УДК 004.4

А.А. Ломов, Д.Ж. Корзун

ОПЕРАЦИЯ ПОДПИСКИ ДЛЯ ПРИЛОЖЕНИЙ В ИНТЕЛЛЕКТУАЛЬНЫХ ПРОСТРАНСТВАХ ПЛАТФОРМЫ SMART-M3

Ломов А.А., Корзун Д.Ж. **Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3.**
Аннотация. В данной работе рассматривается операция подписки, используемая при разработке многоагентных приложений интеллектуальных пространств на платформе Smart-M3. Систематизируются алгоритмы базовой операции подписки, выполняемые на стороне агента. Базовая операция подписки работает на уровне RDF-модели представления данных. На основе выполненной систематизации и с учетом модельно-ориентированного подхода к разработке программ предлагаются новые типы подписки, оперирующие на уровне OWL-модели. Предлагаемые типы подписки реализованы в инструментарии SmartSlog, что позволяет программировать агентов в высокоуровневых онтологических терминах (класс, свойство, индивид).
Ключевые слова: интеллектуальные пространства, Smart-M3, подписка, онтологии, RDF, OWL, SmartSlog.

Lomov A.A., Korzun D.G. **Subscription operation for applications in smart spaces of Smart-M3 platform.**
Abstract. The paper considers the subscription operation for multi-agent application development on Smart-M3 platform. We systematize the algorithms of the basic subscription operation that Smart-M3 agents implement. The basic subscription operation uses the RDF representation model. Based on the systematization and taking into account the model-driven approach, we propose new subscription types that apply the OWL representation model. The proposed subscription types are implemented in SmartSlog SDK, resulting in compact application code that operates with high-level ontological terms (class, property, individual). **Keywords:** smart spaces, Smart-M3, subscription, ontology, RDF, OWL, SmartSlog.

1. Введение. Платформа Smart-M3[1] предназначена для создания многоагентных распределенных приложений, взаимодействующих в общем "интеллектуальном пространстве" (ИП, от англ. "smart space") [1, 2]. Создается разделяемое хранилище динамической информации [3], реализующее среду "повсеместных вычислений" (от англ. "ubiquitous computing"). Информация представлена по модели RDF [4] (от англ. "resource description framework"), и доступ управляется семантическим информационным брокером (от англ. "semantic information broker", далее — SIB).

[1] Открытый код доступен на `http://sourceforge.net/projects/smart-m3/`

Приложение строится как набор процессоров знаний (от англ. “knowledge processor”, далее — KP), каждый из которых реализует программного агента. Агенты могут работать на разнородных вычислительных устройствах (сенсоры, бытовая техника, мобильные устройства, серверные ЭВМ и пр.), взаимодействуя посредством разделения данных в ИП. Доступ KP к ИП происходит через SIB по протоколу SSAP (от англ. “smart space access protocol”) используя такие базовые операции, как “вставка”, “удаление”, “обновление” и “подписка”, аргументами которых выступают RDF-триплеты. Подписка определяет специальный вид запроса — постоянный запрос. При изменении в ИП данных, определяемых таким запросом, KP-подписчик получает уведомление от SIB.

Для Smart-M3-приложений можно использовать методы автоматизированной модельно-ориентированной разработки, если структура предметной области задается онтологией [5]. Такой подход реализуется в инструментарии SmartSlog[2]. При работе с онтологиями используется язык OWL (от англ. “web ontology language”) — стандарт представления знаний в семантическом веб [4, 6]. Соответствующие возможности доступны программисту через интерфейс прикладного программирования (API) онтологической KP-библиотеки, генерируемой по OWL-онтологии.

Цель настоящей работы состоит в систематизации алгоритмов базовой операции подписки платформы Smart-M3 (на уровне RDF) и разработке высокоуровневых типов подписки (на уровне OWL). Последние работают с онтологическими классами и со свойствами индивидов. На стороне KP-подписчика проверка уведомлений проводится как синхронно, так и асинхронно. Предлагаемые типы подписки реализуются в инструментарии SmartSlog сведéнием к алгоритмам базовой операции подписки. Низкоуровневые детали такого сведéния скрыты от программиста, позволяя оперировать в коде программы терминами предметной области из заданной OWL-онтологии. Работа является развитием [7].

