СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4

1 Основная часть 5

1.1 Формализованная постановка задачи 5

1.2 Входные и выходные данные 5

1.3 Реализация 6

1.4 Сравнительный анализ времени выполнения этапов метода 7

ЗАКЛЮЧЕНИЕ 8

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9

ПРИЛОЖЕНИЕ А 11

ПРИЛОЖЕНИЕ Б 14

ВВЕДЕНИЕ

Во время выполнения выпускной квалификационной работы был разработан метод анализа активности пользователей системы автоматизированного проектирования с использованием поиска последовательных шаблонов.

1 Основная часть

1.1 Формализованная постановка задачи

Для выполнения работы необходимо формализовать задачу анализа активности пользователей САПР. Поставленная задача представлена в нотации IDEF0 на рисунке 1.1. На вход программе подаются информация о выполненных командах и пользовательские параметры: минимальный уровень поддержки, минимальный и максимальный разрывы между командами. Используя методы поиска последовательных шаблонов система определяет часто встречающиеся последовательности команд.

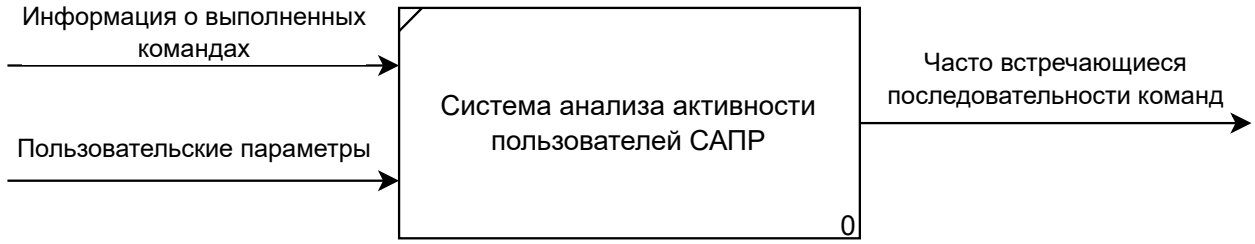

Рисунок 1.1 — IDEF0-диаграмма нулевого уровня

1.2 Входные и выходные данные

Данная программа разрабатывается для анализа логов САПР NanoCAD, которые имеют следующий вид: каждая строчка, содержащая действие, начинается с даты и времени выполнения. Начало каждой команды обозначается ее названием, заключенной в символы ’<’ и ’>’. Завершение команды обозначается аналогично, но перед названием команды добавляется символ ’/’.

Также поддерживается считывание обезличенных логов, которые не содержат информации о параметрах команд, включая время выполнения. В таком случае время выполнения для каждой команды в сессии будет выставлено автоматически, начиная с 0, увеличивая это значения на 1 для каждой следующей команды.

Кроме этого на вход программе подаются минимальный уровень поддержки, а также минимальный и максимальный разрывы между командами.

На выходе программа выдает часто встречающиеся последовательности команд, их уровни поддержки и коэффициент зависимости. Коэффициент зависимости показывает насколько команды в последовательности зависят друг от друга и считается как отношение поддержки последовательности к произведению поддержек всех подпоследовательностей состоящих из 1 команды. Если значение коэффициента $<= 1$, значит зависимости нету. Если же > 1, то зависимость есть. Чем больше единицы, тем вероятней то, что эти команды использовались вместе.

1.3 Реализация

За преобразование логов из текста в таблицу базы данных отвечает класс *LogReader*. Он может считать все файлы с расширением *.log в выбранной директории и её поддиректориях, записывая все команды в таблицу logs для выбранной на текущий момент базы данных. Также можно указать, нужно ли учитывать завершение команды как отдельное действие, если она началась и закончилась одновременно.

За взаимодействие с базами данных отвечает класс *DataBase*. Данный класс позволяет создавать базы данных, переключаться между ними и как записывать в них необходимые данные, так и считывать их.

За реализацию разрабатываемого метода отвечает класс *Calculator*. Он хранит входные параметры метода, а также необходимые для работы данные и последний полученный результат.

Описание данных классов приведено в приложении А.

Пользовательский интерфейс состоит из 4-ех окон. *MainWindow* – основное окно в котором можно выбирать и просматривать базу данных, считывать в нее логи из выбранной директории, настраивать режимы считывания, задавать параметры метода, запускать его для текущей базы данных и наблюдать результат. *DataBaseWindow* и *ResWindow* используются для просмотра базы данных и результатов работы программы в отдельных окнах. Последнее окно *CmdListWindow* содержит расширенные настройки, в котором можно указать

команды которые будут игнорироваться и которые означают начало новой сессии.

