправленная модель отличается от исходной только в пределах заменённых (исправленных) фрагментов. Такие изменения несущественны, если несоответствия модели процесса и журнала событий локальны.

Особенно важно сохранить структуру модели в тех случаях, когда исходная модель была спроектирована экспертом, а значит хорошо читается человеком. Модели, которые строятся алгоритмами автоматического синтеза, нередко лишены такого важного достоинства. Кроме того, метод, предлагаемый в данной диссертации, лучше всего подходит для ситуаций, когда набор действий процесса стабилен и не должен изменяться в ходе исправления.

Во **второй части** представлен один из основных результатов данной диссертации, а именно описывается модульная техника исправления модели процесса.

В первом разделе этой части приводится формальная постановка задачи, решаемой в данной диссертации.

Пусть $A \subseteq \mathcal{U}_A$ — это множество действий процесса. Сеть потоков работ $N = (P, T, F, l)$ с пометками переходов из $\mathcal{U}_A$ — это исходная модель процесса. Также в данной сети могут быть невидимые переходы, помеченные с помощью специальной пометки τ. Таким образом, функция пометки имеет следующий вид: $l\colon T \to A \cup \{\tau\}$.

Журнал событий L — это мультимножество трасс в соответствии с определением из раздела 1.1.3. Имена событий выбираются из того же множества действий A, то есть $L \in \mathcal{B}(A^*)$. Заметим, что имена событий журнала L и пометки переходов модели N берутся из одного множества A, то есть $rng(l) = A$. Это означает, что среди действий процесса нет новых или устаревших, а метод исправления не добавляет новых переходов в модель процесса и не удаляет из неё старых, кроме, может быть, помеченных меткой τ.

Кроме того, предполагается, что все переходы модели N имеют уникальные пометки. Это правило очевидным образом не распространяется на невидимые переходы, помеченные меткой τ. Таким образом, в модели $N = (P, T, F, l)$, $\forall t_1, t_2 \in T$: если $l(t_1) = l(t_2)$, то $t_1 = t_2$. Заметим, что данное предположение потребуется на этапе декомпозиции модели процесса. В худшем случае, если все переходы модели имеют одинаковые пометки, модель не может быть декомпозирована с использованием алгоритмов, применяемых в данной диссертации.

Журнал событий L содержит *нормативное (корректное)* поведение, которое должно верно воспроизводиться моделью N, не в полной мере соответствующей этому журналу, то есть $fitness(L, N) < 1$.

Задача состоит в том, чтобы для заданных сети N и журнала событий L автоматически сконструировать сеть Петри N^r (*исправленная модель*) такую, что $fitness(L, N^r) = 1$. Строго говоря, L *идеально соответствует* N^r. Другими словами, исправленная модель N^r может проиграть все трассы из L. Это *строгое* условие, которое обязательно должно выполняться для успешности исправления модели.

Кроме того, имеются два *мягких* условия, выполнение которых желательно, но не всегда достижимо.

Во-первых, желательно, чтобы исправленная модель N^r была *точной* по отношению к журналу событий L. N^r не должна допускать «слишком много» вариантов исполнения, которые не представлены в L. По меньшей мере модель N^r должна быть насколько же точна, насколько точна была исходная модель N, то есть $precision(N^r) \approx precision(N)$. Другими словами, желательно, чтобы исправленная и исходная модели были *приблизительно* одинаково точными. В лучшем случае $precision(N^r) = precision(N)$.

Во-вторых, желательно, чтобы модели N и N^r имели *сходную* структуру. Алгоритм исправления тем лучше, чем меньше изменений он вносит в исправляемую модель. Именно поэтому в разделе 4.3, где описываются результаты экспериментальной оценки предложенных методов исправления, приводятся результаты подсчёта числа элементов модели, затронутых проце-

дурой исправления. Это число демонстрирует, насколько метод исправления изменяет исходную модель.

Далее во второй части представлены основные результаты диссертации. Во-первых, описана модульная схема предлагаемого подхода исправления модели процесса. Во-вторых, представлены алгоритмы, реализующие обобщённую модульную схему для конкретных вариантов исправления.

Модульная схема, предлагаемая в данной диссертации, основана на идее использования *заплаток*. Алгоритм отыскивает в исходной модели фрагменты с несоответствиями, которые затем заменяются фрагментами, соответствующими поведению, записанному в журнале событий. Подход является модульным. Это означает, что обобщённая схема конструируется из нескольких составляющих блоков. В качестве каждого из таких блоков могут выступать разнообразные конкретные алгоритмы. Таким образом, обобщённая схема может быть материализована в виде многих различных алгоритмов исправления. Для этого необходимо осуществить выбор конкретных алгоритмов, для использования в качестве процедурных аргументов схемы.