2. Базовая операция подписки. Платформа Smart-M3 реализует шаблон проектирования “издатель-подписчик”, определяющий зависимость “один ко многим” между взаимодействующими объектами. При изменении состояния объекта все зависящие от него объекты оповещаются [8]. Шаблон применяется в слабосвязан-

[2]Открытый код доступен на `http://sourceforge.net/projects/smartslog/`

ных распределенных системах [9]. Используются три группы объектов: 1) подписчики, заинтересованные в получении информации, 2) издатели, публикующие информацию и 3) сервис-посредники, уведомляющие о событиях. Такой подход позволяет создавать проактивные и контекстно-ориентированные приложения [10–12].

В платформе Smart-M3 подписчиками и издателями являются KP, а сервис-посредниками — брокеры SIB. Базовая операция использует набор T-шаблонов для определения подписываемых данных. Каждый T-шаблон — это триплет, где допускаются маски для субъекта, предиката или объекта. Набор T-шаблонов хранит SIB для каждой подписки. Если соответствующие шаблону данные кем-либо изменяются в ИП, то подписчики получают уведомления.

Подписка позволяет синхронизировать локальные данные KP с содержимым ИП. Выделим две стадии синхронизации: 1) начальная — при подписании KP получает все триплеты из ИП, удовлетворяющие заданным T-шаблонам и 2) во время уведомления — KP получает те триплеты, которые были изменены в ИП.

Пример T-шаблонов и соответствующее им содержимое ИП приведены на рис. 1. T-шаблон a определяет конкретный триплет. В нем не используется маска, и уведомления будут приходить при помещении в или удалении из ИП соответствующего триплета. T-шаблоны b, c и d используют маску, позволяя отслеживать изменения набора триплетов. Так, T-шаблоны c и d соответствуют любым триплетам с предикатом surname или субъектом Person2.

Базовая операция подписки состоит из четырех основных шагов, выполняемых на стороне KP (см. блок-схему на рис. 2).

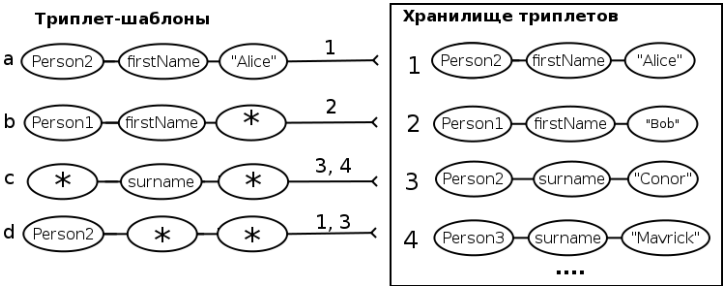

Рис. 1: Отображение T-шаблонов в RDF-хранилище. T-шаблон a определяет триплет 1; b — 2; c — набор $\{3, 4\}$; d — $\{1, 3\}$.

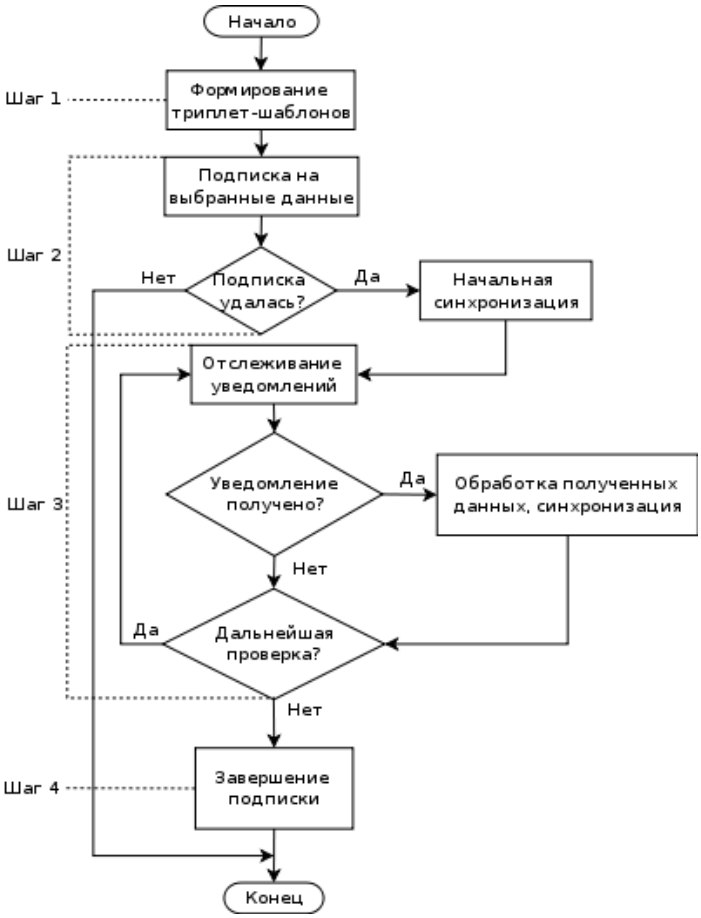

Рис. 2. Блок-схема базовой подписки платформы Smart-M3.

