УДК 004.436.4

А.А. БЛАЖКО, Е.В. ЛЯШЕНКО, ИБАА САУД

Одесский национальный политехнический университет, Украина

ОБНАРУЖЕНИЕ КОНФЛИКТОВ В ПРАВИЛАХ ДОСТУПА К РЕЛЯЦИОННЫМ БАЗАМ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА UML И OCL

Статья посвящена разработке системы автоматизированного управления доступа к данным пользователей корпоративных информационных систем. Цель статьи – повышение достоверности работы системы автоматизированного управления доступа к данным. Для достижения поставленной цели в работе предложено решать задачу верификации на непротиворечивость правил доступа к данным через их представление в виде OCL-ограничений. Это позволило использовать открытый программный пакет DresdenOCL, проверяющий корректность выражений, и пакет UMLtoCSP, обнаруживающий конфликты между OCL-ограничениями правил.

Ключевые слова: *корпоративные информационные системы (КИС), база данных (БД), OCL(Object Constraint Language), UML(Unifed Model Language), XMI (XML Metadata Interchange), MOF (Meta-Object Facility).*

.

Введение

Для корпоративных информационных систем (КИС), обслуживающих сотни пользователей при их доступе к сотням таблиц реляционной БД, важно автоматизировать процесс управления доступом к БД с использованием разных политик безопасности [1]. В работе [2] предложена технология подобной автоматизации, которая генерирует базу данных виртуальных таблиц представлений субъектов об объектах, но в ходе экспериментов была обнаружена её уязвимость к конфликтам в правилах доступа к данным из спецификаций и неоднозначности в формировании структурно-должностной иерархии пользователей КИС.

Решение указанных проблем может быть основано на формализации правил управления доступом в предметной области и верификации этих правил на непротиворечивость. В большинстве методов формализации используется логика предикатов первого порядка. При этом желательно использовать формализации, поддерживаемые современными стандартами в проектировании. Существует множество формальных моделей представления правил на основе языка объектно-ориентированного проектирования *UML (Unifed Model Language)* и *OCL (Object Constraint Language).* В системе *Dresden OCL* [3] поддерживается спецификация и валидация *OCL*-ограничений. В системе *USE* [4] выражения, написанные на *OCL*, используются для уточнения добавочной интеграции ограничений на модель, которая может быть анимирована для валидации спецификаций. В системе *UMLtoCSP* [5] предложено инструментальное средство, позволяющее автоматически верифицировать *UML*-модель с *OCL*-ограничениями. В тоже время, в указанных системах отсутствуют реализации этих моделей на примере процессов управления доступом к БД КИС.

Целью данной работы стало повышение достоверности работы системы. Для достижения поставленной цели необходимо решить задачу снижения уровня конфликтности в наборе требований спецификаций правил доступа пользователей к данным в БД КИС, которые предварительно формализованы с использованием *OCL*-языка.

1. Модели ролевой формализации

Для достижения указанной цели работы предложена программная система, схема взаимодействия компонент которой представлена на рис. 1.

Исходными данными для работы системы являются:

1) *UML*-диаграмма концептуальных классов системы;

2) Словесное описание набора требований спецификаций правил доступа сотрудников к документам в производственных процессах.

Пример *UML*-диаграммы концептуальных классов для информационной системы «Электронный деканат» представлен на рис. 2. Данная диаграмма разработана в открытом программном пакете *ArgoUML* и там же была экспортирована в *XMI*-файл.

Для последующей автоматизированной обработки *UML*-диаграмма должна быть представлена в *XMI*-формате. *XMI (XML Metadata Interchange)* — стандарт *OMG* для обмена метаданными с помощью языка *XML*.