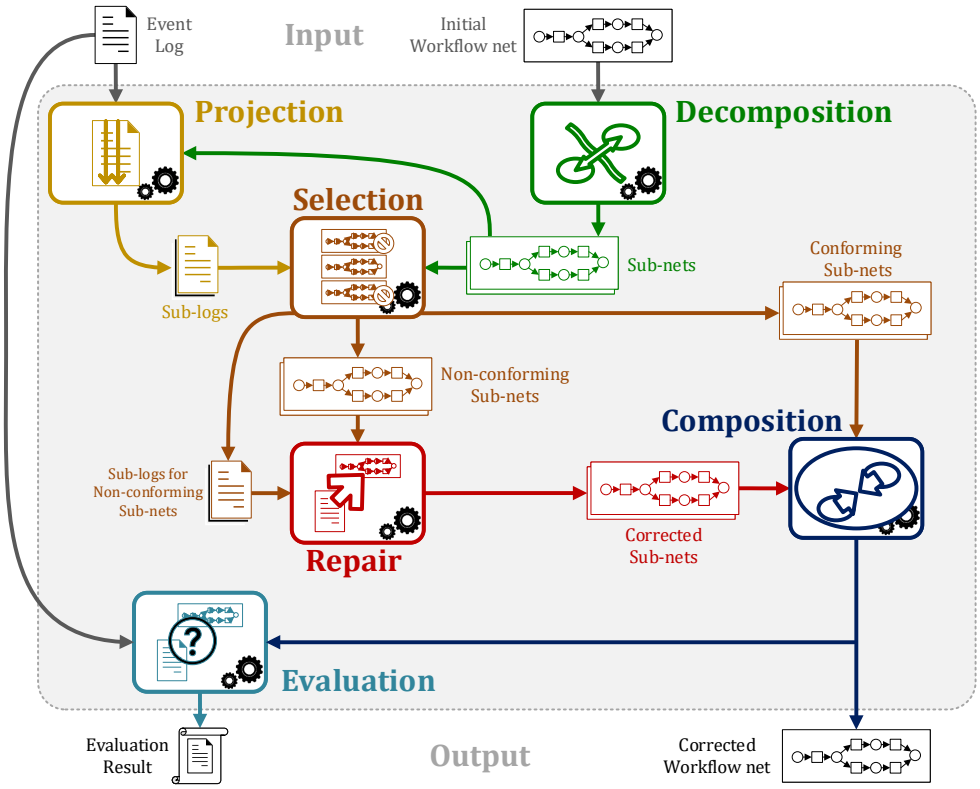

Рисунок 4: Модульная схема исправления

Каждый составляющий блок модульной схемы выполняет один из шагов исправления. Графическое изображение модульной схемы приводится на

рисунке 4. На этом рисунке составляющие строительные блоки показаны с помощью ярких цветных прямоугольников.

На вход схемы подаётся пара (L, N), где L — корректный (нормативный, доверенный) журнал событий, а N — исходная модель с несоответствиями. Модель N декомпозируется на k фрагментов N^1, $N^2, \ldots,$ N^k с использованием одного из алгоритмов декомпозиции модели. В блоке проецирования алгоритм разделяет журнал событий на k суб-журналов L^1, $L^2, \ldots,$ L^k, каждый из которых соответствует одному фрагменту модели (суб-сети). В блоке выбора помещается алгоритм проверки согласованности модели и журнала событий. Этот алгоритм применяется для вычисления *степени согласованности* каждой пары (L^i, N^i), где $i = 1, 2, \ldots, k$. Все фрагменты модели разделяются на два множества в соответствии с подсчитанным уровнем согласованности. Первое из этих множеств содержит фрагменты без несоответствий (*хорошие* суб-сети). Во второе множество попадают суб-сети, которые не полностью согласованы со своими суб-журналами (*плохие* фрагменты). Алгоритм, помещаемый в блок исправления, заменяет фрагменты с несоответствиями на исправленные суб-сети. Затем исправленная модель собирается из частей в блоке композиции, парном блоку декомпозиции. Результат исправления оценивается алгоритмами блока оценки, для чего также могут использоваться методы проверки согласованности или иные.

Определим данную обобщённую схему более формально.

Определение (Модульная схема исправления)**.** *Пусть $\mathcal{U}_A$ — это универсальное множество действий, а $\mathcal{U}_N$ — множество всех сетей потоков работ с пометками переходов из $\mathcal{U}_A$.*

Определим ***модульную схему исправления*** *как процедуру, принимающую на вход журнал событий $L \in \mathcal{B}(\mathcal{U}_A^*)$ и помеченную сеть потоков работ $N \in \mathcal{U}_N$, и возвращающую в качестве результата своей работы сеть Петри $N^r = ModularRepair(L, N)$, идеально соответствующую журналу событий L, то есть $fitness(L,N^r) = 1$.*

Модульная схема исправления имеет следующие процедурные параметры:

- $eval \in (\mathcal{B}(\mathcal{U}_A^*) \times \mathcal{U}_N) \to [0; 1]$ *— это функция оценки,*
- $repair \in (\mathcal{B}(\mathcal{U}_A^*) \times \mathcal{U}_N) \to \mathcal{U}_N$ *— это алгоритм исправления,*
- $split \in \mathcal{U}_N \to \mathcal{P}(\mathcal{U}_N)$ *— это алгоритм декомпозиции,*
- $compose \in \mathcal{P}(\mathcal{U}_N) \to \mathcal{U}_N$ *— это алгоритм композиции.*

Как правило, соответствие модели журналу событий рассматривается в качестве основного критерия качества модели. Действительно, модель,

которая не может воспроизвести поведение анализируемой системы, приносит мало пользы. Поэтому и в данной работе соответствие рассматривается как главный критерий. Сформулируем теперь условия исправления модели до идеального соответствия.

Определение (Исправление сети потоков работ до идеального соответствия)**.** *Пусть* $L \in \mathcal{B}(A^*)$ *— журнал событий, для которого* $A \subseteq \mathcal{U}_A$*, а* $N \in \mathcal{U}_N$ *— сеть потоков работ такая, что* $fitness(L, N) < 1$*. Пусть* $N' = ModularRepair(L, N)$ *— это модульная схема исправления. Будем говорить, что схема* $ModularRepair(L, N)$ ***исправляет до идеального соответствия****, если* $fitness(L, N') = 1$*.*

Модульный подход исправления гарантирует, что исправленная модель будет идеально соответствовать журналу событий, если в блоке исправления используется *идеальный алгоритм синтеза*[5], а в блоке декомпозиции применяется алгоритм *валидной декомпозиции*. По определению идеальный алгоритм синтеза для любого журнала событий синтезирует идеально ему соответствующую модель процесса. Алгоритм декомпозиции называется валидным, если с его помощью можно успешно декомпозировать на фрагменты и модель, и её разметку, а также имеется известный способ композиции, позволяющий однозначным образом собрать из этих фрагментов исходную модель.

Теорема (Условия исправления до идеального соответствия)**.** *Модульная схема исправления* $ModularRepair(L, N)$ *обеспечивает идеальное соответствие, если*

1. *split — это функция валидной декомпозиции;*
2. *repair — это идеальный алгоритм синтеза, то есть для любого журнала событий он синтезирует идеально соответствующую ему сеть Петри;*
3. *compose — это функция слияния переходов, которая объединяет все переходы с одинаковыми пометками.*

Доказательство этой теоремы приводится во второй части диссертации.

Затем описывается алгоритм, полученный в результате использования конкретных алгоритмов в качестве процедурных аргументов обобщённой модульной схемы.

[5]Такие алгоритмы всегда синтезируют модели с идеальным соответствием.

