УДК 004.75:510.63

М.Н. Дубовенко, В.М. Белов

Некоторые аспекты разработки социальных приложений, адаптированных к социальным сетям в задачах Интернет-зависимости

Показана возможность адаптации социального приложения в социальные сети «Вконтакте», «Мой мир» и «*Facebook*» путем расширения структуры приложения слоем агентов инициализации.

The possibility of the adaptation of social application into social network «Vkontakte», «Moy Mir» and «Facebook», by extending the structure of application with agent layer is shown.

Показано можливість адаптації соціального додатку в соціальні мережі «Вконтакте», «Мой мир» і «*Facebook*» шляхом розширення структури додатку агентами ініціалізації.

Введение. Современные тенденции развития Интернета свидетельствуют об актуальности социальных сетей. Последние, в частности, ориентированы на интеграцию всех возможных Интернет-ресурсов, доказательством чего есть рост количества распределенных приложений в социальных сетях и активная разработка и усовершенствование *API* (*Application Programming Interface*). Существующие *API* социальных сетей, в рамках внутренних правил, дают возможность работать как со своими пользователями, так и с их данными. Интернет-ресурсы или распределенные приложения, построенные на взаимодействии с *API* социальных сетей, часто называют *социальными приложениями*. А эти приложения как одно из ответвлений распределенных информационных систем и как новое перспективное направление начали разрабатывать три–четыре года назад.

Социальная сеть как ресурс предоставляет механизмы для интеграции социальных приложений, что, собственно, вызывает особый интерес, в том числе и к разработке социальных приложений научно-исследовательского направления. Некоторые научные направления требуют широкой и разноплановой аудитории для исследования, например, такие, как изучение особенностей психосоциальной сферы человека, зависимого от Интернета [1]. Как известно, большинство социальных сетей ориентировано на разные возрастные группы, поэтому возникает задача разработки программного механизма, способного обеспечить работоспособность социального приложения в нескольких социальных сетях для охвата всех доступных возрастных категорий. Решение данной задачи позволит собрать единую базу экспериментальных данных и упростит процесс исследования.

Анализ спецификаций и подходов разработки социальных приложений

У каждой социальной сети есть свои спецификации, излагаемые в документациях по *API*. Проведем анализ спецификаций социальных сетей «Вконтакте» [2], «Мой мир» [3] и «*Facebook*» [4]. Для удобства рассмотрения выделим в спецификациях две линии, которые прослеживаются в отмеченных социальных сетях: *организационная* и *функциональная.* При этом под *организационной линией* здесь понимают правила и способы интеграции в социальную сеть стороннего Интернет-ресурса или распределенного приложения, а под *функциональной линией* – особенности использования того или иного *API*.

Синтезируем по *организационной линии* следующие общие положения в обозначенных социальных сетях:

1. Интеграция социальных приложений в отмеченные выше социальные сети на данный момент возможна только двумя способами: *flash-интеграция* либо *iframe-интеграция*. В первом случае каркас социального приложения хранится на сервере социальной сети, но не исключа-

ется возможность общения *flash*-приложения с сервером разработчика, с использованием, например *AJAX*; во втором – только на сервере разработчика.

2. Каждая социальная сеть предоставляет систему управления метаданными и администраторами приложения, а также все стандартные статистические инструменты и показывает доступность приложения в онлайн-каталоге социальной сети.

3. При регистрации приложения создаются секретные ключи, необходимые при обмене данными.

4. Для настройки связи с *API* в клиентский код социального приложения добавляются скрипты.

5. Социальное приложение и социальные сети могут общаться как в синхронном, так и асинхронном режимах.

6. Во всех социальных сетях существует механизм проверок приложения. Каждая социальная сеть предоставляет ряд правил, которые необходимо соблюдать при подаче приложения на модерацию для опубликования и при функционировании уже в открытом доступе.

7. Каждое социальное приложение, которое претендует на публикацию в социальной сети в открытом доступе должно использовать *API*.

8. Регистрация разработчика, в сравнении с мобильными приложениями типа *iOS*, *Android* и другие не требует специальной процедуры регистрации аккаунта, поскольку достаточно быть пользователем социальной сети.