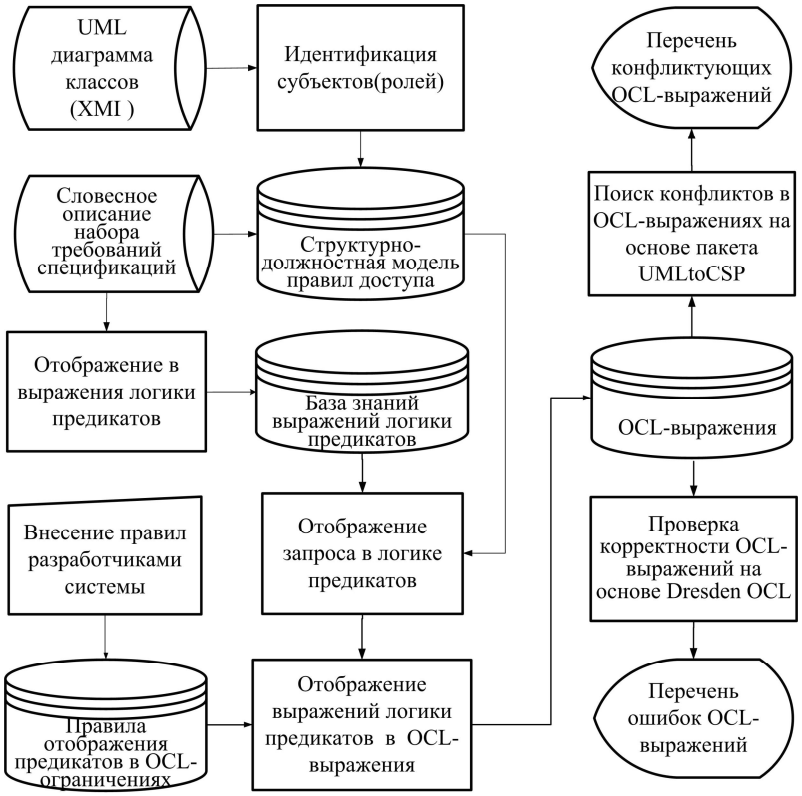

Рис. 1. Схема взаимодействия компонент программной системы

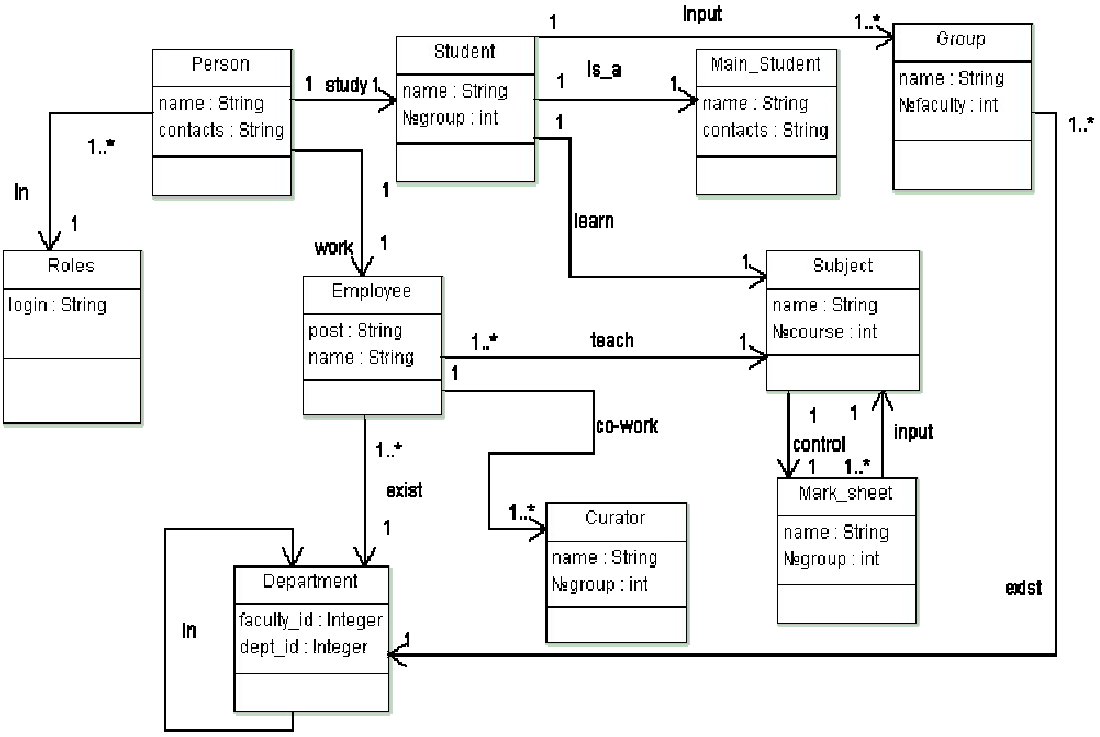

Рис. 2. Пример *UML*-диаграммы концептуальных классов системы «Электронный деканат»

Формат может использоваться для любых метаданных, если их метамодель может быть выражена с помощью *MOF (Meta-Object Facility).* Наиболее часто *XMI* применяется как формат обмена *UML*-моделями, однако может использоваться и для других языков моделирования.