Зададим значения процедурных параметров модульной схемы исправления следующим образом:

- в качестве функции оценки $eval \in (\mathcal{B}(\mathcal{U}_A^*) \times \mathcal{U}_N) \to [0;1]$ используем **алгоритм проверки соответствия, использующий выравнивания** [22],
- в качестве функции исправления $repair \in \mathcal{B}(\mathcal{U}_A^*) \times \mathcal{U}_N) \to \mathcal{U}_N$ используем **индуктивный алгоритм синтеза** [23] или **алгоритм синтеза на базе решения задачи целочисленного линейного программирования** [24],
- в качестве функции декомпозиции $split \in \mathcal{U}_N \to \mathcal{P}(\mathcal{U}_N)$ используем **алгоритм максимальной декомпозиции**,
- в качестве функции композиции $compose \in \mathcal{P}(\mathcal{U}_N) \to \mathcal{U}_N$ используем алгоритм **слияния переходов** с одинаковыми пометками.

Такой выбор процедурных аргументов даёт **алгоритм декомпозированного исправления**. В данной части показывается, что этот алгоритм удовлетворяет первому, второму и третьему условиям исправления до идеального соответствия.

Для получения максимальной декомпозиции сети Петри в данной работе предлагается собственный алгоритм, который также описывается в части 2.

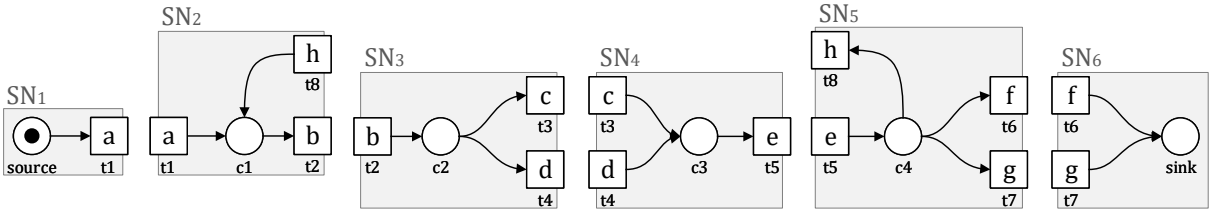

Рисунок 5: Максимальная декомпозиция модели, показанной на рисунке 3

На рисунке 5 демонстрируется максимальная декомпозиция модели процесса, показанной на рисунке 3. Сеть декомпозирована на шесть фрагментов: SN_1, SN_2, SN_3, SN_4, SN_5 и SN_6. Все переходы исходной модели видимы и имеют уникальные пометки. Поэтому переходы t_1 `(a)`, t_2 `(b)`, t_3 `(c)`, t_4 `(d)`, t_5 `(e)`, t_6 `(f)`, t_7 `(g)` и t_8 `(h)` являются граничными узлами. Внутренними являются позиции *source*, c_1, c_2, c_2, c_3, c_4, а также *sink*. Нетрудно заметить, что начальная разметка тоже была разбита тривиальным образом. Единственный маркер вместе с позицией *source* помещается в суб-сети SN_1. Ясно, что

представленные фрагменты могут быть композированы в сеть потоков работ, которая показана на рисунке 3. Для этого надо выполнить слияние переходов с одинаковыми пометками.

Назовём этот метод наивной техникой модульного исправления. На рисунке 6 показывается результат наивного исправления модели, изображённой на рисунке 3, для случая, когда журнал событий состоит всего из двух трасс: $\langle a, d, b, e, f \rangle$ и $\langle a, b, c, e, g \rangle$.

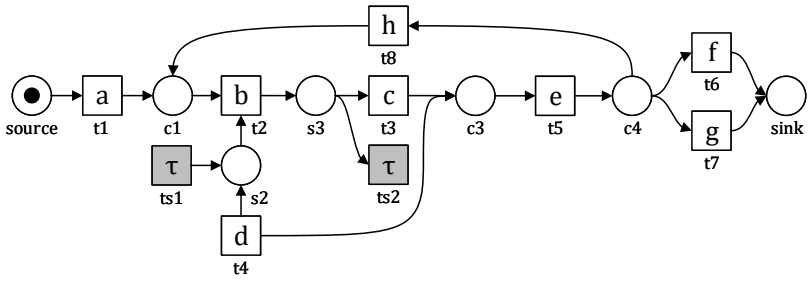

Рисунок 6: Модель процесса, показанная на рисунке 3, исправленная с помощью наивного метода

Результаты применения наивного метода могут быть улучшены, так как этот метод имеет тенденцию к построению неточных моделей. Дело в том, что данная процедура довольно существенным образом увеличивает количество возможных вариантов исполнения модели. Во второй части диссертации содержится подробное обсуждение причин данного снижения точности. Для устранения данного недостатка наивного метода в диссертации предлагается усовершенствованный алгоритм исправления.

Его ключевая идея состоит в том, чтобы увеличить каждый из фрагментов, требующих замены, таким образом, чтобы процедура исправления не затрагивала граничные переходы увеличенных фрагментов. На рисунке 7 показано, каким образом работает эта процедура увеличения. В данном случае к фрагменту SN_3 присоединяются соседние фрагменты SN_2 и SN_4.

Определим *увеличение* $\mathcal{W}(N^i)$ для фрагмента N^i как суб-сеть, получаемую путём присоединения к N^i всех его соседей, то есть $\mathcal{W}(N^i) = N^i \cup \mathcal{N}(N^i)$. Здесь $\mathcal{N}(N^i) = \{N^j \mid N^j \in D_N \wedge T_i \cap T_j \neq \emptyset\}$, а $i, j \in \{1, 2, \ldots, k\}$.

Процедура увеличения фрагментов применяется в случаях, когда декомпозиция модели содержит более одного фрагмента с несоответствиями. Она определяется следующим образом.

