Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Информационные системы и базы данных

Акбаров Артур Александрович

Метод улучшения документации программного обеспечения на основе поиска нечётких повторов

Бакалаврская работа

Научный руководитель:
д.т.н., профессор кафедры системного программирования Кознов Д. В.

Рецензент:
старший преподаватель Луцив Д. В.

Санкт-Петербург
2018

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Information Systems and Databases

Artur Akbarov

Method of improving software documentation by searching for near duplicates

Bachelor's Thesis

Scientific supervisor:
Doctor of Engineering, Professor Dmitrij Koznov

Reviewer:
Senior lecturer Dmitry Luciv

Saint-Petersburg
2018

Оглавление

Введение **4**

1. Постановка задачи **6**

2. Обзор **7**

2.1. Концепция архетипа и дельт 7

2.2. Проект DocLine и неточные повторы 8

2.3. API-документация 8

2.4. Инструмент Duplicate Finder 9

3. Исследование иерархических связей повторов в документации ПО **11**

3.1. Мотивация 11

3.2. Рассмотренные проекты и их документация 12

3.3. Подход к построению иерархических групп 13

3.4. Иерархические группы повторов 16

3.5. Выводы 19

4. Инструмент по визуализации иерархических связей **20**

Заключение **23**

Список литературы **24**

Введение

В современном мире во всех сферах жизнедеятельности человеку помогает ПО. Для поддержки и совершенствования ПО необходима качественная документация.

ПО является сложным сплетением составляющих его модулей, и каждому из них соответствует своя часть документации. Модули зависят друг от друга, имеют схожий функционал — всё это находит отражение в документации в виде повторяющих друг друга описаний. Отсутствие контроля за такими повторениями порождает нестыковки, разночтения, опечатки, что ухудшает качество документации и усложняет понимание работы ПО. Исследованию повторов в документации ПО посвящено ряд исследований [2, 7, 11, 21].

При разработке документации её фрагменты могут быть раскопированы и изменены. Таким образом в документации появляются неточные повторы, которые ещё более затрудняют её дальнейшую поддержку. Каждый раз при внесении изменений в раскопированный участок нужно найти и соответственно исправить все его повторы, что непросто для неточных повторов [3]. Поэтому важно автоматически находить и отслеживать неточные повторы в документации.

В рамках проекта DocLine [22] был разработан подход для поиска и анализа повторов в документации ПО. Был реализован алгоритм поиска нечётких повторов на основе поиска клонов в ПО [13]. Был создан прототип алгоритма поиска нечётких повторов на основе N-gram модели [8], а также создан интерактивный подход к анализу нечётких повторов и выделению семантически значимых повторов [3, 17, 20, 21]. Предложенные идеи и алгоритмы были реализованы в программном инструменте Duplicate Finder [18].

Результатом работы инструмента Duplicate Finder является список найденных групп неточных повторов. Каждая группа представляется в «плоском» виде без учёта возможных разветвлённых зависимостей между повторами. Но документация сложно структурирована, и повторы в ней имеют многочисленные зависимости, пересечения, могут

образовывать иерархическую структуру. Для эффективной разработки и поддержки документации важно учитывать эту структуру. Однако исследований в области иерархических нечётких повторов не проводилось. Их изучению и посвящена данная работа.

1. Постановка задачи

Целью данной дипломной работы является разработка нового подхода для работы с нечёткими повторами, решающего проблему неоднозначной классификации повторов. Для достижения этой цели в рамках работы были сформулированы следующие задачи.

1. Изучить проект DocLine и инструмент Duplicate Finder.
2. Выполнить эмпирическое исследование иерархических повторов в реальной документации ПО.
3. Реализовать прототип инструмента по визуализации найденных иерархических повторов.

2. Обзор

2.1. Концепция архетипа и дельт

Архетипом множества текстовых фрагментов называется упорядоченный набор текстовых интервалов, входящих в каждый фрагмент в заданном порядке. Между этими интервалами во фрагментах возможны различные наполнения, называемые **вариативной частью** или **дельтами**. Эти понятия вместе с идеей параметризованных фреймов для переиспользования фрагментов контента были введены П. Бассеттом в 1997 году [1].

Проиллюстрируем введённые понятия на примере описаний двух схожих функций из Linux Kernel Documentation [5].

```
fixup_init
This function is called from the debug code whenever a problem in
debug_object_init is detected. The function takes the address of the
object and the state which is currently recorded in the tracker.
Called from debug_object_init when the object state is:
•    ODEBUG_STATE_ACTIVE
The function returns 1 when the fixup was successful, otherwise 0.
The return value is used to update the statistics.
Note, that the function needs to call the debug_object_init()
function again, after the damage has been repaired in order to keep
the state consistent.
```

```
fixup_activate
This function is called from the debug code whenever a problem in
debug_object_activate is detected.
Called from debug_object_activate when the object state is:
•    ODEBUG_STATE_NOTAVAILABLE
•    ODEBUG_STATE_ACTIVE
The function returns 1 when the fixup was successful, otherwise 0.
The return value is used to update the statistics.
Note that the function needs to call the debug_object_activate()
function again after the damage has been repaired in order to keep
the state consistent.
```

Рис. 1: Пример архетипа двух текстовых фрагментов

На рис. 1 представлены описания следующих функций:

- fixup_init — вспомогательная функция, вызываемая при проблемах в функции debug_object_init, принимающая адрес и состояние объекта.