1. Формирование Т-шаблонов.
2. Подписка на выбранные данные, начальная синхронизация, начинается сессия подписки.
3. Отслеживание уведомлений от ИП, синхронизация данных.
4. Завершение подписки (отписка), закрытие сессии подписки (SIB высылает KP уведомление об отписке).

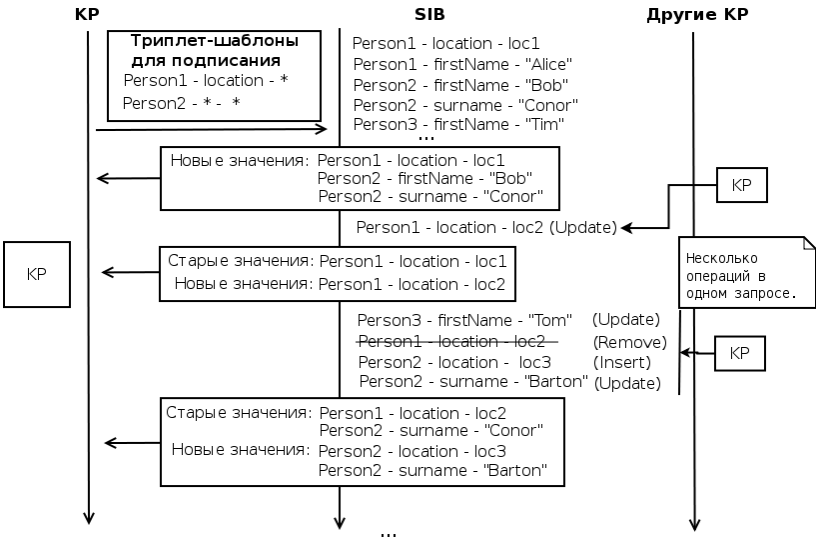

Рис. 3. Пример работы базовой подписки (уровень RDF).

Пример работы базовой подписки показан на рис. 3. Сформированные два Т-шаблона отслеживают местоположение человека Person1 и изменение данных о Person2. При начальной синхронизации КР получает все триплеты из ИП, удовлетворяющие Т-шаблонам. В ходе подписки поступают уведомления об изменениях, совершаемых другими КР. Каждое уведомление содержит списки с удаленными и добавленными в ИП триплетами, используя которые КР проводит локальную синхронизацию.

Обновление в ИП — это замена старого триплета на новый. В примере на рис. 3 при изменении местоположения человека триплет Person1–location–loc1 удаляется, а триплет Person1–location–loc2 добавляется. Тем самым, КР должен находить соответствие между локальными триплетами и триплетами из уведомления.

Проверка поступления уведомлений и обработка могут проводиться многократно в течении сессии подписки. В зависимости от реализации шага 3 (рис. 2) появляются разные типы подписки. Рассмотрим синхронную и асинхронную проверку уведомлений.

3. Синхронная подписка. При синхронной проверке текущий поток выполнения программы на стороне КР блокируется 1) до

получения уведомления или 2) по таймеру. В первом случае, КР ожидает изменения данных в ИП. Во втором случае, блокировка прекращается, если за указанное время не получено уведомления.

На стороне КР синхронная подписка доступна через интерфейс KPI (от англ. "KP interface"), см. список KPI в [5]. Программа КР вызывает соответствующую KPI-функцию. При получении уведомления в КР передаются только списки измененных триплетов, без синхронизации данных. Код КР для подписки показан на листинге 1 (на основе интерфейса C_KPI[3]).

Обозначим через Δt интервал повторной проверки подписки — время от окончания работы (**) до нового вызова KPI-функции (*), как показано на рис. 4. На Δt проверка поступления уведомлений программой КР не выполняется. В то же время SIB на Δt может отправлять уведомления. Эти уведомления накапливаются в буфере сетевого сокета, реализуемого ОС устройства, на котором запущен КР. Такие уведомления КР сможет обработать при очередном вызове KPI-функции синхронной подписки. Размер сетевого буфера ограничен, поэтому возможна потеря уведомлений, поступивших на Δt. Длина Δt определяется программной логикой КР — увеличение частоты вызывов KPI-функции уменьшает Δt.