Определение (Процедура увеличения суб-сетей)**.** *Пусть* $D = \{N^1, \ldots, N^n\}$ *— это декомпозиция сети потоков работ* N*, а* $D = D_b \cup D_c$*, где* D_b *— это*

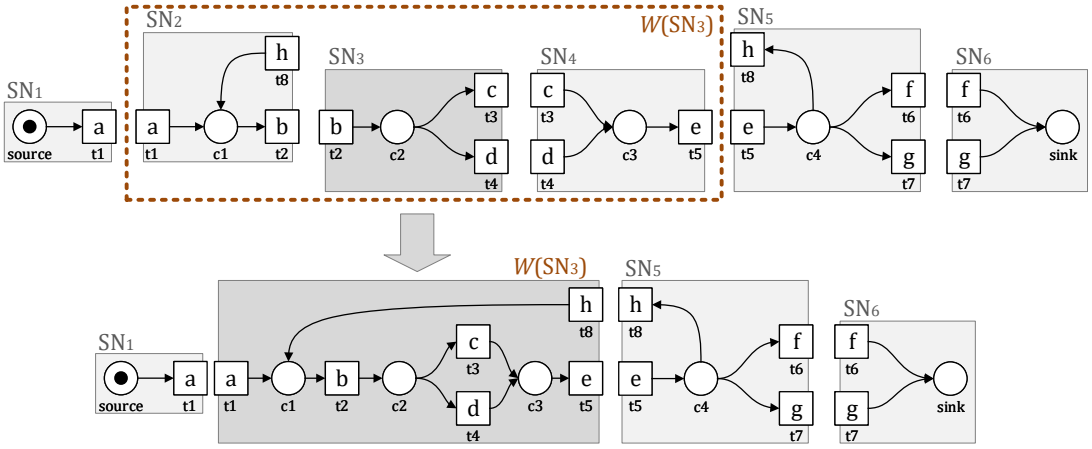

Рисунок 7: Суб-сеть увеличивается за счёт присоединения соседних фрагментов

множество фрагментов сети с несоответствиями, тогда как D_c *— это множество остальных фрагментов. Очевидно, что* $D_b \cap D_c = \emptyset$. ***Процедура увеличения суб-сетей*** *состоит из следующих шагов:*

1. *Каждый фрагмент из* D_b *увеличивается за счёт присоединения его соседей:* $\forall N^i \in D_b : D_w^i = \mathcal{W}(N^i)$, $D_n^i = \mathcal{N}(N^i)$; *здесь* $D_w = \bigcup_i D_w^i$ *— это множество всех увеличенных фрагментов, а* $D_n = \bigcup_i D_n^i$ *— множество всех фрагментов, которые были затронуты процедурой, где* $i = 1, 2, \ldots, n$; *заметим, что фрагменты могут иметь общих соседей;*

2. *Все фрагменты, затронутые процедурой, удаляются из исходной декомпозиции* $D_r = D_c \setminus D_n$;

3. *Увеличенные фрагменты включаются в новую декомпозицию:* $D_N^w = D_w \cup D_r$.

В данной диссертации показано, что процедура увеличения суб-сетей сохраняет валидность декомпозиции, а потому может быть включена в модульную схему исправления. Этот факт сформулирован в виде следующей теоремы из части 2.

Теорема (Увеличение суб-сетей сохраняет валидность декомпозиции)**.** *Пусть* $D_N = \{N^1, \ldots, N^n\} \in \mathcal{D}(N)$ *— это валидная декомпозиция сети* N. *Пусть* D_N^w *— это декомпозиция после увеличения суб-сетей, во время которого два фрагмента были объединены, то есть* $\exists i, j$ *такие, что* $1 \leq i < j \leq n$ *и* $D_N^w = \{N^i \cup N^j\} \cup D_N \setminus \{N^i, N^j\}$. *Тогда* D_N^w *— это также валидная декомпозиция, то есть* $D_N^w \in \mathcal{D}(N)$.

Доказательство данной теоремы содержится во второй части диссертации.

Назовём продвинутый алгоритм, использующий процедуру увеличения суб-сетей, усовершенствованным модульным алгоритмом исправления. Модели, исправленные с его применением (см. рисунок 8), обычно более точны, когда исправляются локальные несоответствия журналу событий.

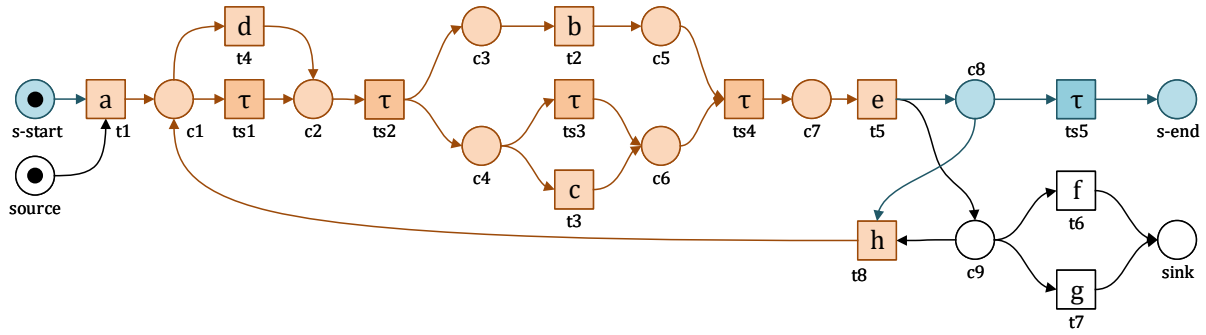

Рисунок 8: Модель процесса, исправленная с помощью усовершенствованного алгоритма

Локальными несоответствиями будем называть такие, которые могут быть исправлены путём изменения отношений потока в рамках одного фрагмента модели. Для исправления нелокального несоответствия в действительности требуется «*перенести*» переход из одного фрагмента в другой, что невозможно сделать с применением методов исправления локальных несоответствий. В таких случаях наивный и усовершенствованный алгоритмы исправления справятся с получением модели, способной воспроизвести все трассы журнала событий, но точность исправленной модели будет снижена. Два типа несоответствий графически показаны на рисунке 9.