- fixup_activate — функция для исправления проблем, возникающих в функции debug_object_activate.

Выделенный текст является общим для обоих описаний. Это и есть архетип данных описаний. Остальной текст представляет собой вариативную часть этих описаний.

2.2. Проект DocLine и неточные повторы

В документациях часто встречаются не только точные, но и неточные повторы. На их важность при работе с документацией указывалось в работах [2, 7, 11]. Однако их исследование проводились только в контексте проекта DocLine [13, 17, 20, 21, 22].

В проекте DocLine [20] введено формальное определение неточных повторов, которое можно понимать следующим образом. Множество текстовых фрагментов называется **группой неточных повторов**, если их архетип больше вариативной части каждого фрагмента.

Проиллюстрируем введённое понятие на примере группы, извлечённой из документации Eclipse SWT API [14].

На рис. 2 представлены 5 однотипных описаний классов. Каждый из них стандартно реализует один из интерфейсов-приёмников для работы с соответствующими интерфейсу событиями, что и указано в описании. Выделенный архетип, практически полностью исчерпывающий каждый повтор, наглядно показывает, что описания классов различаются лишь типами реализуемого интерфейса и его событиями.

2.3. API-документация

Согласно Дэвиду Парнасу [12] документация бывает двух видов: описательная (narrative) и справочная (reference). Описательная документация подразумевает, что её читают от начала и до конца, а справоч-

```
This adapter class provides default implementations for the methods
described by the ControlListener interface. Classes that wish to
deal with ControlEvents can extend this class and override only the
methods which they are interested in.
```

```
This adapter class provides default implementations for the methods
described by the ExpandListener interface. Classes that wish to deal
with ExpandEvents can extend this class and override only the
methods which they are interested in.
```

```
This adapter class provides default implementations for the methods
described by the FocusListener interface. Classes that wish to deal
with FocusEvents can extend this class and override only the methods
which they are interested in.
```

```
This adapter class provides default implementations for the methods
described by the KeyListener interface. Classes that wish to deal
with KeyEvents can extend this class and override only the methods
which they are interested in.
```

```
This adapter class provides default implementations for the methods
described by the MenuListener interface. Classes that wish to deal
with MenuEvents can extend this class and override only the methods
which they are interested in.
```

Рис. 2: Пример простой «плоской» группы повторов

ная предназначена для быстрого поиска нужной информации. Характерным примером справочной документации является API-документация. Её качество особенно важно, так как она является основным, а зачастую и единственным, источником по работе с описываемым ею ПО. Она состоит из логически замкнутых описаний функций, классов, интерфейсов. Данные описания удобно взять в качестве единицы текста в задаче поиска повторов, что и было сделано в примере на рис. 1.

2.4. Инструмент Duplicate Finder

Duplicate Finder — это инструмент, позволяющий находить и анализировать неточные повторы в документации [13, 18]. Инструмент читает

документы в «плоском» тексте, а также в специализированном формате DocBook [16], автоматически находит группы неточных повторов. Он позволяет вручную изменять границы каждого повтора, опираясь на его контекст. Представляет возможность для более тонкого интерактивного поиска дубликатов, их извлечения и рефакторинга. На данный момент инструмент не умеет однозначно определять границы повторов и требует экспертного уточнения каждого отдельного случая. Более того, от выбора этих границ сильно зависит размер найденной группы повторов, что сильно усложняет их анализ. За этим стоит тот факт, что повторы в документе представляют собой более сложную структуру нежели «плоскую» группу [21], которую на данный момент использует инструмент Duplicate Finder.

3. Исследование иерархических связей повторов в документации ПО

3.1. Мотивация

Кроме простых групп неточных повторов, подобных примеру с рис. 2, встречаются сложные, из которых можно выделить связанные подгруппы, и часто несколькими способами.

Рассмотрим для примера повторы из документации Eclipse SWT API, содержащие фрагмент «*Removes the listener from the collection of listeners who will be notified when*» — см. рис. 3. Всего таких повторов в документации содержится 59 штук (на рис. 3 представлены всего 5). Из них можно по-разному выделить две подгруппы, связанные с определённым событием или контроллером.