Интерфейс программы см. в приложении Б.

1.4 Сравнительный анализ времени выполнения этапов метода

Чтобы провести сравнительный анализ времени выполнения этапов метода, замерялось их время выполнения с разными значениями минимальной поддержки и количеством записей 1000 раз, а затем делилось на количество замеров. Параметр min_gap был равен нулю, а max_gap имел максимально возможное значение На рисунке 1.2 представлен результат исследования в виде графиков.

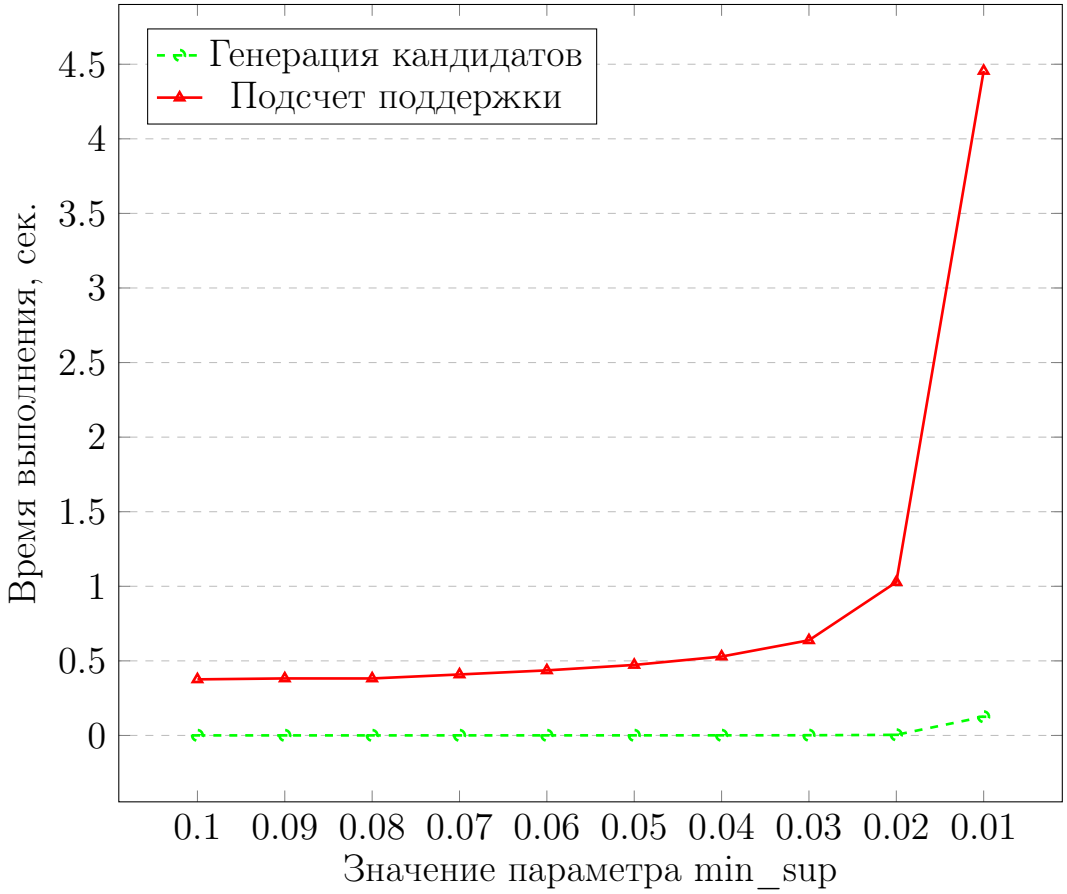

Рисунок 1.2 — Зависимость времени выполнения разных этапов метода от минимальной поддержки для 66788 записей

В результате анализа, можно сделать вывод, что подсчет поддержки кандидатов занимает большую часть времени, чем их генерация.

ЗАКЛЮЧЕНИЕ

Было разработано программное обеспечение, демонстрирующее практическую осуществимость спроектированного в ходе выполнения выпускной квалификационной работы метода анализа активности пользователей системы автоматизированного проектирования с использованием поиска последовательных шаблонов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Siochi A.C., Ehrich R.W Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions. // ACM Transactions on Information Systems. – 1991. – Т. 9. – № 4. – С. 309–335.

2. Данилов Н.А., Шульга Т.Э. Метод построения тепловой карты на основе точечных данных об активности пользователя приложения // Прикладная информатика. – 2015. – Т. 10. – № 2. – С. 49–58.