Общие функциональные особенности API социальных сетей. Каждая социальная сеть предоставляет несколько типов *API*, которые условно можно разделить на *клиентские* (*JavaScript API* и др.) и *серверные* (*Rest API* и др.), а также базовые готовые решения в виде *framework*'ов для мобильных платформ. Рассмотрим некоторые положительные и отрицательные стороны с учетом того, что социальное приложение – ресурс, на который потенциально возложена высокая нагрузка со стороны пользователей.

Изначально существовали версии *API*, ориентированные на сторону сервера разработчика, например *Rest API*, рассчитанные на серверные языки социального приложения *php* или *asp*. Общение между серверами разработчика и социальной сети предусмотрено протоколом *http*. При высокой нагрузке возникают некоторые помехи в связи с обработкой транзакций на обеих сторонах. Этот недостаток можно компенсировать путем увеличения серверов, распределения задач, оптимизации кода, а также расширения канала трафика, что финансово затратно. Поэтому в последние годы начали разрабатывать *API* под клиентские технологии, например, *JS API*, *Open API* для минимизации нагрузок на серверы разработчика и увеличения скорости загрузки социальных приложений. Практика показывает, что в данной ситуации также наблюдаются недостатки, поскольку часто приходится выносить параметры социального приложения, которые нежелательно раскрывать, на клиентскую сторону. Кроме того, разработка социального приложения, основанная на клиентском типе A*PI*, может стать достаточно громоздкой. Клиентские *API* в приложениях, которые постоянно работают с базами данных, не очень удобно использовать, что вынуждает искать иные гибридные решения.

Помимо *API* для социальных приложений, существуют сервисы социальных виджетов для Интернет-ресурсов. Это: аутентификация средствами социальных сетей, *sharing* и перечень социальных блоков групп и сообществ, комментирования, опросов и других, которые при необходимости могут быть интегрированы и в социальные приложения.

Постановка задачи

Из изложенного видим, что принципы интеграции социальных приложений в социальные сети сводимы к одному подходу, *API* в рассмотренных социальных сетях существуют как для клиентской, так и для серверной стороны, и это дает возможность создания единой платформы, адаптируемой к нескольким социальным сетям на определенном типе *API*. Для этого требуется решить ряд задач, где первостепенными, по мнению авторов, есть:

- разработка принципиальной схемы социальных приложений;

• разработка механизмов аутентификации социальных сетей и их пользователей социальным приложением на основе входящих параметров.

Структура социального приложения

В текущих версиях спецификаций не рассматривается вопрос предпочтительной структуры социального приложения, а для разработки механизма идентификации различных социальных сетей и пользователей он достаточно актуален. Выделим особенности:

• на основу приложения должен поступать авторизированный и аутентифицированный пользователь, поскольку он уже прошел данную процедуру при входе в социальную сеть;

• представление данных зависит от выделенного социальной сетью фрейма, размеры интерфейса которого ограничены;

• интеграция социального приложения в несколько социальных сетей подразумевает четкое отделение представления данных от модели предметной области в связи с различными правилами формирования интерфейсов, а также необходимость механизма распределения задач, поступивших со стороны пользователя.

В литературе описывают двух-(клиент-серверная), трех-(*Model-View-Controller*) [5] и пятислойную архитектуру [6]. Очевидно, что двухслойная архитектура здесь не подходит, исходя из выделенных особенностей. Трехслойная *MVC*-архитектура достаточно близка для распределенных приложений, исходя из [5], но этот шаблон может работать после автоматической идентификации пользователя и социальной сети. Пятислойная архитектура, как показано в [6], состоит из слоев представления данных, контроллеров, предметной логики, сопоставления данных и источника данных. Однако внимание авторов [6] сфокусировано на разработке социальной сети, а не на постоянном общении между социальной сетью и социальным приложением, и для социального приложения такая архитектура перегружена. Таким образом, социальное приложение может базироваться на модифицированной *MVC*-модели, расширенной слоем агентов, который берет на себя задачи идентификации социальных сетей и пользователей. Рассмотрим принципиальную схему социального приложения на рис. 1. Отметим, что для большей иллюстративности в схеме социального приложения не представлена социальная сеть «*Facebook*».

Агенты идентификации связаны прямой связью с *API* социальных сетей, а в социальном приложении – только с контроллерами. В момент загрузки социального приложения поступает поток параметров как от серверов по *API*, так и через сессии и *uri* на фрейм, в который встроено социальное приложение. Задачи этих агентов – обеспечить корректность генерации социального приложения в социальную сеть для данного пользователя, и выделяется *агент идентификации и аутентификации социальных сетей* (АИАСС) и *агент идентификации пользователя* (АИП). В таком виде рассмотренная структура позволяет интегрировать социальное приложение в несколько социальных сетей и базироваться на едином программном механизме и единой базе данных на уровне модели предметной области, а также способствует упрощению получения экспериментальных данных.