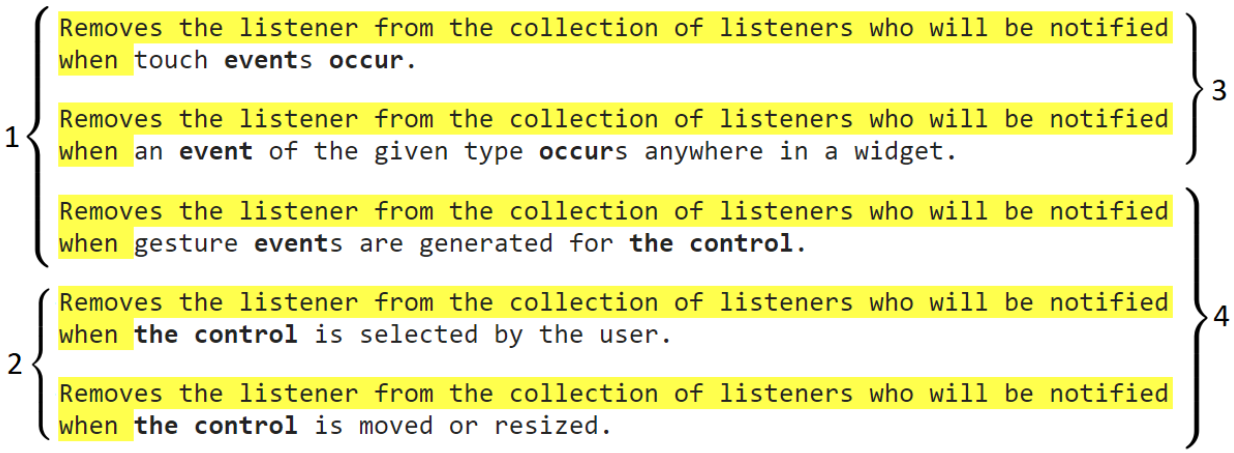

Рис. 3: Пример сложной «плоской» группы повторов

На рис. 3 показаны 2 разбиения представленных повторов. Первое разбиение повторов на подгруппы 1 и 2 задано фигурными скобками слева, второе делит повторы на подгруппы 3 и 4 фигурными скобками справа. В таблице 1 для каждой группы указаны определяющий её фрагмент и общее количество содержащих его повторов.

Таким образом очевидна необходимость анализа структуры встречающихся сложных групп повторов.

Таблица 1: Пример 2 различных разбиений повторов из документации Eclipse SWT API [14], содержащих фрагмент «*Removes the listener from the collection of listeners who will be notified when*»

Разбиение	**Номер группы**	**Фрагмент**	**Общее кол-во повторов**
левое	1	Removes the listener from the collection of listeners who will be notified when * event	12
	2	Removes the listener from the collection of listeners who will be notified when the control	17
правое	3	Removes the listener from the collection of listeners who will be notified when * event* occur	6
	4	Removes the listener from the collection of listeners who will be notified when *the control	23

3.2. Рассмотренные проекты и их документация

Для исследования были выбраны фрагменты документаций следующих крупных открытых проектов.

- Eclipse SWT — библиотека виджетов для языка Java [15].
- Linux Kernel — ядро операционной системы Linux [4].
- OpenLDAP — реализация протокола LDAP для доступа к службе распределённого каталога сети [9].

Каждый документ был взят с официального сайта в стандартном виде из множества веб-страниц и преобразован в один Word-документ с помощью утилиты Pandoc [6]. Для представления объёма документов в таблице ниже указано количество слов и страниц в полученных Word-документах.

Таблица 2: Объём выбранной документации ПО

Документ	Кол-во страниц	Кол-во слов	Источник
Eclipse SWT API Reference	1592	219 977	[14]
Linux Kernel Documentation	326	82 164	[5]
OpenLDAP Manual Pages	608	134 412	[10]