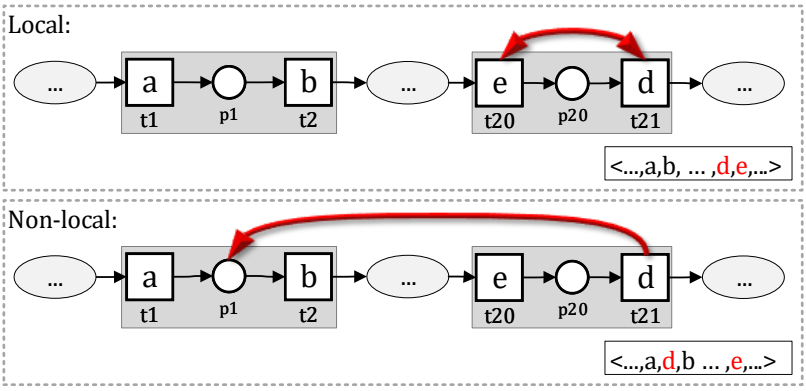

Рисунок 9: Два типа несоответствий

В данной части предлагается жадный метод исправления, также базирующийся на модульном подходе. Этот метод может удачно исправлять модель и в тех случаях, когда имеют место нелокальные несоответствия.

На вход данного алгоритма, как и ранее, подаются две составляющие: сеть потоков работ N, требующая исправления, а также журнал событий L, содержащий запись поведения системы. Алгоритм декомпозирует модель с помощью алгоритма максимальной декомпозиции. Затем выбирается субсеть N_b^w с наихудшим уровнем соответствия суб-журналу событий. После этого запускается итеративная процедура. На каждом её шаге выбранная субсеть увеличивается, то есть объединяется со своими соседями $N_b = \mathcal{W}(N^i)$, а затем этот увеличенный фрагмент заменяется моделью, синтезированной с помощью алгоритма автоматического синтеза, переданного в процедурный параметр $discover(L \restriction_{A(N_b)})$. Затем все фрагменты композируются в промежуточную сеть N'. Алгоритм повторяет итерации до тех пор, пока промежуточная сеть N' не окажется идеально соответствующей нормативному журналу событий L. Когда это условие выполняется, алгоритм заканчивает работу. Модель N' — это исправленная сеть Петри.

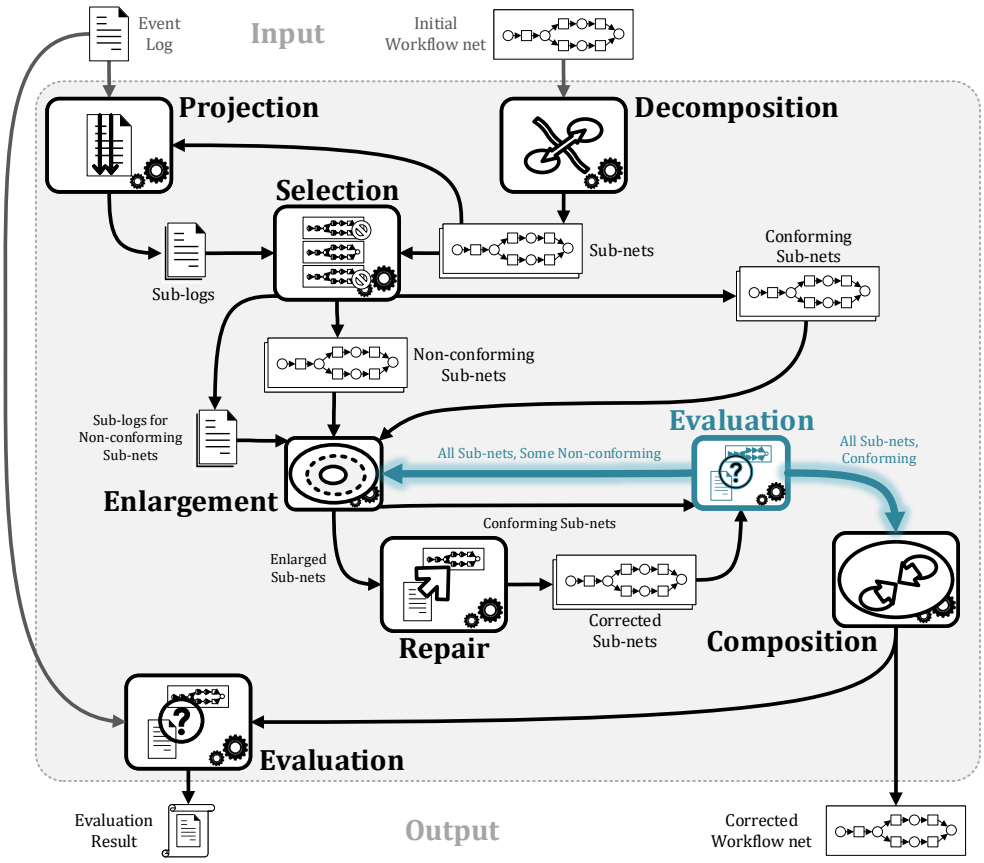

Рисунок 10: Схема исправления модели процесса для нелокального случая

Схема алгоритма изображена на рисунке 10. Светло-синим цветом выделена ключевая модификация модульной схемы исправления, которая лежит в основе жадного алгоритма.

Пример работы жадного алгоритма исправления показан на рисунке 11. Данная модель содержит два фрагмента с несоответствиями, которые выделены на рисунке красным цветом. В данном примере были переставлены местами пометки двух переходов из выделенных фрагментов. Жадный алгоритм начинает работу с одного из фрагментов и выполняет девять итераций, на каждой из которых увеличивается размер заменяемой суб-сети. В конечном итоге вся часть модели, выделенная на рисунке, будет заменена заново синтезированной моделью.

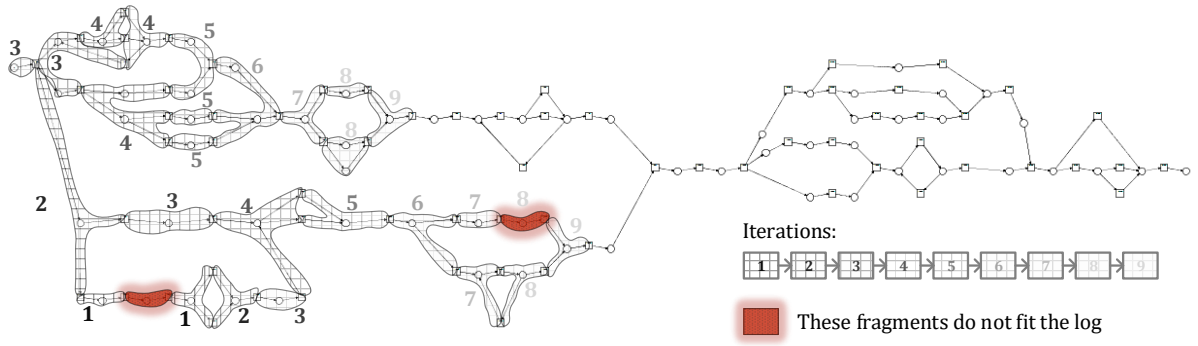

Рисунок 11: Пример работы жадного алгоритма исправления модели процесса

Во второй части диссертации предлагается семейство алгоритмов исправления модели процесса, которые базируются на общей модульной схеме. Основная идея, лежащая в основе всех этих алгоритмов, состоит в том, чтобы декомпозировать исходную модель, найти суб-сети с несоответствиями, а затем заменить их заново синтезированными моделям, которые бы соответствовали журналу событий.