3. Danilov N., Shulga T., Frolova N., Melnikova N., Vagarina N., Pchelintseva E. Software usability evaluation based on the user pinpoint activity heat map. // Advances in Intelligent Systems and Computing. – 2016. – Т. 465. – С. 217–225.

4. Balbo S., Goschnick S., Tong D., Paris C. Leading Usability Evaluations to WAUTER. // Proc. 11th Australian World Wide Web Conf. (AusWeb), Gold Coast, Australia, Southern Cross Univ. – 2005. – С. 279–290.

5. Swallow J., Hameluck D., Carey T. User interface instrumentation for usability analysis: a case study. // CASCON'97. – Toronto, Ontario. – 1997.

6. Shah I. Event patterns as indicators of usability problems. // Jour. of King Saud Univ., Comp. and Inform. Sci. – 2008. – С. 31–43.

7. Mabroukeh N.R., Ezeife C.I. A taxonomy of sequential pattern mining algorithms. // ACM Computing Surveys (CSUR). – 2010. – Т. 43. – № 1. – статья № 3.

8. Aloysius G., Binu D. An approach to products placement in supermarkets using prefixspan algorithm. // Jour. of King Saud Univ. Comp. and Inform. Sci. – 2013. – Т. 25. – № 1. – С. 77–87.

9. Сытник А.А., Шульга Т.Э., Данилов Н.А., Гвоздюк И.В. Математическая модель активности пользователей программного обеспечения. // Программные продукты и системы. – 2018. – Т. 31. – № 1. – С. 79-84

10. Agrawal R., Imielinski T., Swami A.N. Mining Association Rules between Sets of Items in Large Databases // Proceedings of the 1993 ACM

SIGMOD international conference on Management of data. SIGMOD '93. – Washington, D.C., USA. – 1993. – T. 22(2). – C. 207-216.

11. Agrawal R., Srikant R. Fast algorithms for mining association rules // Proceedings of the 20th International Conference on Very Large Data Bases, VLDB. – Santiago, Chile. – 1994. – C. 487-499.

12. Zaki J Mohammed, Meira Jr Wagner. Data Mining and Analysis: Fundamental Concepts and Algorithms. – New York: Cambridge University Press, 2014. – C. 595

13. Agrawal R., Srikant R. Mining Sequential Patterns // Proc. of the 11th Int'l Conference on Data Engineering. – 1995. – C. 3–14.

14. Srikant R., Agrawal R. Mining Sequential Patterns: Generalizations and Performance Improvements // EDBT. Springer Berlin Heidelberg. – 1996. – C. 1–17.

15. Интеллектуальный анализ данных: учеб. пособие. – Томск: Издательский Дом Томского государственного университета, 2016. – С. 120

16. Card S., Moran T., Newell A. The keystroke-level model for user performance time with interactive systems. // Communications of the ACM. – 1980. – Т. 23. – № 7. – С. 396–410.

17. Бьёрн Страуструп Язык программирования C++ - специальное издание. Москва: Бином, 2010. 1136 с.

18. Qt documentation [Электронный ресурс] // Qt. URL: `http://doc.qt.io/` (дата обращения: 14.08.2021)

ПРИЛОЖЕНИЕ А

Листинг 1.1 — Класс LogReader

```
class LogReader
{
public:
  LogReader();

  static shared_ptr<LogReader> instance();

  void readLogs(QString dir_name = "./");
  void readLogsWithoutTime(QString dir_name = "./");
  void includeEndCmds(bool val = true);
  QStringList getIgnoreList() const;
  QStringList getNewSessionCmdsList() const;
  void setIgnoreList(QStringList list);
  void setNewSessionCmdsList(QStringList list);

private:
  void readFile(const QFileInfo& file_info, QList<QString> &commands, int
     &session_id);
  void readDir(const QString& abs_path, QList<QString> &commands, int
     &session_id);
  void readFileWithoutTime(const QFileInfo& file_info, QList<QString>
     &commands, int &session_id);
  void readDirWithoutTime(const QString& abs_path, QList<QString> &commands,
      int &session_id);
  int getTimeFromRecord(QString r);
  bool getCommandFromRecord(QString r, QString& res);
  QStringList getAllCommandsFromRecord(QString r);
  QStringList getTwoCommandsFromRecord(QString r);

private:
  QDir::Filters dir_filters;
  QStringList ignore_commands;
  QStringList new_session_commands;
  bool end_cmds;
};
```