3.3. Подход к построению иерархических групп

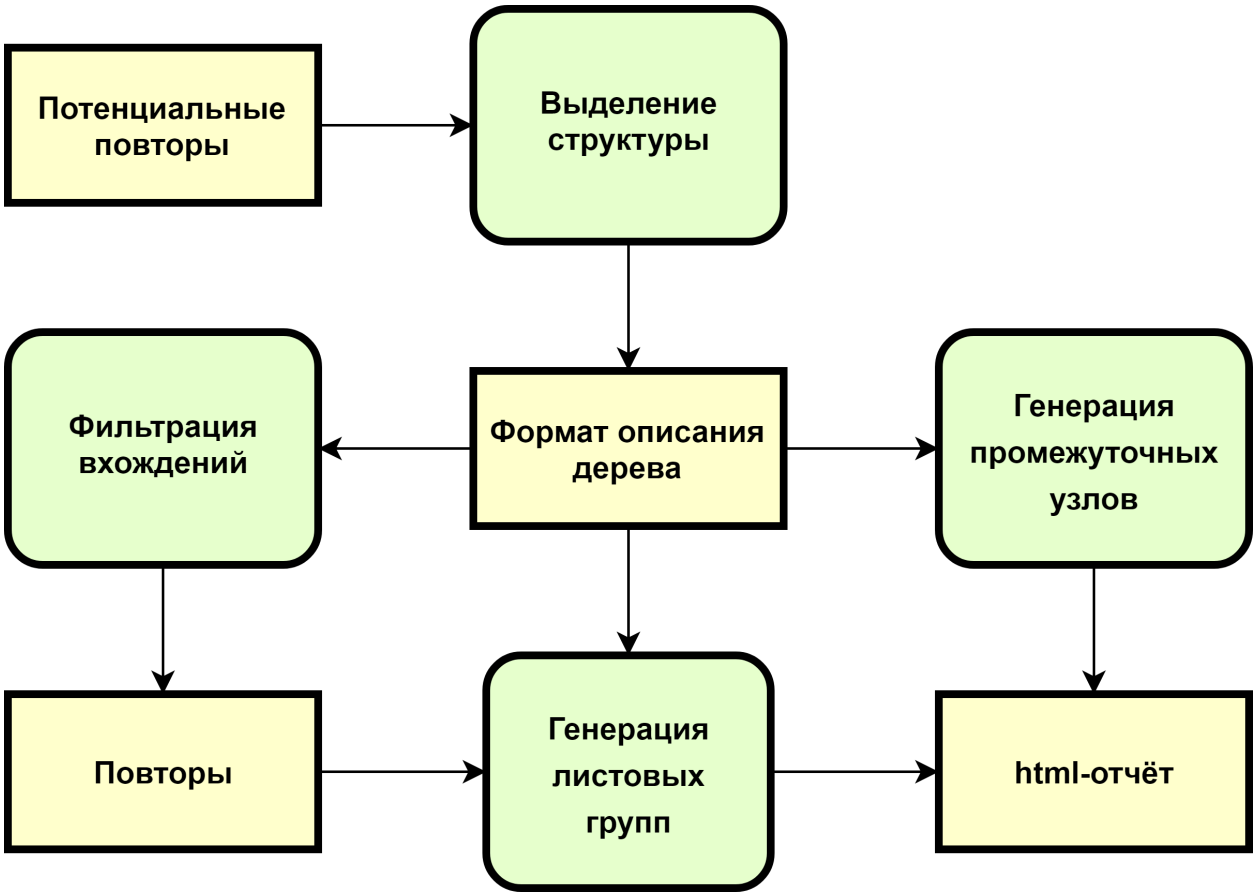

Рис. 4: Схема построения дерева повторов

На рис. 4 представлена схема построения дерева повторов в виде DFD-диаграммы (диаграмма потоков данных, data flow diagram [19]). На ней прямоугольниками обозначены данные, овалами — процедуры по их обработке. Опишем эту схему подробнее.

1. Процесс построения иерархической группы начинается от некой выборки повторов в анализируемом документе, найденной любым

способом (Потенциальные повторы).

2. Далее определяется архетип всех найденных повторов и характерная часть его записывается в корень дерева (Выделение структуры).

3. После этого выделяются все связанные подгруппы повторов. Для них также ищется архетип и его характерная часть записывается в соответствующий узел дерева (Выделение структуры).

4. Далее структура дерева сохраняется в своём специальном формате (Формат описания дерева).

5. Далее рассматриваются вхождения соответствующие построенному дереву и в специальном формате записываются только осмысленные из них (Фильтрация вхождений, Повторы).

6. Наконец, по структуре дерева и отфильтрованному множеству повторов строится html-отчёт (Генерация промежуточных узлов, Генерация листовых групп, html-отчёт).

Процесс построения дерева из исходной группы неточных повторов представляет собой семантическую кластеризацию повторов. Дерево задаётся регулярными выражениями в узлах таким образом, что выражение соответствует всем дочерним повторам и только им. Такое представление, несмотря на богатство языка регулярных выражений, скорее является синтаксическим, чем семантическим. Это задаёт некоторые рамки построения дерева, но облегчает процесс кластеризации и даёт наглядный смысл узлам дерева, в том числе промежуточным, объединяющим под собой совокупности различных семантически замкнутых подгрупп.

Процесс исследования отталкивается от произвольной группы повторов, которая берётся, например, из результатов работы инструмента Duplicate Finder. Определяется общий для всех повторов значимый фрагмент текста для задания корня дерева повторов. Исследуется синтаксический и семантический контекст повторов. Близкие повторы груп-

пируются в подгруппы, из которых также извлекается общий фрагмент для задания их листового узла. После этого по такому же принципу полученные подгруппы объединяются друг с другом до тех пор, пока не останется единственная корневая группа.

3.4. Иерархические группы повторов

В работе изучены 9 групп нечётких повторов. Для 6 групп были построены по дереву, для одной группы — три дерева. Из остальных 2 групп были выделены пересекающиеся подгруппы. В таблицах 3 и 4 указаны количественные характеристики разобранных групп, после каждая группа детально описывается.

Таблица 3: Количественные характеристики построенных деревьев

Номер	**Корень дерева**	**Высота**	**Общее кол-во узлов**	**Промеж. узлы**	**Конечные узлы**	**Повторы**	**Документ**
1	Removes the listener from...	5	36	10	26	60	Eclipse
2	the collection of polygons...1	3	12	3	9	12	
3	the collection of polygons...2	4	15	6	9	12	
4	the collection of polygons...3	3	13	4	9	12	
5	Note that a construct like...	2	4	1	3	36	OpenLDAP
6	strings reside in memory...	2	4	1	3	42	
7	The array is terminated by...	3	5	2	3	40	
8	This pointer should be...	2	4	1	3	9	
9	Error correction code byte	2	4	1	3	32	LKD

Таблица 4: Количественные характеристики групп повторов с выделенными пересекающимися подгруппами

Номер	**Название группы**	**Подгруппы**	**Повторы**	**Документ**
10	debug functions	4	7	LKD
11	fixup functions	4	5	

Пример 1 включает в себя 60 повторов, описывающих функции, удаляющие приёмники различных событий из коллекции приёмников. Вся совокупность повторов была выстроена в дерево с 26 конечными узлами, 10 из которых задают точные группы. В корне дерева расположено начало первого предложения описания функций «*Removes the listener from the collection of listeners (who/that) will be notified when*». Некорневые узлы уточняют это предложение, специфицируя ожидаемое приёмником событие. Полученное дерево имеет высоту 5 и 10 промежуточных узлов, объединяющих подгруппы со схожими ожидаемыми событиями.