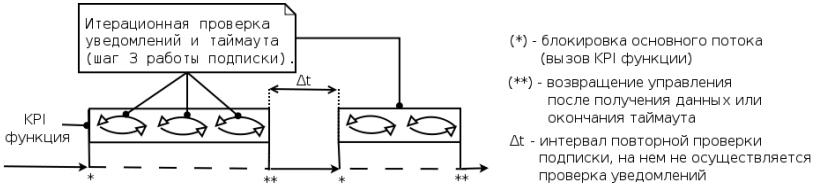

Рис. 4. Блокировка при синхронной подписке.

4. Асинхронная подписка. Отслеживание уведомлений выполняется в фоновом режиме, что требует создания дополнительных потоков (помимо потока программы КР) и применения функций обратного вызова (callback).

4.1. Асинхронная подписка с одним потоком. Выделяется один фоновый поток для обслуживания всех подписок $i = 1, 2, \ldots, n$ на стороне КР. Поток проверяет подписки по круговому циклу (round robin). Проверка отдельной подписки реализуется как син-

[3]Открытый код доступен на http://sourceforge.net/projects/smartslog/

```
1: // T-шаблон для местоположения: Person1–location–*
2: ss_triple_t triple_rqst = NULL;
3: ss_add_triple(&triple_rqst, "Person1", "location",
       SS_RDF_SIB_ANY, SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);

4: // Подписываемся
5: ss_triple_t *triple = NULL;
6: ss_subs_info_t subs_info;
7: if (ss_subscribe(ss_info, &subs_info, triple_rqst,
   &triple) != 0) {
8:     printf("Failed to subscribe.");
9:     ...
10: }
11: ss_delete_triples(triple_rqst);

12: // Поступление уведомлений и синхронизация данных
13: ss_triple_t *n_val = NULL;
14: ss_triple_t *o_val = NULL;
15: int status;
16: while(1) {
17:     status = ss_subscribe_indication(
18:     ss_info, &subs_info, &n_val, &o_val, 1000);
19:     if (status == 1) {  // Получено уведомление
20:         // Работа с триплетами и их удаление, если не нужны
21:         ss_delete_triples(n_val);
22:         ss_delete_triples(o_val);
23:         n_val = NULL; o_val = NULL;
24:     }
25:     ...
26: }
27: // Завершение подписки
28: ss_unsubscribe(ss_info, &subs_info);
```

Листинг 1. Пример кода на языке C для синхронной подписки на триплеты (используется интерфейс C_KPI).

хронная проверка с таймером. При поступлении уведомления процесс проверки прерывается и полученные данные обрабатываются. Затем проверка продолжается, начиная со следующей подписки.

Такой вариант ориентирован на вычислительные устройства, где можно использовать лишь ограниченное число потоков.

На рис. 5 показан пример работы фонового потока для $n = 5$ подписок. При проверке подписок $i = 1, 2$ уведомлений не обнаружено. При проверке подписок $i = 3, 4$ уведомления поступили.

Интервал повторной проверки для подписки j определяется как

$$\Delta t_j = \sum_{i=1, i \neq j}^{n} \left(t_i^{\text{ch}} + t_i^{\text{pr}} \right) + t_j^{\text{pr}}, \tag{1}$$

где n — число подписок, t_i^{ch} — время на проверку подписки i, t_i^{pr} — время обработки уведомления для подписки i. Пример для (1) показан на рис. 6.

Параметры в (1) допускают управление на стороне KP. Объединение нескольких подписок в одну уменьшает n. Уменьшение таймаута проверки отдельной подписки уменьшает t_i^{ch}. Значения t_i^{pr} определяются реализацией обработки уведомлений в используемом интерфейсе KPI или KP-библиотеке (напр., трансляция XML-представления уведомления, пришедшего по протоколу SSAP).

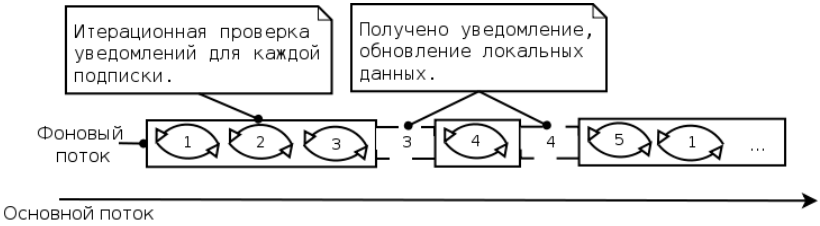

Рис. 5. Асинхронная подписка с одним потоком проверки.