Словесное описание набора требований спецификаций должно быть представлено в соответствии с

рекомендациями раздела «Контроль доступа» стандарта по информационной безопасности *ISO* 17799-2005. Рассмотрим пример словесного описания правила доступа в системе «Электронный деканат»: «Директор читает данные о личности студентов, которые учатся на факультете под его управлением». Данное описание в соответствии со стандартом должно быть утверждено и опубликовано как нормативный документ.

На рис. 1 представлены следующие БД: база знаний логики предикатов, правил отображения предикатов в *OCL*-ограничениях, *OCL*-выражений. Данные БД формализованы с помощью продукционной модели, представленной в виде структуры:

$$i = < A, B >,$$

где A – условие, B – действие.

Используя данную модель, правила принимают вид: „Если *условие*, то *действие*" .

Из рис. 1 видно, что процесс распараллеливается на: идентификацию ролей(субъектов) и преобразование словесного описания набора требований в выражения логики предикатов.

Процесс идентификации ролей разделяется на подзадачи:

- заполнение структурно-должностной модели правил доступа;

- автоматизированная идентификация субъектов (ролей).

Структурно-должностная модель правил доступа представляется в виде пятерки

$$<S, OP, O, Oc, (S,D)=(O,D)>$$

где S (*Subject*) – субъект доступа (имена ролей);

OP (*Operation*) – операция доступа (чтение, запись);

O (*Object*) – объект доступа (имена концептуальных классов);

Oc (*Object*) – конкретизирующий объект доступа (имена концептуальных классов);

$(S,D)=(O,D)$ – соответствие по структурному подразделению D в иерархической структуре организации.

Заполнение структурно-должностной модели правил доступа предложено автоматизировать с помощью алгоритма, на вход которого подается *UML*-диаграмма классов и словесное описание на естественном языке.

Алгоритм включает в себя следующую последовательность шагов.

Шаг1. Переход к субъекту. Роль «название» - S (*Subject*) является субъектом доступа к БД.

Шаг2. Определение операции, выполненной субъектом (чтения или записи). Операция OP (*Operation*).

Шаг3. Определение объекта доступа - O (*Object*) - выражается в имени концептуального класса, например, *Student*.

Шаг4. Определение конкретизированного объекта - Oc (*Object*), например, личность, и выражается в имени концептуального класса.

Шаг5. Определение связи $(S, D) = (O, D)$ субъекта (S) и объекта (O) через подразделение в иерархической структуре организации - D (*department*).

Конец алгоритма.

Для определения субъектов (ролей) с помощью *UML*-диаграммы классов можно использовать предложенные способы:

1)поиск стереотипов (*actor* - в частности) в БД, если диаграмма классов написана с использованием стереотипов;

2) перебор по именам классов или их атрибутов;

3) поиск соответствий атрибутов классов по их значению в связанных с ними БД КИС колонками таблиц;

4) анализ наследников *Subject*.

Используя перечисленные выше способы можно найти роль. При условии доступа к БД находим атрибут таблицы БД и получаем имя класса, содержащий атрибут. Этот класс представляет собой роль.

2. Процесс отображения словесного описания правил доступа в OCL-выражения

Рассмотрим пример правила доступа к документам в учебном заведение: «Директор читает данные о личности студентов, которые учатся на факультете под его управлением». Для этого правила можно сформировать следующие предикаты:

1) Если A: = «субъект является ролью: имя Роли (в данном примере: Директор)» - (S = R3) и

2) а) Если Op1: = «операция между субъектом и объектом является чтение» или б) Если Op2: = «операция между субъектом и объектом является запись» и

3) Если O: = «объект является студентами» - O4 и

4) Если C: = «таблица является департаментом» T: = D и

5) Если E: = «субъект и объект принадлежат одному и тому же департаменту» *(S & O):= D*, то Res1: = «роль Директор имеет доступ к данным о личности студентов на его факультете» Res1