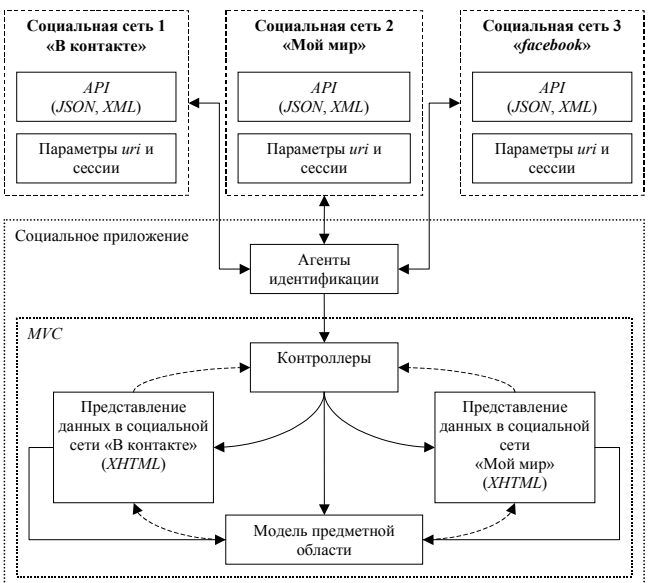

Рис. 1. Принципиальная схема социального приложения (стрелками со сплошными линиями показаны прямые связи, пунктиром – косвенные)

Агенты идентификации

Параметры пользователя и социальной сети. В процессе взаимодействия каждого поль-

зователя p с социальным приложением передается перечень данных, которые, проходя через некоторую среду S (социальную сеть), формируют первичный массив D [7]. Обозначим его как $D=\left\langle D_1,...,D_j,D_{1|p},...,D_{j|p},D_{1|S},...,D_{k|S}\right\rangle$. В него входят параметры среды и пользователя. Иные параметры клиента и технического устройства захода, не учитываются. Для иллюстрации данного подхода приведем некоторые из параметров среды социальной сети «В контакте»:

– ***api_url*** – адрес сервиса *API*, по которому осуществляются запросы;

– ***api_id*** – *id* запущенного приложения;

– ***sid*** – *id* сессии для осуществления запросов к *API*;

– ***secret*** – ключ, необходимый для осуществления подписи запросов к *API*;

– ***viewer_id*** – *id* пользователя, который просматривает приложение;

– ***auth_key*** – ключ, необходимый для авторизации пользователя на стороннем сервере;

– ***api_result*** – результат первого *API*-запроса, который выполняется при загрузке приложения;

– ***api_settings*** – битовая маска настроек текущего пользователя в данном приложении и др.

Некоторые параметры среды социальной сети «Мой мир»:

– ***app_id*** – идентификатор приложения;

– ***session_key*** – идентификатор сессии;

– ***oid*** – идентификатор пользователя, установившего приложение;

– ***vid*** – идентификатор пользователя, запустившего приложение;

– ***ext_perm*** – пользовательские настройки приложения;

– ***sig*** – подпись параметров и др.

Из всего множества параметров на стадии идентификации пользователя и среды интерес представляют только два: *идентификатор среды* и *идентификатор пользователя*. Возможны три комбинации их взаимодействия: данные параметры могут существовать одновременно, не одновременно либо какой-то один из параметров не существует вообще. Отсюда следует, что первая задача – это выяснение, существуют ли эти два параметра, и как в дальнейшем на них реагировать.

Агент идентификации и аутентификации социальных сетей. В отличие от существования идентификатора пользователя, идентификатор среды можно контролировать. Это реализуется путем добавления в адресную строку параметра, например *domain*, в выделенной системе управления приложением в каждой социальной сети.

Пусть социальное приложение имеет некое доменное имя, например, *diagnostics-center.com*. Ссылаясь через технологию *iframe* из социальных сетей на домен, вводится сразу параметр *domain*, и выглядит так: *diagnostics-center.com*? *domain=vkontakte*.