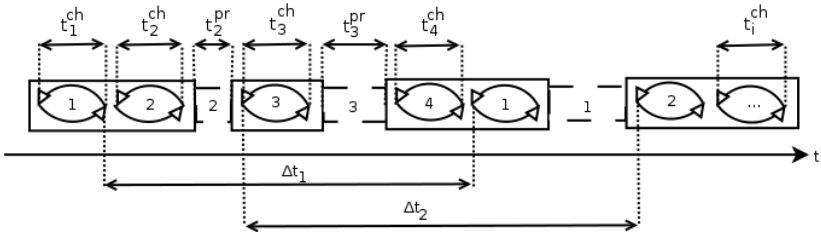

Рис. 6. Асинхронная подписка: интервал повторной проверки

4.2. Асинхронная подписка с множеством потоков. Для каждой подписки на стороне КР выделяется отдельный поток. Для проверки и обработки уведомлений существенным становится предоставляемый ОС механизм управления потоками. Пример для $n = 4$ показан на рис. 7.

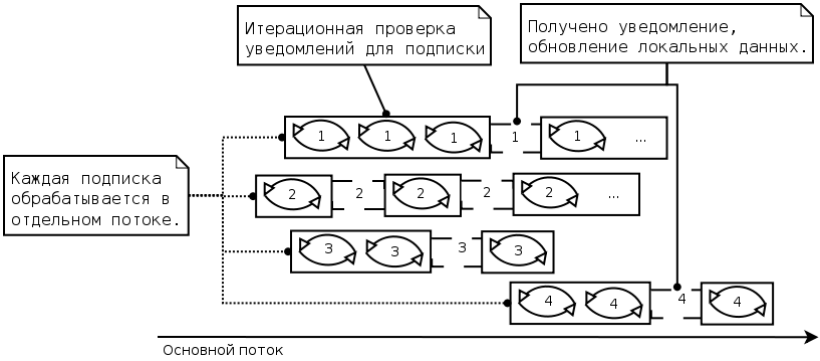

Рис. 7. Асинхронная подписка: множество потоков.

Здесь $\Delta t = t^{\mathrm{pr}}$, что эффективнее, чем (1). В то же время, увеличивается потребность в ресурсах для управления множеством потоков, что делает невозможным использование такого типа подписки на ряде низкопроизводительных устройств.

4.3. Обратные вызовы. Функции обратного вызова выполняются на стороне КР после получения уведомления. Могут применяться как в асинхронной, так и синхронной подписке. В последнем случае это позволяет удобным образом вынести код для обработки данных в отдельную функцию, которая вызывается автоматически перед завершением KPI-функции подписки (рис. 8, вверху).

В асинхронной подписке с одним потоком (рис. 8, внизу) вызов происходит из потока после обработки уведомления перед проверкой следующей подписки. Функции обратного вызова должны быть небольшими, т.к. они приостанавливают проверку уведомлений, увеличивая интервал повторной проверки для подписки j:

$$\Delta t_j = \sum_{i=1, i\neq j}^{n} \left(t_i^{\mathrm{ch}} + t_i^{\mathrm{pr}} + t_i^{\mathrm{cb}}\right) + t_j^{\mathrm{pr}}, \qquad (2)$$

где t_i^{cb} — время работы функции обратного вызова для подписки i.

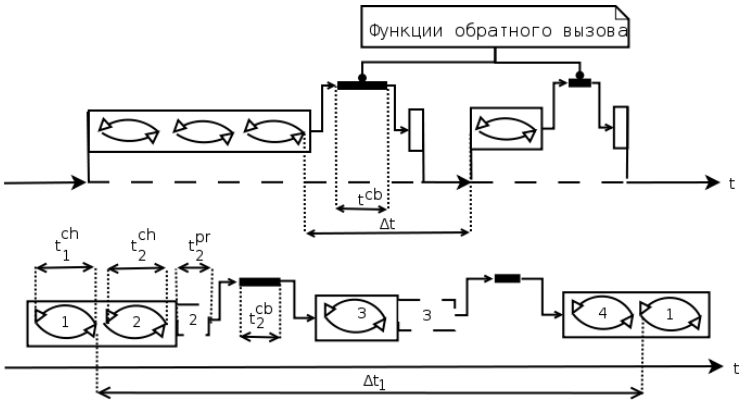

Рис. 8: Функции обратного вызова для синхронной (вверху) и асинхронной (внизу) подписки.

В отличие от t_i^{pr} это время определяется программой КР, т.е. непосредственно контролируется программистом.