Извлечение и анализ процессов — это практическая дисциплина, а потому новые алгоритмы, как правило, оцениваются в том числе и экспериментально с использованием примеров журналов событий. Крайней трудно подобрать реальные модели процессов и журналы событий информационных систем так, чтобы они обладали характеристиками, необходимыми для их использования при тестировании алгоритмов исправления. Поэтому в данной диссертации предлагаются методы генерации синтетических журналов событий, которые позволяют тонко настраивать характеристики получающихся журналов событий. Эти методы описываются в третьей части диссертации.

С использованием генератора журналов событий был подготовлен большой набор подходящих журналов событий, которые затем использованы

для экспериментальной оценки модульного подхода исправления. Результаты этой оценки приводятся в четвёртой части диссертации.

В **третьей части** диссертации описываются вспомогательные инструменты, которые использовались для генерации примеров журналов событий. В данной работе предлагается набор алгоритмов генерации журналов событий посредством симуляции сетей Петри и моделей BPMN 2.0. Предлагаемые алгоритмы реализованы в виде программных прототипов, которые также описываются в третьей части работы.

Данная часть состоит из двух основных разделов. В первом разделе описываются алгоритмы генерации журналов событий в результате симуляции сетей Петри. Аналогичный подход, применимый для моделей в нотации BPMN 2.0, описывается во втором разделе.

Предлагаемый метод симуляции сети Петри основывается на стандартных правилах срабатывания переходов в сетях Петри. Описываемый генератор обладает следующими основными возможностями:

- Пользователь имеет возможность задать количество журналов событий для генерации, а также количество трасс в каждом из журналов. Также требуется задать максимальное количество событий в трассе, что позволяет останавливать алгоритм в случае наличия бесконечных циклов в модели. Все журналы событий генерируются во время одного запуска программы. По умолчанию генерируется 5 журналов событий, каждый из которых содержит 10 трасс с не более чем 100 событиями.

- Если в модели присутствует недетерминированный выбор, пользователь имеет возможность задать правила выбора перехода для срабатывания. Инструмент может выбирать случайный переход с равной вероятностью. Кроме того, пользователь может задать приоритеты срабатывания, а также более и менее предпочтительные переходы для срабатывания с заданным уровнем вероятности.

- Кроме того, поддерживается симуляция моделей, имеющих временные характеристики. Пользователь может задать время срабатывания для каждого перехода, а также точность срабатывания путём определения границ отклонения времени срабатывания от эталонного. В таком случае каждому срабатыванию перехода в журнале событий будут соответствовать два события. Первое из них отмечает начало исполнения действия процесса. Второе событие отмечает окончание исполнения действия.

- Кроме трасс, точно отражающих варианты исполнения модели, инструмент способен подмешивать в журнал событий шум различного характера.

В данной части также описываются алгоритмы, предлагаемые для симуляции моделей в нотации BPMN 2.0. Поддерживаются модели, содержащие базовые элементы BPMN (задачи, шлюзы AND/XOR), объекты данных, потоки сообщений. Данные алгоритмы также позволяют задавать временные характеристики моделей и генерировать журналы событий с указанием времени выполнения действий процесса.

Кроме того, в данной части описывается тестирование и оценка алгоритмов генерации. Сами эти алгоритмы использованы для генерации синтетических журналов событий, которые используются для оценки модульных алгоритмов исправления моделей процессов. Результаты оценки приводятся в следующей части диссертации.

В **четвёртой части** диссертации описывается программный прототип, реализующий модульный подход исправления моделей процессов, а также приводятся результаты экспериментальной оценки подхода с использованием данного прототипа.

Модульный подход исправления реализован в виде программного прототипа, который выполнен как расширение среды ProM 6 Framework [26], являющейся наиболее распространённым открытым инструментом в сообществе исследователей process mining. В данной части кратко описываются главные особенности этой реализации и её архитектура. Исходный код прототипа свободно доступен на странице проекта: `http://pais.hse.ru/research/projects/iskra`.

Для оценки алгоритмов исправления была использована экспериментальная схема, показанная на рисунке 12. Цель экспериментальной оценки — применить прототипные реализации алгоритмов к тестовым примерам, а затем измерить характеристики результирующих исправленных моделей. Результаты этих измерений демонстрируют сильные и слабые стороны алгоритмов, когда они применяются для исправления несоответствий различного характера.

В начале четвёртой части приводится описание нескольких моделей процессов, а также соответствующих синтетических журналов событий. Для каждого примера модели был сгенерирован журнал событий, идеально соответствующий модели.