Примеры 2, 3 и 4 представляют собой одну и ту же группу повторов функций добавления, удаления, пересечения элементов коллекции многоугольников, задающих графические области. В качестве примера неоднозначного синтаксического построения дерева они представлены различными деревьями с одинаковым корневым элементом. Разница состоит в структуре и способе выбора промежуточных узлов, которых в разных деревьях 3, 5, 4 штук, а высота деревьев равна 3, 4, 3 соответственно. В корне у всех стоит конец первого предложения описания «*the collection of polygons the receiver maintains to describe its area.*». В последующих узлах дополняется начало корневого предложения, определяющее объект и действие по работе с коллекцией. Наиболее удобным для анализа представляется дерево из примера 3, чьи промежуточные узлы на втором уровне определяют действие, совершаемое с данной коллекцией многоугольников, а на третьем — объект этого действия. Деревья примеров 2 и 4 выделяют большую ветку с частым объектом действия, что подчёркивает факт популярности использования этого объекта, но не даёт понятной классификации всех случаев на верхнем уровне дерева.

Пример 5 представляет собой группу повторов кодов исправления ошибок с корнем «*Error correction code byte*». Группа представлена деревом из 3 конечных узлов и высотой 2. Листовые группы разнятся незначительными расхождениями, возможно случайными, и подробностью описания кодов.

В документации OpenLDAP встречаются таблицы описания флагов

форматных строк для функций наподобие printf, scanf. В примере 6 из этих описаний была выделена группа, состоящая из 36 повторов. Она была выстроена в дерево с корнем «*Note that a construct like '{letter}' is required to get an actual SEQUENCE OF octet strings.*». Дерево имеет высоту 2 и содержит 3 конечных узла, специфицирующих описываемый флаг «'{letter}'». Каждая листовая группа состоит из 12 точных повторов, отличающихся флагом и ожидаемым типом аргумента, который флаг задаёт.

Схожая группа из 40 однотипных описаний флагов форматных строк разобрана в примере 7. Она была выстроена в дерево с корнем «*The array is terminated by a struct berval with a NULL bv_val*». Дерево имеет высоту 3 и состоит из одного промежуточного узла и 3 конечных, задающих 3 точные группы повторов размерами по 14, 14 и 12 элементов. Промежуточный узел объединяет два конечных узла из таблиц флагов функций декодирования, в то время как оставшийся третий конечный узел относится к функциям кодирования.

В примере 8 рассмотрена группа 42 повторов описаний флагов и функций декодирования, составляющих параграф «LBER_DECODE». Из них построено дерево с корнем «*strings? *resides? in memory assigned to the BerElement,? and must not be freed by the caller.*». Дерево имеет высоту 2 и 3 конечных узла, задающих 2 точные группы описаний флагов и одну точную группу описаний функций. Каждая точная группа состоит из 14 элементов.

В примере 9 разобрана группа повторов, состоящая из 9 описаний функций для анализа результатов LDAP операций. Из них построено дерево с корнем «*This pointer should be supplied on a subsequent call to word() to get the next *, the result of which should be supplied to the next call to word(), etc. word() will return NULL when there are no more*». Дерево имеет высоту 2 и 3 конечных узла, задающих 3 группы точных повторов, каждая размером по 3. Эти 3 группы описывают функции по работе с 3 составляющими результатов LDAP операций — цепочками ссылок, сообщениями и самими результатами.

В примере 10 выделена группа повторов описания 7 отладочных

функций «debug functions». Они целиком составляют отдельный параграф документации с одноимённым названием. Описания имеют множественные попарные заимствования, потому не были собраны в дерево. Вместо этого были выделены 3 пересекающиеся подгруппы, не считая общей группы всех 7 описаний, с наибольшими архетипами. Размеры групп составили 5, 3, 2.

Аналогичная ситуация обстоит с примером 11. Он включает в себя повторы описаний 5 исправительных функций «fixup functions», составляющих следующий за «Debug functions» параграф и тесно с ним связанных. Из них были выделены 3 пересекающиеся подгруппы, не считая общей группы из 5 описаний, с наибольшими архетипами. Размеры групп составили 4, 2, 2.

3.5. Выводы

В исследованных документах содержатся группы повторов явно имеющие иерархическую структуру. Выделение этой структуры только на основе синтаксиса несложно, но бóльшую ценность для поддержания документации представляет собой семантичное выделение. Оно, в свою очередь, затруднительно, требует ручного разбора и дополнительных подходов.