Листинг 1.2 — Класс DataBase

```
enum Status
{
  OK = 0,
  EXEC_ERROR,
  EMPTY_RES,
  DATABASE_OPEN_ERROR,
  DATABASE_DOES_NOT_EXISTS,
  DATABASE_IS_NOT_VALID
};

class DataBase
{
public:
  DataBase();

  static shared_ptr<DataBase> instance();

  Status setSQLiteDataBase(QString db_name = "db_name");
  Status resetSQLiteDataBase();
  bool databaseExists(QString db_name);
  Status getRowsInLogs(QString db_name, int &rows_number);
  Status getSessionsInLogs(int &sessions_n);
  Status addCommand(int session_id, const QString& datetime, const QString
     &cmd, int &id);
  Status addCommand(int session_id, int int_time, const QString &cmd, int &id);
  Status getCmdsMap(QMap<int, QString>& cmds_map);
  Status getSessionsNum(int& sessions_num);
  Status getAllLogs(int commands_num, QList<Session>& sessions);

  QString lastError();

private:
  inline Status execQuery(QString query);

private:
  QString m_last_error;
  QString cur_db_name;
  QSqlQuery m_query;
};
```

Листинг 1.3 — Класс Calculator

```
class Calculator
{
public:
  Calculator();

  QList<Sequence> getFrequentSequences(double _min_sup = -1, int _min_gap = -1,
      int _max_gap = -1);
  void printFrequentSequences();
  QString getSeqStr(const Sequence &seq);
  void setSameCmds(bool val);

private:
  void prepareGSP();
  QList<Sequence> generateCandidates1();
  QList<Sequence> generateCandidates();
  bool findCommand(int cmd, const Session& session, int min_time, int
      prev_cmd_id, int &time, int &id) const;
  bool sessionSupportsSequence(const Session& session, const Sequence& seq);
  Sequence findFreqSequenceByCommand(int cmd);
  double calcLift(Sequence seq);
  int countSupport(QList<Sequence> &candidates, const QList<Session> &sessions);
  void sortFrequentSequences();

private:
  QList<Sequence> freq_seqs;
  QList<Sequence> cur_freq_seqs;
  QMap<int, QString> cmds_map;
  int sessions_count = 0;
  QString db_file_path;

  double min_support = 0.5;
  int min_gap = 0;
  int max_gap = INT_MAX;
  bool same_cmds = true;
};
```