Для асинхронной подписки с множеством потоков обратные вызовы выполняются из разных потоков. Проверка уведомлений для других подписок не прекращается. В то же время этот вариант создает для программиста дополнительные трудности, т.к. функция обратного вызова должна быть реентерабельна, если используется в нескольких подписках.

5. Подписка на основе онтологических объектов. Рассмотренная базовая операция подписки работает на уровне RDF, отслеживая изменение триплетов в ИП. Модельно-ориентированная разработка КР требует перехода на уровень онтологий, когда подписка позволяет отслеживать изменения индивидов и их свойств в ИП. Высокоуровневые типы подписки для OWL-объектов реализуются в инструментарии SmartSlog [5] на основе базовой подписки, но низкоуровневые детали обработки RDF-триплетов и операции по синхронизации данных скрыты от программиста.

5.1. Шаги подписки. Совпадают с базовой подпиской (см. рис. 2). Изменения касаются способа определения подписываемых данных — используются OWL-объекты, локально хранимые на стороне КР. OWL-объект определяется как RDF-граф [6], т.е. всегда возможно преобразование в набор триплетов и построение соответ-

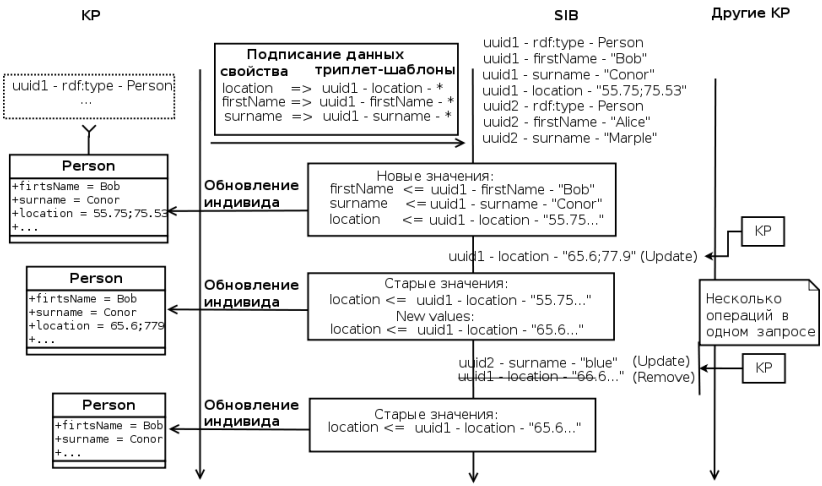

Рис. 9. Пример работы подписки на основе OWL-объектов.

ствующих Т-шаблонов. Это свойство позволяет свести подписку к базовой. При получении уведомлений находится соответствие между триплетами, получаемыми из ИП, и триплетами, составляющими локальную копию OWL-объекта. Таким образом, синхронизация с содержимым ИП происходит автоматически при обработке уведомления, позволяя KP работать с актуальными OWL-объектами.

В примере на рис. 9 показано, что KP подписывается на свойства firstName, surname и location класса Person. Выполняется начальная синхронизация локального объекта: для индивида класса Person заполняются свойства на основе полученных из ИП триплетов. Далее, при поступлении уведомлений от SIB выполняется синхронизация локального объекта. При первом уведомлении обновляется местонахождение (свойство location). Затем свойство удаляется (напр., человек вышел за границы отслеживаемой местности).

Использование онтологических объектов порождает новые типы высокоуровневых подписок, включая синхронную и асинхронную подписку на OWL-классы и свойства индивида.

5.2. Подписка на свойства. Отслеживаются изменения свойств индивидов. В качестве OWL-объекта задается набор пар “индивид – список свойств”, притом каждая определяет, какие свойства какого индивида нужно отслеживать.

Синхронизации локальных свойств выполняется автоматически. Она отслеживает три вида изменений: установка, обновление и удаление. При начальной синхронизации заданные свойства всегда устанавливаются на основе данных их ИП. При синхронизации выполняются следующие шаги.

1. Определение измененных свойств на основе полученных в уведомлении триплетов из ИП.
2. Для каждого свойства определяется необходимое действие (установить, удалить, изменить).
3. Выполнение действий по приведению локальных индивидов в актуальное состояние.

На третьем шаге обработка свойств данных и объектных свойств различается. Для свойства данных его значение берется непосредственно из триплета. Для объектных свойств из триплета берется идентификатор индивида, используя который проверяется наличие локальной копии индивида. Если ее нет, то она создается.

5.3. Подписка на классы. OWL-класс индивида определяется триплетом `rdf:type`. Подписка на него отслеживает появление или удаление индивидов заданного класса в ИП.