Результатом создания выражения логики предикатов с использованием выражений на естественном языке является выражение вида:

(A & Op1 & O4 & C & E) -> Res1

На основании примера правила «Студент читает данные о личности студентов, которые учатся на его факультете», которое заведомо является конфиктующим, можно описать базу знаний продукционной модели, которая включает предикаты:

1) если *B*: = «субъект является ролью: имя Роли (в данном примере: студент)» - (*S* = *R6*) и

2) а) если *Op1*: = «операция между субъектом и объектом является чтение» или б) Если *Op2*: = «операция между субъектом и объектом является запись» и

3) если *O*: = «объект является студентами» - *O4* и

4) если *C*: = «таблица является департаментом» *T*: = *D* и

5) если *E*: = «субъект и объект принадлежат одному и тому же департаменту» *(S & O):= D*,

то *Res2*: = «роль Студент имеет доступ к данным о личности студентов на его факультете» *Res2*

Результатом создания выражения логики предикатов с использованием выражений на естественном языке является выражение вида:

(B & Op1 & O4 & C & E) -> Res2

Полученные выражения подаем на вход модуля отображения логики предикатов в *OCL*-выражения. Предложен алгоритм генерации OCL-выражений, который включает следующие шаги.

Шаг 1. Нахождение результата (терма) выражения логики предикатов в БД логики предикатов

Шаг 2. Результат (терм) выражения логики предикатов заменяется стандартным выражением: *Context субъект inv:*

Шаг 3. Определение субъекта.

Шаг 4. Включение символа «.»

Шаг 5. Определение операции между субъектом и объектом.

Шаг 6. Повтор шагов 4 и 5 до тех пор, пока не закончится разбираемое выражение.

Шаг 7. Получение следующего выражения. Если не найдено, то переход к шагу 10. Иначе – переход к шагу 8.

Шаг 8. Определение операции между предыдущим и последующим выражением (обращение к БД правил отображения предикатов в OCL-ограничения).

Шаг 9. Определение следующего выражения, формирование происходит в соответствии с шагами с 3-6.

Шаг 10. Определение окончания условного выражения

Шаг 11. Включение символа « -> ».

Шаг 12. Введение фразы: *select*.

Шаг 13. Включение результативного выражения в скобках «( )».

Конец алгоритма.

На основе представленных ранее примеров выражений логики предикатов в результате выполнения алгоритма сгенерированы следующие *OCL*-выражения:

1) *Context Person inv:*
Set(Roles) =
self.study.student.input.group.exist.
department.dep_id.link.department.
exist.employee -> select(post = 'Директор') and
employee.work.
person.in.roles -> select(login = 'Current_user');

2) для конфликтующего правила
Context Student inv:
self.input.group.exist.department.dep_id.
link.department.exist.
employee -> select(post = 'Студент') and
employee.work.person.in.
roles -> select(login = 'Current_user')

Дополнительно рассмотрим правила доступа к документам в учебном заведении:

1) «куратор читает данные об успеваемости студентов, которые учатся в группе *AC062* под его руководством»;

2) конфликтующее правило – «студент читает данные об успеваемости студентов, которые учатся в его группе»;

3) «староста читает данные о личности студентов, которые учатся в его группе»;

4) конфликтующее правило – «Студент читает данные о личности студентов, которые учатся в его группе».

Для указанных правил получены следующие *OCL*-выражения:

1) *Context Curator inv:*
self.co-work.employee.work.person.study.student.input.
group.exist.department.dep_id.link.department.
exist.employee->select(post='Куратор') and
employee.work.peson.study.student.learn.subject.control.
mark_sheet ->select(№group = 'AC062') and
employee.work.person.in.roles->
select(login='Current_user');

2) для конфликтующего правила –
Context Student inv:
self.co-work.employee.work.person.study.student.input.
group.exist.department.dep_id.link.department.exist.
employee->select(post = 'Студент') and
employee.work.peson.study.student.learn.subject.control.
mark_sheet -> select(№group = 'AC062') and
employee.work.person.in.roles ->
select(login = 'Current_user');

3) *Context Main_Student inv:*
self.is_a.student.input.group.exist.department.dep_id.link.
department.exist.employee ->
select(post = 'Староста') and employee.work.
person.in.roles -> select(login = 'Current_user');

4) для конфликтующего правила –
Context Student inv:
self.input.group.exist.department.dep_id.
link.department.exist.employee ->
select(post = 'Студент') and employee.work.
person.in.roles -> select(login = 'Current_user');

3. Проверка корректности и верификации OCL-выражений

Для проверки корректности *OCL*-выражений использован программный пакет *Dresden OCL*. Входным потоком данных для пакета являются файлы:

- *XMI*-файл диаграммы концептуальных классов;
- файл с *OCL*-выражениями с *ocl*-расширением.