4. Инструмент по визуализации иерархических связей

Инструмент представляет собой браузер с разобранными примерами групп повторов. Каждый пример представлен интерактивным деревом, в листьях которого перечислены повторы, а в узлах выписан фрагмент, присутствующих во всех дочерних повторах и только в них. Архетип листовой группы повторов и родительский узел дерева выделены в повторах цветом. По клику в дереве на повтор контекстный модуль в нижней панели инструмента открывает контекст повтора в документации.

Инструмент помогает увидеть тесно связанные подгруппы повторов внутри одной большой группы повторов. Помогает отличить случайные повторы от закономерных. Наглядно отделяет одни группы от других по тексту родительских узлов. Даёт возможность структурно взглянуть на образованные в документации зависимости.

На рис. 5 представлен скриншот браузера, в котором отражены коллекции разобранных групп повторов, сгруппированных по документам.

На рис. 6 представлен скриншот дерева из примера 1 (см. таблицу 3). В верхней панели расположены повторы и узлы дерева с порядковыми номерами дочерних повторов, а в нижней — контекст повтора в документации.

На рис. 7 представлена диаграмма компонент инструмента по визуализации, а на рис. 8 — его диаграмма последовательности.

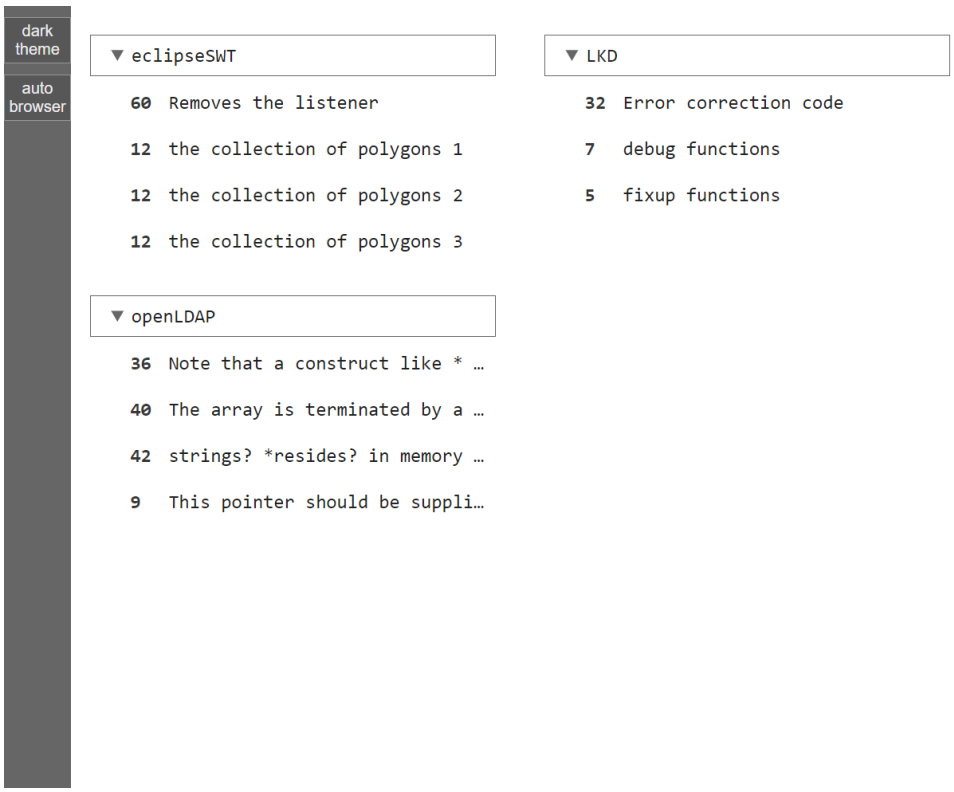

Рис. 5: Скриншот страницы с браузером инструмента по визуализации

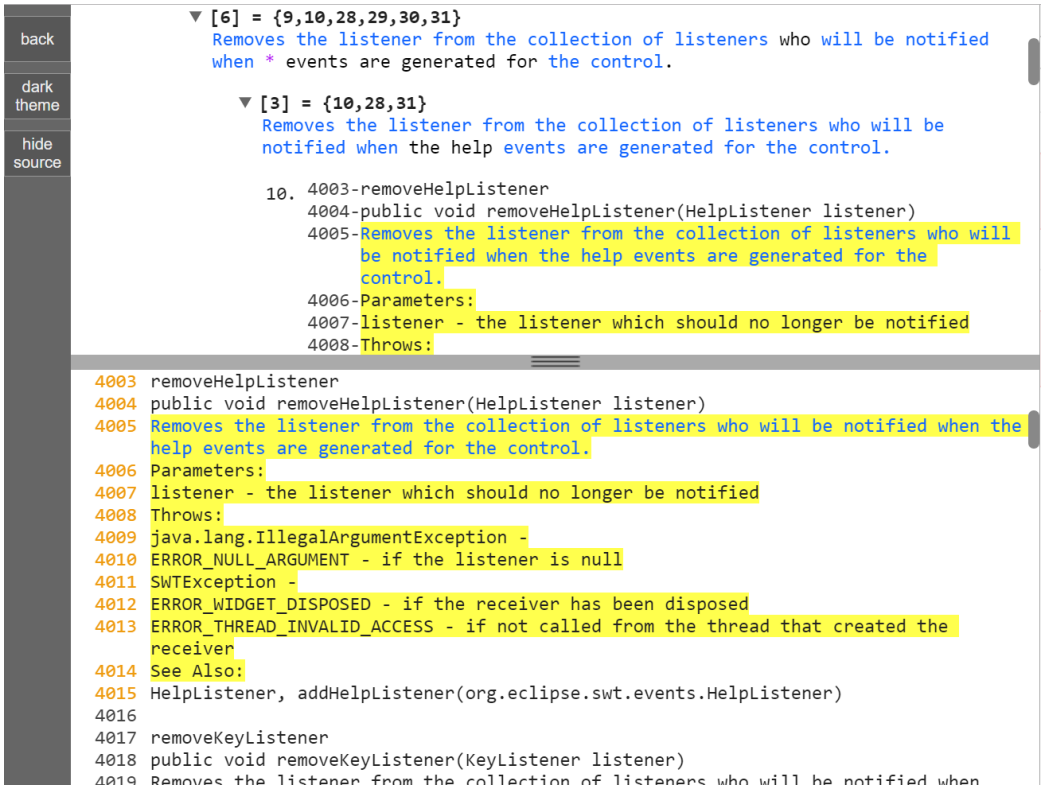

Рис. 6: Скриншот страницы с деревом инструмента по визуализации

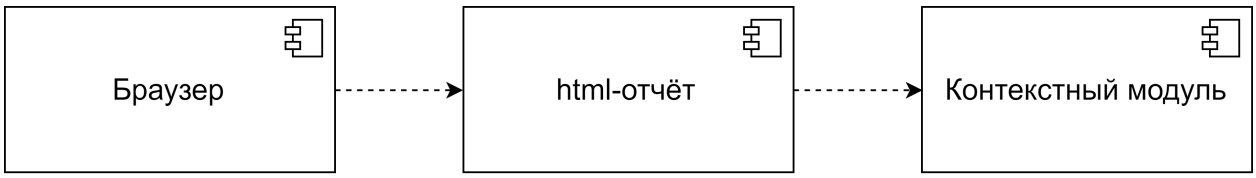

Рис. 7: Диаграмма компонент инструмента визуализации

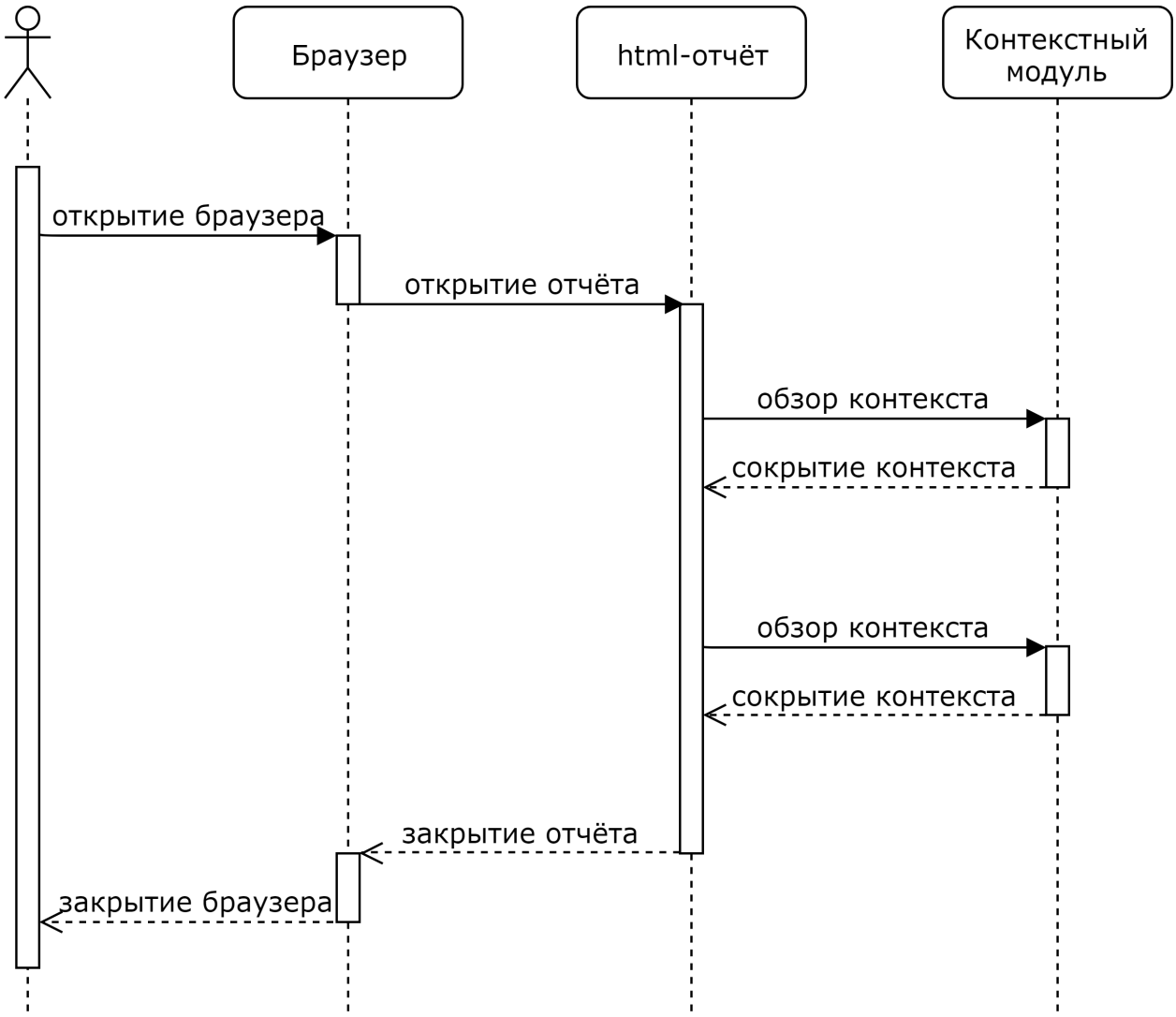

Рис. 8: Диаграмма последовательности инструмента визуализации

Заключение

В рамках данной работы были достигнуты следующие результаты.