Агент, получая методом *GET* значение параметра *domain*, который в приведенном примере равен *vkontakte*, осознает информацию о среде – социальная сеть «В контакте». Таким образом, формируется решение первой задачи – идентификация и аутентификация среды, что представлено в виде схемы алгоритма на рис. 2.

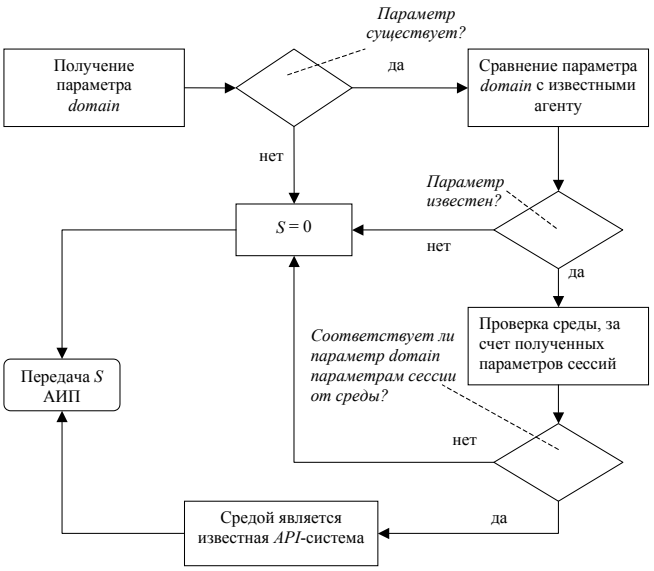

Рис. 2. Схема алгоритма агента идентификации и аутентификации среды

Параметр *domain* в социальном приложении, исходя из рассматриваемых спецификаций, принимает значения: *website*, *vkontakte*, *mailru*, *facebook*. Обозначим их как $D'_{1|S}=\left\langle D'_{1|0},\ D'_{1|1},\ D'_{1|2},\right.$

$= D'_{1|3}\rangle$, а получаемый параметр *domain* – как D_1. Сопоставление $D'_{1|S}$ с получаемым параметром D_1 может принимать значения ноль либо единица. Таким образом, идентификация среды будет происходить путем перебора до момента, пока $D_1 \wedge D'_{1|S} = true$, и принимаемым решением будет значение S в интервале [0–3]. В случае ошибки – ситуации неопределенности, когда любое из значений $D_1 \wedge D'_{1|S} = false$, $S=0$, логическим решением АИАСС будет то, что *API*-система средой не является. В случае, когда $1 \le S \le 3$, АИАСС уже идентифицировал среду и принимает параметры сессий $D_{2|S}$ со стороны социальных сетей для аутентификации.

Существующие параметры в системе соответственно хранятся в виде $D'_{2|S} = \langle D'_{2|1}, D'_{2|2}, D'_{2|3}, D'_{2|4}\rangle$. Для $S=1$ значение параметра: $D'_{2|1}$ = *api_result*, для $S=2$ $D'_{2|2}$ = *session_key*, для $S=3$ $D'_{2|3}$ = *signed_request*, для $S=0$ $D'_{2|0}=\varnothing$. Под аутентифицированной средой АИАСС понимает выполнение условия $D_{2|S} \wedge D'_{2|S} = true$, где $D_{2|S} \notin \varnothing$ и S получит текущее значение, в любых других – S присваивается значение ноль. Таким образом, происходит двойная проверка по взятому параметру из адресной строки и полученной сессии, а сходимость говорит о корректности поведения АИАСС, что необходимо на случай попыток несанкционированного доступа.

Агент идентификации пользователя. После того, как АИАСС определил среду, начинается следующее действие – определение пользователя последующим *агентом идентификации пользователя*. Схема алгоритма представлена на рис. 3.

Получив значение среды *S*, для АИП становится понятен последующий способ аутентификации пользователя. Для случая, когда $S=0$, работает стандартная система авторизации пользователя путем заполнения формы логина и пароля. Проходя аутентификацию, идентификатор и часть личных данных пользователя загружаются из базы данных, передаваемых далее. При $S \ne 0$ авторизация средствами формы не требуется, поскольку пользователь уже известен как участник социальной сети. В зависимости от значения *S* АИП получает значение параметра идентификатора $D_{1|p}$ данного пользователя из социальной сети из переданной параметрической строки методом *GET* либо использовав значение из полученной сессии (в зависимости от социальной сети). Имея идентификатор, АИП делает запрос в базу данных социального приложения относительно существования параметров данного пользователя, и если результатом запроса будет нулевая величина, АИП формирует запрос в социальную сеть и вступает в общение с ней относительно данного пользователя.