На рис. 10 показан пример, где КР A отслеживает присутствие

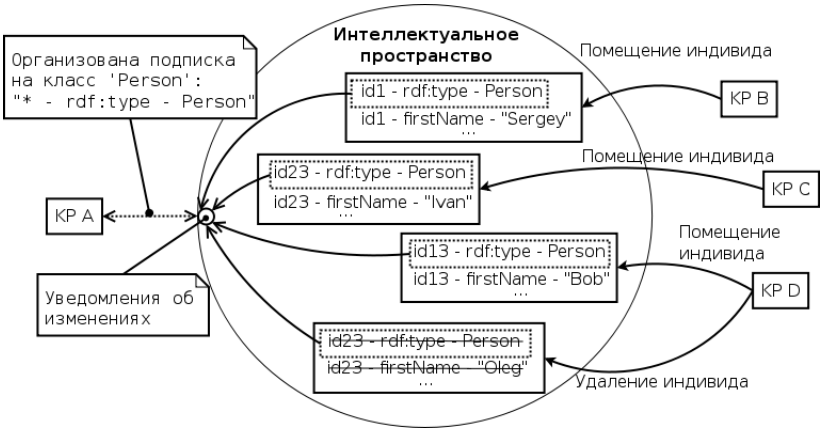

Рис. 10. Подписка на OWL-класс.

```
1: // Создаем объект управления подпиской.
2: subscription_t subscription =
       sslog_new_subscription(false);

3: // Формирование OWL-объекта для подписки на класс Person
4: sslog_sbcr_add_class(subscription, CLASS_PERSON);

5: // Подписываемся на появление и удаление индивидов
6: if (sslog_sbcr_subscribe(subscription) != ERROR_NO) {
7:     ...
8: }
9: // Проверяем наличие уведомлений
10: while (true) {
11:     // Синхронная проверка до получения данных.
12:     if (sslog_sbcr_wait(subscription) != ERROR_NO) {
13:         ...
14:     }
15:     // Получение новых объектов
16:     ...
17: }
18: // Отмена подписки.
19: sslog_sbcr_unsubscribe(subscription);
```

Листинг 2. Пример кода на языке C для синхронной подписки на класс (онтологическая КР-библиотека SmartSlog).

"людей" в ИП. Для этого A подписывается на класс Person. Персональные агенты (КР B, C, D) публикуют в ИП информацию о конкретном человеке (в виде индивида). Уведомления, поступающие к A, содержат (вместе с информацией о классе) идентификатор индивида. Идентификатор затем используется для получения свойств этого индивида или для удаления его локальной копии в A, если индивид был удален из ИП.

Пример кода КР для подписки на класс приведен в листинге 2. Для управления подпиской создается специальный объект, который далее используется как параметр в функциях онтологической КР-библиотеки (генерируется инструментарием SmartSlog).

5.4. Коллизии при одновременном изменении свойств. В отличие от традиционной модели публикации/подписки [9] плат-

форма Smart-M3 допускает, чтобы КР-подписчик выступал и издателем [2]. Такой КР может изменять данные, на которые подписан.

При работе с индивидом КР хранит локальную копию. Другие КР могут изменять в ИП значения свойств этого индивида. Обновленные значения для учета в локальной копии можно получить с помощью разовых запросов к ИП. Если индивид участвует в подписке, то изменения синхронизируются автоматически на основе значений, содержащихся в уведомлениях.

Если же КР вносит свои изменения в локальную копию, то синхронизация может быть невозможна из-за нарушения соответствия между данными из уведомления и локальными данными. На рис. 11 показана последовательность шагов при изменении подписанных свойств индивида. КР выполняет локальное изменение (шаг 1) и направляет в ИП запрос на изменение (шаг 2). В результате, SIB разошлет уведомления всем КР, подписанным на это свойство. Текущий КР также его получит (шаг 3), но в этом случае локальные данные уже синхронизированы. Если данные в ИП изменяются между шагами 1 и 2, то первым придет уведомление, инициированное другим КР. Для избежания такой коллизии КР может использовать механизм блокировки данных в ИП [13]. Тогда КР должен заблокировать в ИП свойства перед их локальным изменением.