1. Изучены возможности поиска и представления групп нечётких повторов инструмента Duplicate Finder. В том числе выявлена проблема отображения взаимосвязанных групп.

2. Найдены и проанализированы 9 иерархических групп нечётких повторов в документации проектов Eclipse SWT, Linux Kernel, OpenLDAP. Средняя высота и среднее количество промежуточных узлов построенных деревьев равны 3. Сделан вывод о наличии иерархических групп нечётких повторов в реальной документации. Предложен подход к их построению.

3. Реализован прототипный веб-инструмент, визуализирующий найденные иерархические группы повторов. В него входят построенные деревья, браузер по ним и контекстный модуль.

Инструментарий и разобранные примеры можно найти по ссылке:
`https://github.com/arthur-akbarov/hierarchical-near-duplicates.git`

Список литературы

[1] Bassett P. Framing software reuse — lessons from real world. — Prentice Hall, 1996.

[2] Can clone detection support quality assessments of requirements specifications? / E. Juergens, F. Deissenboeck, M. Feilkas et al. — Proceedings of ACM/IEEE 32nd International Conference on Software Engineering, 2010.

[3] Clone Detection in Reuse of Software Technical Documentation / Dmitrij Koznov, Dmitry Luciv, Hamid Abdul Basit et al. — 2015.

[4] The Linux Kernel Archives, Linux Kernel Organization, Inc. — URL: `https://www.kernel.org/`.

[5] The Linux Kernel documentation, The Kernel Development Community. — URL: `https://www.kernel.org/doc/html/latest/`.

[6] MacFarlane J. Pandoc: an universal document converter. — URL: `http://pandoc.org/`.

[7] Nosál' M., Porubän J. Reusable software documentation with phrase annotations. — Central European Journal of Computer Science, 2014.