ПРИЛОЖЕНИЕ Б

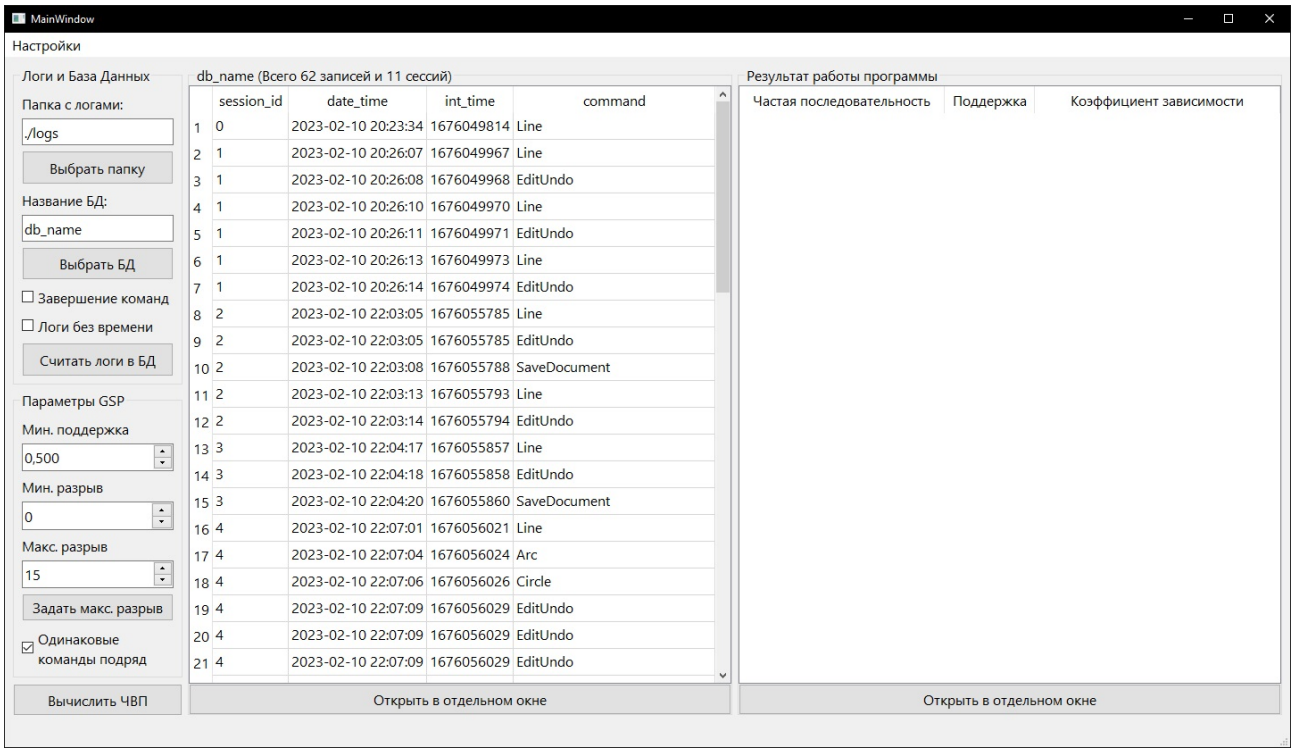

Рисунок 1.1 — Интерфейс программы 1

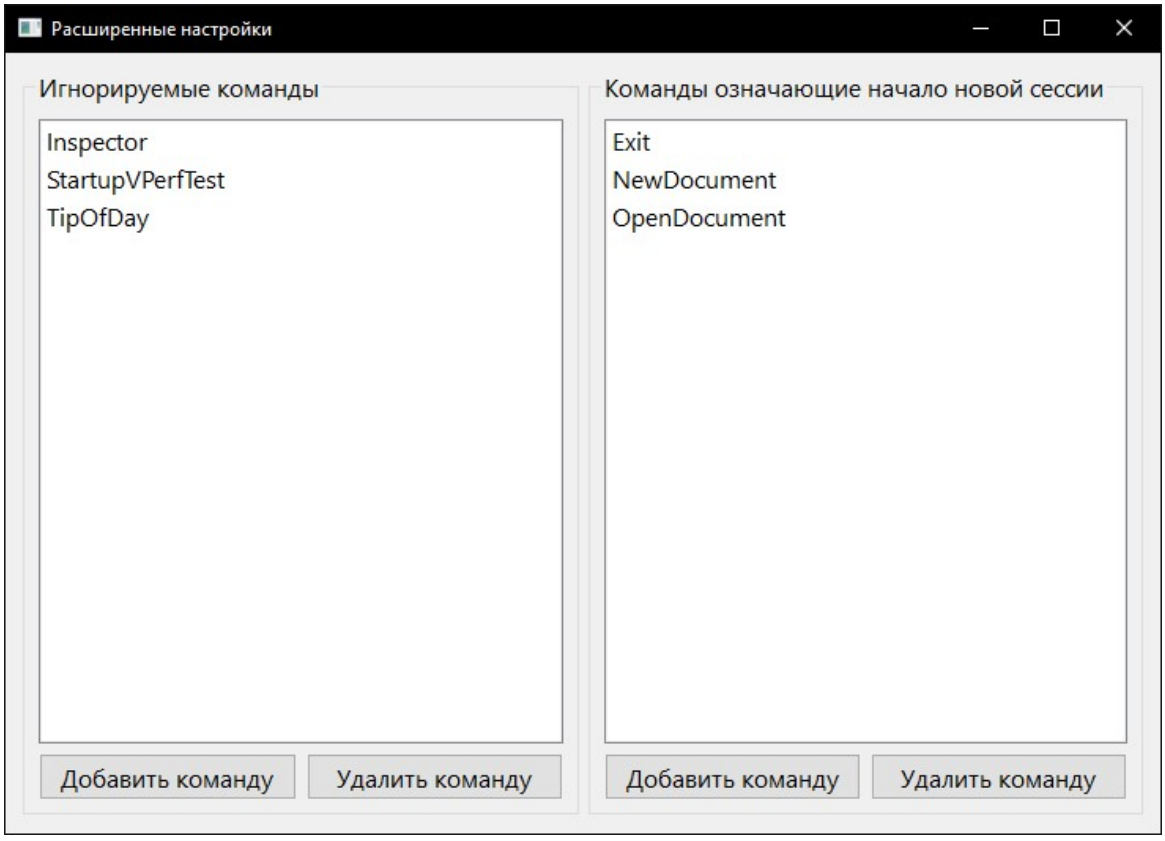

Рисунок 1.2 — Интерфейс программы 2