Естественный способ протестировать и оценить *метод исправления* устроен следующим образом. Возьмём корректно функционирующий объ-

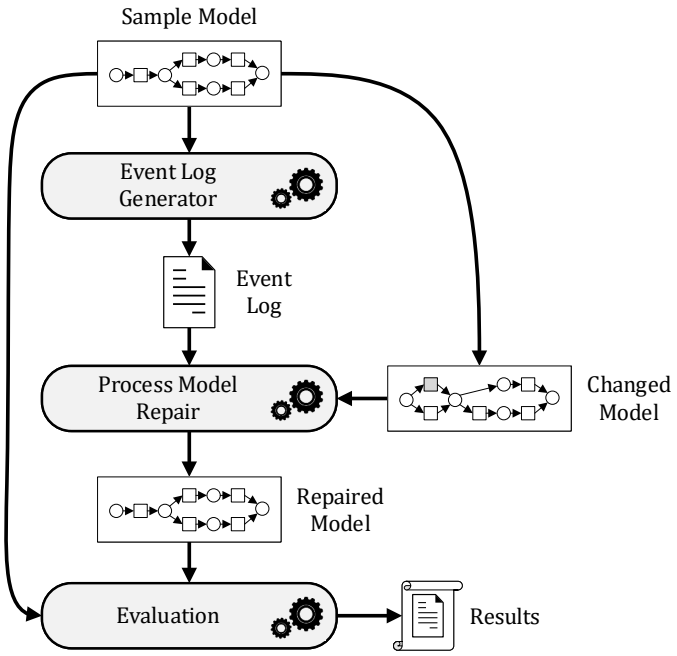

Рисунок 12: Схема экспериментальной оценки

ект Obj, внесём в него искусственные несоответствия (получим Obj^b), а затем попробуем исправить Obj^b с использованием метода, который требуется оценить/протестировать. Если метод исправления работает в соответствии с ожиданиями его разработчика, исправленный объект Obj^r должен работать так же, как и исходный исправный объект Obj. Если это не так, тест провален. При этом характеристики исправленного объекта могут быть измерены с использованием подходящих измерительных инструментов. Результаты измерений, полученные при проведении различных экспериментов, могут сравниваться между собой и анализироваться.

Именно таким образом и была устроена экспериментальная оценка модульного подхода исправления модели процесса, которая описывается в четвёртой части диссертации. В нашем случае метод исправления успешно проходит тест, если он способен реконструировать исходную модель, которая использовалась для генерации нормативного журнала событий. Кроме предложенных в данной работе, оценивались и другие методы исправления, чтобы сравнить результаты работы различных подходов. В данной части приводятся таблицы и рисунки с результатами исправления моделей процессов разнообразными алгоритмами.

Проведённые эксперименты показали работоспособность и применимость подхода исправления, предлагаемого в данной диссертации.

Наивный метод позволяет исправить модель по критерию соответствия журналу событий, но существенным образом снижает точность модели.

Более того, после исправления этим методом иногда получаются модели, допускающие очень много вариантов поведения, что не позволяет алгоритму, основанному на использовании выравниваний, успешно вычислить точность модели на имеющихся аппаратных средствах. Такое происходит в случаях, когда исправленная модель содержит много невидимых переходов и переходов без входных/выходных дуг.

Усовершенствованный метод справился с реконструкцией исходной модели в большинстве экспериментов и может быть признан эффективным по итогам экспериментальной оценки.

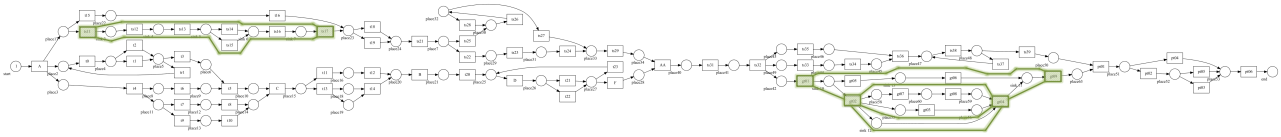

Рисунок 13: Модель, показанная на рисунке 1, исправленная с помощью усовершенствованного метода

Пример исправления модели, показанной на рисунке 1, с помощью усовершенствованного метода приводится на рисунке 13. Фрагменты модели, которые были заменены при исправлении, выделены зелёным цветом. Можно заметить, что алгоритм исправления сохранил структуру исходной модели.

Кроме того, оценивалась эффективность методов исправления в тех случаях, когда несоответствия нелокальны. В таких экспериментах лучше себя показал жадный метод исправления. Тем не менее, этот метод, как правило, вносит больше изменений в модель, чем усовершенствованный алгоритм. Используя данные, приводимые в данной части, нетрудно сравнить предложенные в диссертации методы исправления между собой и с другими методами, известными из литературы. Для рассмотренных примеров точность моделей, исправленных жадным методом, существенно не отличалась от моделей, синтезированных с нуля.

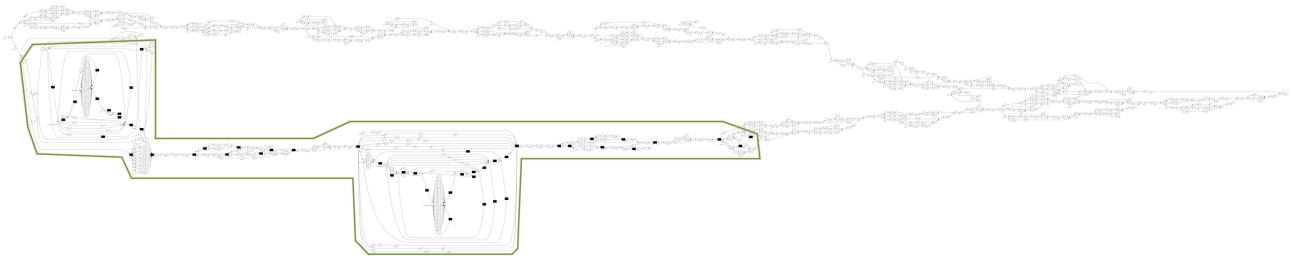

Рисунок 14: Большая модель, исправленная с помощью жадного метода с использованием индуктивного алгоритма синтеза

Методы, предложенные в диссертации, также показали свою применимость для моделей существенного размера. Например, на рисунке 14 пока-

зана модель, содержащая более тысячи узлов. Такие большие модели редко встречаются на практике. Данная модель была успешно исправлена с помощью жадного метода. Заменённый в ходе исправления фрагмент выделен на рисунке зелёным цветом.

Наконец, последней частью диссертации является **заключение**. В этой части обобщаются основные результаты данной диссертации. Кроме того, рассматриваются вопросы, оставшиеся открытыми и требующие дальнейших исследований. Возможные направления будущих работ, развивающих результаты диссертации, также приводятся в этой части.

Список литературы

1. *Dumas M., van der Aalst W.M.P., ter Hofstede A.H.M.*. Process-Aware Information Systems: Bridging People and Software through Process Technology. — Wiley & Sons, 2005.

2. *van der Aalst W.M.P.*. Process-Aware Information Systems: Design, Enactment and Analysis // Wiley Encyclopedia of Computer Science and Engineering / Ed. by B.W. Wah. — Wiley & Sons, 2009. — Pp. 2221–2233.