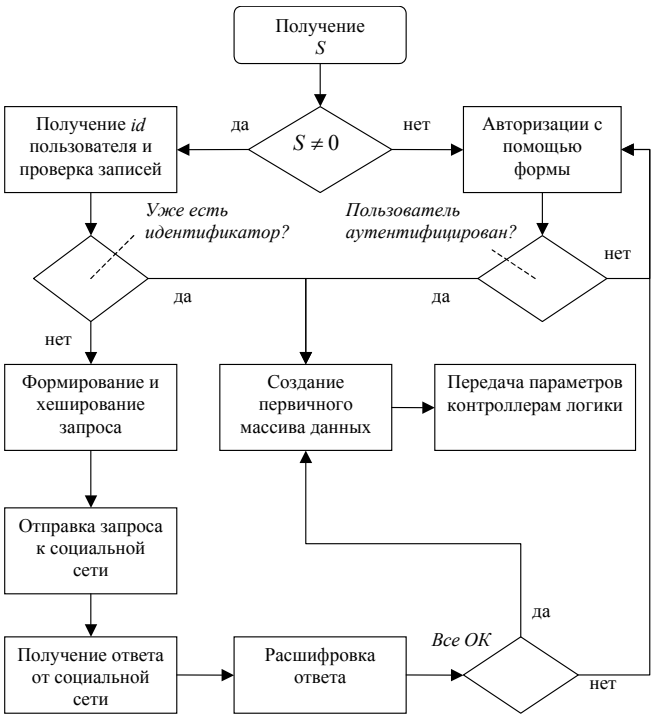

Рис. 3. Схема алгоритма агента идентификации пользователя

Формирование запроса к социальной сети происходит в виде адресной строки на *web*-сервере социального приложения. В адресную строку входят: *адрес социальной сети*, на который поступит запрос, *запрашиваемые параметры*, *метод API*, который должен реализовать получение данных, и в некоторых случаях *сигнатура*. Последняя формируется путем хеширования названий параметров, которые необходимо получить, и ключи социального приложе-

ния. Далее на серверной стороне социального приложения по протоколу *http* передается строка, на ответ которой ожидается входящий ответ. Ответ принимается в форматах *xml* или *json*.

В запрос относительно пользователя заложены параметры, необходимые для последующей работы. Такими параметрами чаще других бывает *возраст* (день, месяц, год рождения) $D_{2|p}$, $D_{3|p}, D_{4|p}$ и *пол* $D_{5|p}$. Выбор параметров зависит от поставленной задачи.

Далее в примере приведен фрагмент листинга правила АИП по формированию, отправке, получению и обработке запроса в социальную сеть для аутентификации пользователя.

Пример. Получение пользовательских данных из социальной сети средствами языка *php* при $S = 2$.

```
// Параметры запроса
$session_key = $_GET['session_key'];
$app_id = "518***";
$private_key =
"5743f55ec5667ed5aeb24c1e5f0***";
$secret_key =
"5dbdc6559081d7d546a1f97645bd6***";
$format = "json";
$uids = $_GET['vid'];
$method = "users.getInfo";
// Хеширование параметров и ключей
$siggetIn-
fourl=md5("app_id=".$app_id."format=".$format
."method=".$method."secure=1session_key=".$se
ssion_key."uids=".$uids.$secret_key);
// Формирование запроса для сервера социаль-
ной сети
$getIn-
fourl="http://www.appsmail.ru/platform/api?ap
p_id=".$app_id."&format=".$format."&method=".
$method."&secure=1&session_key=".$session_key
."&sig=".$siggetInfourl."&uids=".$uids;
// Отправка запроса и получение ответа
$getInfo = file_get_contents($getInfourl);
// Декодирование ответа
return json_decode($getInfo, true);
// Далее создается массив по выбранным па-
раметрам, и с данными
// можно уже работать
```

Известно, что в любой вычислительной операции существует вероятность возникновения ошибок. Причины могут быть разные: неправильное значение параметра, нарушение сигнатуры, технический сбой работы одного из серверов, в результате чего запрос не был обработан. Поскольку риск предоставления пользователю ложной информации должен быть исключен, в схеме алгоритма на рис. 3 предусмотрено, что в случае возникновения ошибки, АИП действует по сценарию $S = 0$, избегая критической ситуации ложной авторизации.