Процесс обнаружения ошибок в *OCL*-выражениях на основе пакета *Dresden OCL* включает следующие шаги.

Шаг 1. Создание *XMI*-файла в пакете *ArgoUML*.

Шаг 2. Создание файла с *ocl*-расширением.

Шаг 3. Запуск в среде *Eclipse* модуля *DresdenOCL*

Шаг 4. Подключение к модулю созданных на шаге 1, 2 файлов.

Шаг 5. Выполнение процесса обнаружения ошибок.

Шаг 6. В случае возникновения ошибки на консоль выводится соответствующее сообщение, пример которого представлен на рис. 3.

Конец алгоритма.

Как видно из рис. 3, в ходе проверки было выявлено, что тестовые *OCL*-выражения корректны.

Следующим этапом работы, представленной на рис. 1 системы, является верификация на наличие конфликтов ограничений в *OCL*-выражениях с использованием программного пакета *UMLtoCSP*.

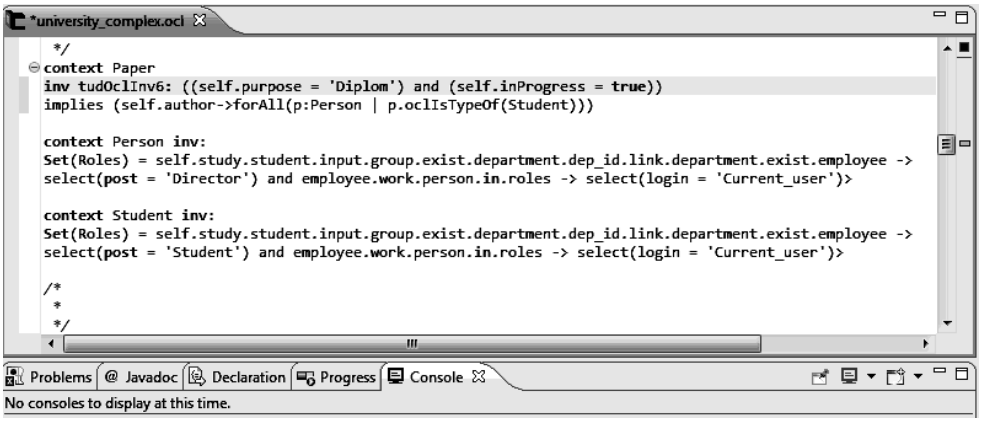

Рис. 3. Проверка корректности *OCL*-выражений

Процесс обнаружения конфликтов включает следующие шаги.

Шаг 1. Подключение входных файлов с расширениями *.xmi* либо *.uml* (файл с концептуальной диаграммой классов) и *.ocl* (файл с *OCL*-выражениями).

Шаг 2. Установка промежутка верификации *OCL*-выражений, пример экранной формы которого представлен на рисунке 6.

Шаг 3. Трансляция исходных файлов в один *XMI*-файл.

Шаг 4. Верификация, основанная на использовании *ECLiPSe Constraint Programming System*, пример экранной формы результатов которой представлен на рис. 4.

Конец алгоритма.

Рис. 4. Результат верификации

Как видно из сообщения на рисунке, система обнаруживает конфликт в ограничениях правил доступа к данным в системе «Электронный деканат». Данный конфликт создан искусственно, но также успешно система обнаружит и реальные конфликты в описании правил доступа.