3. *Stahl Thomas, Voelter Markus, Czarnecki Krzysztof.* Model-Driven Software Development: Technology, Engineering, Management. — USA: John Wiley and Sons, Inc., 2006.

4. *Brambilla Marco, Cabot Jordi, Wimmer Manuel.* Model-Driven Software Engineering in Practice. — 1st edition. — Morgan and Claypool Publishers, 2012.

5. *van der Aalst Wil M. P.* Process Mining - Data Science in Action, Second Edition. — Springer, 2016.

6. *Fahland D., van der Aalst W.M.P.* Repairing Process Models to Reflect Reality // International Conference on Business Process Management (BPM 2012) / Ed. by A. Barros, A. Gal, E. Kindler. — Vol. 7481 of *Lecture Notes in Computer Science*. — Springer-Verlag, Berlin, 2012. — Pp. 229–245.

7. Generating event logs for high-level process models / Mitsyuk A.A., Shugurov I.S., Kalenkova A.A., van der Aalst W.M.P. // *Simulation Modelling Practice and Theory*. — 2017. — Vol. 74. — Pp. 1 – 16.

8. *Mitsyuk A.A., Lomazova I.A., van der Aalst W.M.P.* Using Event Logs for Local Correction of Process Models // *Automatic Control and Computer Sciences*. — 2017. — Vol. 51, no. 7. — Pp. 709–723.

9. *Мицюк А.А., Ломазова И.А., ван дер Аалст В.М.П.* Использование журналов событий для локальной корректировки моделей процессов // *Моделирование и анализ информационных систем*. — 2017. — Vol. 24, no. 4. — Pp. 459–480.

10. *Shugurov I.S., Mitsyuk A.A.* Iskra: A Tool for Process Model Repair // *Proceedings of the Institute for System Programming of the RAS*. — 2015. — Vol. 27, no. 3. — Pp. 237–254.

11. *Mitsyuk A.A., Shugurov I.S.* On process model synthesis based on event logs with noise // *Automatic Control and Computer Sciences.* — 2016. — Vol. 50, no. 7. — Pp. 460–470.

12. *Мицюк А.А., Шугуров И.С.* Синтез моделей процессов по журналам событий с шумом // *Моделирование и анализ информационных систем.* — 2014. — Vol. 21, no. 4. — Pp. 181–198.

13. *Mitsyuk A.A.* Non-Local Correction of Process Models Using Event Logs // Proceedings of the 2017 Ivannikov ISPRAS Open Conference / Los Alamitos : IEEE Computer Society. — 2018. — Pp. 6–11.

14. Process Model Repair by Detecting Unfitting Fragments / Mitsyuk A.A., Lomazova I.A., Shugurov I.S., van der Aalst W.M.P. // Supplementary Proceedings of the 6th International Conference on Analysis of Images, Social Networks and Texts (AIST-SUP 2017), Moscow, Russia, July 27-29, 2017. — Vol. 1975 of *CEUR-WS.org.* — CEUR-WS.org, 2017. — Pp. 301–313.

15. *Shugurov I.S., Mitsyuk A.A.* Generation of a Set of Event Logs with Noise // Proceedings of the 8th Spring/Summer Young Researchers Colloquium on Software Engineering (SYRCoSE 2014). — 2014. — Pp. 88–95.

16. Process mining can be applied to software too! / Rubin V.A., Mitsyuk A.A., Lomazova I.A., van der Aalst W.M.P. // ESEM. — ACM, 2014. — Pp. 57:1–57:8.

17. *Nesterov R.A., Mitsyuk A.A., Lomazova I.A.* Simulating Behavior of Multi-Agent Systems with Acyclic Interactions of Agents // *Proceedings of the Institute for System Programming.* — 2018. — Vol. 30, no. 3. — Pp. 285–302.

18. *Shugurov I.S., Mitsyuk A.A.* Applying MapReduce to Conformance Checking // *Proceedings of the Institute for System Programming of the RAS.* — 2016. — Vol. 28, no. 3. — Pp. 103–122.

19. *Nikitina N., Mitsyuk A.A.* Carassius: A Simple Process Model Editor // *Proceedings of the Institute for System Programming.* — 2015. — Vol. 27, no. 3. — Pp. 219–236.

20. Using process mining for the analysis of an e-trade system: A case study / Mitsyuk A.A., Kalenkova A.A., Shershakov S.A., van der Aalst W.M.P. // *Business Informatics.* — 2014. — Vol. 29, no. 3. — Pp. 15–27.

21. *Mitsyuk A.A., Kotylev Y.V.* Layered Layouts for Software Systems Visualization Using Nested Petri Nets // Tools and Methods of Program Analysis / Ed.

by Vladimir Itsykson, Andre Scedrov, Victor Zakharov. — Cham: Springer International Publishing, 2018. — Pp. 127–138.

22. *Adriansyah Arya.* Aligning observed and modeled behavior: Ph.D. thesis / Technische Universiteit Eindhoven. — 2014.

23. *Leemans S.J.J., Fahland D., van der Aalst W.M.P.* Discovering Block-Structured Process Models from Incomplete Event Logs // Application and Theory of Petri Nets and Concurrency / edited byGianfranco Ciardo, Ekkart Kindler. — Springer International Publishing, 2014. — Vol. 8489 of *Lecture Notes in Computer Science.* — Pp. 91–110.

24. Process Discovery using Integer Linear Programming / van der Werf J.M.E.M., van Dongen B.F., Hurkens C.A.J., Serebrenik A. // *Fundam. Inform.* — 2009. — Vol. 94, no. 3-4. — Pp. 387–412.

25. *Fahland D., van der Aalst W.M.P.* Model Repair - Aligning Process Models to Reality // *Inf. Syst.* — 2015. — Vol. 47. — Pp. 220–243.

26. ProM 6: The Process Mining Toolkit / Verbeek H.M.W., Buijs J.C.A.M., van Dongen B.F., van der Aalst W.M.P. // Proc. of BPM Demonstration Track 2010 / Ed. by M. La Rosa. — Vol. 615 of *CEUR Workshop Proceedings.* — 2010. — Pp. 34–39.