Организация интерфейса. Получив параметры пользователя и социальной сети, можно считать, что идентификация в целом завершена и данные передаются контроллеру, который отвечает за организацию базового интерфейса. Параметры вывода интерфейса можно сгруппировать в виде правил, и, в зависимости от значения S, формировать как массив $I = \langle I_{1|s}, I_{2|s}, I_{3|s}, I_{4|s}, I_{5|s}, I_{6|s} \rangle$, в который закладываются следующие параметры, используемые как переменные подстановки или *View*-шаблон:

$I_{1|s}$: ширина интерфейса;

$I_{2|s}$: ширина рабочей части интерфейса;

$I_{3|s}$: ширина рабочей части интерфейса со средним отступом;

$I_{4|s}$: ширина рабочей части интерфейса с максимальным отступом;

$I_{5|s}$: количество выводов запросов;

$I_{6|s}$: параметр перехода (для *AJAX* и *pagination*).

Эти параметры необходимы для корректности отображений выводов на уровне представления данных. Сгенерировав интерфейс, каркас социального приложения для данного пользователя в данной среде создан. Далее приложение готово к выполнению задач, заложенных в основу его предметной логики.

Практическое применение. По данной технологии реализован программный исследовательский комплекс, в задачи которого входит изучение психосоциальной сферы пользователей с Интернет-зависимостью. Интеграция программного исследовательского комплекса в социальные сети «В контакте», «Мой мир», «*Facebook*» позволила получить обширную экспериментальную базу данных, в которой зафиксированы данные диагностики на Интернет-зависимость около 30 тыс. чел. различных возрастных категорий за 2010–2011 годы [1, 8].

Окончание на стр. 53

Заключение. Добавление агентов идентификации социальных сетей и пользователей в расширенную *MVC*-архитектуру распределенных приложений позволяет интегрировать социальные приложения в несколько социальных сетей одновременно и взаимодействовать с их пользователями без дополнительной разработки программных систем. Гибкость такой архитектуры позволит в дальнейшем расширять бизнес-логику социального приложения новыми задачами, а также адаптировать его под мобильные платформы.

Социальные приложения как направление распределенных информационных систем дают инструментальную возможность масштабным исследованиям и мониторингам в Интернете.

Управление социальными приложениями можно развивать до корпоративного уровня для решения совместных научных задач при взаимодействии двух и более организаций независимо от территориального расположения.

1. *Белов В.М.*, *Дубовенко М.Н.* К проблеме Интернет-зависимости // Кибернетика и вычислительная техника. – 2010. – **161**. – С. 53–60.
2. *http*://*vkontakte.ru*/ *developers.php#devstep*2
3. Как создавать социальные приложения для *Mail.Ru*. – *http*://*api.mail.ru*/*docs*/*guides*/*social-apps*/
4. *Apps* on Facebook.com. – *http*://*developers.facebook.com*/*docs*/*guides*/*canvas*/
5. *Соловйова К.О., Мовчан В.В*. Розробка моделі програмного засобу візуалізації мережевих структур. – *http*://*www.nbuv.gov.ua*/*portal*/*natural*/*vcpi*/*Sa*/2010_9/*statya*25_9.*pdf*
6. *Карякин О.И*., *Кочетова Е.О*., *Щербак С.С.* Технологии разработки распределенных приложений и их применения в социальных сетях // Нові технології. Наук. вісн. Кременчуцького ун-ту економіки, інформаційних технологій і управління. – 2008 – № 4. – С. 70–77.
7. *Дубовенко М.Н*., *Белов В.М.* Информационная модель взаимодействия программного исследовательского комплекса с внешними средами в Интернете / Мат. науч.-тех. шк.-сем., ФМШ Жукин, 21–24 июня 2011 г. – С. 33–35.
8. *Дубовенко М.Н*., *Белов В.М.* Концептуальный алгоритм классификации психологических проблем пользователей на основе приложений в социальных сетях // Кибернетика и вычислительная техника. – 2011. – Вып. 165. – С. 3–15.

Поступила 12.10.2011
Тел. для справок: (044) 503-9565 (Киев)
E-mail: *dep*150*@ukr.net*