Выводы

В результате разработки системы был автоматизирован процесс преобразования запроса, написанного на естественном языке в *OCL*-выражения, которые проверяются на корректность в программном пакете *Dresden OCL* и впоследствии верифицируются на наличие конфликтов в программном пакете *UMLtoCSP*. В ходе проведенных экспериментов в информационной системе «Электронный деканат» с использованием тестовых конфликтующих правил доступа к данным сотрудников факультета разработанная система обнаруживала конфликты в *OCL*-выражениях, что продемонстрировало повышение безопасности механизма автоматизированного управления доступа пользователей к таблицам БД. В дальнейшем планируется разработать механизм автоматизированного исправления обнаруженных конфликтов, а также механизм обнаружения неоднозначности в описании модели структурно-должностной иерархии доступа пользователей к данным.

Литература

1. Joshi, J. Spafford: Security models for web-based applications [Text] / James Joshi, Walid G. Aref, Arif Ghafoor, H. Eugene // Communications of the ACM (CACM). –2001. – Vol. 44. – P. 38 – 44.

2. Блажко, А.А. Автоматизированное создание правил управления доступом к данным средствами СУБД [Текст] / А.А. Блажко, Сауд Ибаа // Проблеми програмування. – 2010. – № 2–3, – С. 414 – 418.

3. On Better Understanding OCL Collections or An OCL Ordered Set Is Not an OCL Set [Text] / Fabian Büttner, Martin Gogolla, Lars Hamann, Mirco Kuhlmann // The Procs. Of MoDELS Workshops, 2009. – P. 276 – 290.

4. Richters M. Validating UML models and OCL constraints [Text] / Mark Richters, Martin Gogolla // The Procs. of The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, LNCS. – Vol. 1939, Springer, 2000. – P. 265 – 277.

5. Bisztray, D. Case study: UML to CSP transformation [Text] / D. Bisztray, K. Ehrig, R. Heckel // The Procs. of Applications of Graph Transformations with Industrial Relevance, Third International Symposium AGTIVE, 2007- Kassel, Germany, October 10-12, 2007, LNCS. – Vol. 5088, Springer 2008 – P. 540 – 565.

Поступила в редакцию 23.02.2012

Рецензент: д-р техн. наук, проф. Б.М. Конорев, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

ВИЯВЛЕННЯ КОНФЛІКТІВ У ПРАВИЛАХ ДОСТУПУ ДО РЕЛЯЦІЙНИХ БАЗ ДАНИХ З ВИКОРИСТАННЯМ МОВ UML ТА OCL

О.А Блажко, Є.В. Ляшенко, Ібаа Сауд

Стаття присвячена розробці системи автоматизованого управління доступом до даних користувачів корпоративних інформаційних систем. Мета статті – підвищення достовірності роботи системи автоматизованого управління доступу до даних. В роботі запропоновано вирішувати завдання верифікації на несуперечність правил доступу до даних крізь їх зображення у вигляді OCL-обмежень. Це дозволило використовувати відкритий програмний пакет DresdenOCL, перевіряючий коректність виразів, та пакет UMLtoCSP, виявляючий конфлікти між OCL-обмеженнями правил.

Ключові слова: корпоративні інформаційні системи (КІС), база даних (БД), OCL (Object Constraint Language), UML (Unifed Model Language), XMI (XML Metadata Interchange), MOF (Meta-Object Facility).

CONFLICTS DETECTION IN ACCESS RULES TO THE RELATIONAL DATABASES WITH USING UML AND OCL LANGUAGES

A.A. Blazhko, E.V. Liashenko, Ebaa Saoud

The article is devoted to the system development of automated control users access to data in corporate information systems. A purpose of the article is an increase of authenticity of work of the system of the automated management of access to to to information. In the work were proposed to solve the verification problem of access rules consistency to data through their representation in the OCL-constraints form. This was allowed to use the open software package DresdenOCL, which is checked the correctness of expressions, and package UMLtoCSP, which is detected conflicts between OCL-constraints rules.

Key words: Corporate information systems (CIS), database (DB), OCL (Object Constraint Language), UML (Unifed Model Language), XMI (XML Metadata Interchange), MOF (Meta-Object Facility).

Блажко Александр Анатольевич – канд. техн. наук, доцент кафедры системного программного обеспечения Одесского национального политехнического университета «ОНПУ», Одесса, Украина.

Ляшенко Елизавета Владимировна – магистр кафедры системного программного обеспечения Одесского национального политехнического университета «ОНПУ», Одесса, Украина.

Ибаа Сауд – аспирант кафедры системного программного обеспечения Одесского национального политехнического университета «ОНПУ», Одесса, Украина.