[8] On Fuzzy Repetitions Detection in Documentation Reuse / D.V. Luciv, D.V. Koznov, A.N. Terekhov, H.A. Basit. — Programming and Computer Software, 2016.

[9] OpenLDAP, OpenLDAP Foundation. — URL: `https://www.openldap.org/`.

[10] OpenLDAP Manual Pages, OpenLDAP Foundation. — URL: `https://www.openldap.org/software/man.cgi`.

[11] Oumaziz M.A. et al. Documentation Reuse: Hot or Not? An Empirical Study. — Proceedings of 16th International Conference on Software Reuse, 2017.

[12] Parnas D.L. Precise Documentation: The Key to Better Software. — 2011.

[13] Poster: Duplicate Finder Toolkit / Dmitry Luciv, Dmitrij Koznov, George Chernishev et al. — 2018.

[14] SWT Documentation, Eclipse Foundation, Inc. — URL: `https://www.eclipse.org/swt/docs.php`.

[15] SWT: The Standard Widget Toolkit, Eclipse Foundation, Inc. — URL: `https://www.eclipse.org/swt/`.

[16] Walsh N. DocBook 5: The Definitive Guide. — O'Reilly Media, 2010.

[17] Задачи поиска нечётких повторов при организации повторного использования документации / Д.В. Луцив, Д.В. Кознов, Х.А. Басит, А.Н. Терехов. — Программирование, 2016.

[18] Инструмент Duplicate Finder, кафедра системного программирования СПбГУ. — URL: `http://www.math.spbu.ru/user/kromanovsky/docline/duplicate-finder-ru.html`.

[19] Кознов Д.В. Основы визуального моделирования. — БИНОМ. Лаборатория знаний, 2007. — С. 248.

[20] Луцив Дмитрий. Поиск неточных повторов в документации программного обеспечения. — 2018.

[21] Обнаружение неточных повторов в документации программного обеспечения / Д.В. Луцив, Д.В. Кознов, Г.А. Чернышёв и др. — Программирование №5 (Принята), 2018.

[22] Проект DocLine, кафедра системного программирования СПбГУ. — URL: `http://www.math.spbu.ru/user/kromanovsky/docline/index.html`.