Другой вариант, позволяющий избежать коллизии, показан на рис. 12. Когда КР нужно локально изменить подписанные свойства, он вначале отправляет запрос на их изменение в ИП (шаг 1). Локальная копия будет синхронизирована при получении уведом-

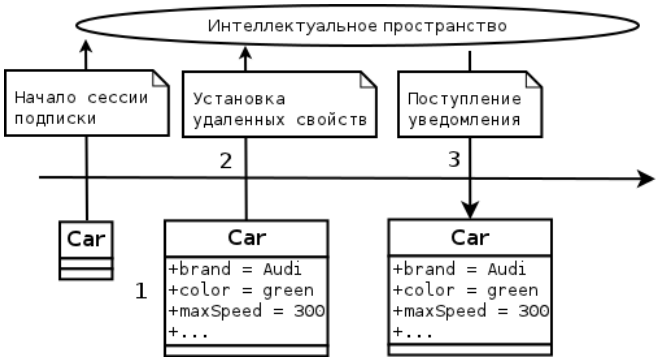

Рис. 11. Одновременное изменение подписанных свойств.

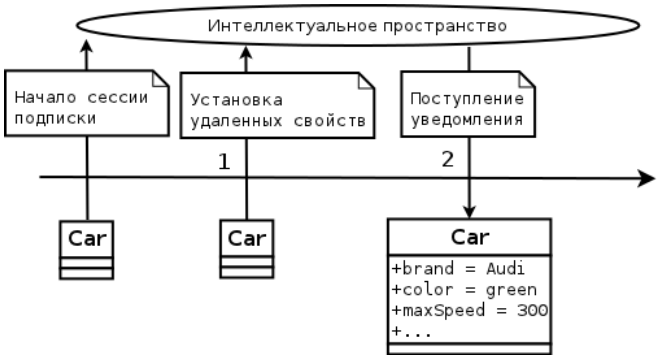

Рис. 12. Корректное изменение подписанных свойств.

ления (шаг 2). Отметим, что производительность снижается, т.к. тратится время на ожидание уведомления.

6. Заключение. Разработка многоагентных приложений для интеллектуальных пространств требует использования модели публикации/подписки. Модель является основополагающей при программировании взаимодействия агентов в такой слабосвязанной распределенной среде как ИП, реализуемое на платформе Smart-M3. Операция подписки позволяет избежать излишней передачи и обработки данных и востребована в проактивных сервисах. Последние характерны для интеллекутальных пространств, которые охватывает широкий спектр вычислительных устройств из разнообразного окружения (аудитории, офисы, дома, общественные места, транспорт и т.п.). Несмотря на то, что платформа Smart-M3 поддерживает базовую операцию подписки как коммуникационный примитив, его использование требует низкоуровневого программирования в терминах RDF-триплетов.

Проведенная в статье систематизация алгоритмов базовой подписки показывает, с какими трудностями обработки и синхронизации данных сталкивается программист при работе на уровне RDF. Рассмотренные способы обработки уведомлений по подписке сведены в табл. 1. Рассмотренные типы подписки перечислены в табл. 2. Предлагаемые высокоуровневые типы подписки оперируют на онтологическом уровне (OWL). В частности, поддерживается автоматическая синхронизация локальных данных на стороне КР с содержимым ИП. Такой подход следует принципам автомати-

Таблица 1. **Способы обработки уведомлений по подписке**

№	Способ	Доп. потоки	Описание
1	Синхронная	Нет	Выполняется в основном потоке. Простота обработки.
2	Синхронная, обратный вызов	Нет	Выполняется в основном потоке. Обработка вынесена в функцию обратного вызова для удобства.
3	Асинхронная, один поток	Один	Каждая подписка ожидает обработки других подписок, обработка подписок по кругу.
4	Асинхронная, один поток, обратный вызов	Один	Обратный вызов определяет задержку обработки подписки.
5	Асинхронная, множество потоков	Равно числу подписок	Для каждой подписки создается отдельный поток. Нет ожидания других подписок.
6	Асинхронная, множество потоков, обратный вызов	Равно числу подписок	Обратный вызов является синхронным по отношению к потоку обработки подписки.

Таблица 2. **Типы подписок**

№	Тип	Описание
1	Подписка на RDF-триплеты	Набор T-шаблонов определяет множество подписываемых триплетов.
2.1	Подписка на свойства данных	Отслеживаются изменения значений свойств данных.
2.2	Подписка на объектные свойства	Отслеживаются изменения ссылок индивидов на других индивидов, включая удаление и появление ссылки.
2.3	Подписка на произвольный набор свойств	Комбинация 2.1 и 2.2.
3	Подписка на класс	Отслеживаются появление и удаление индивидов заданного класса.

зированной модельно-ориентированной разработки и реализуется нами в инструментарии SmartSlog. Скрытие низкоуровневых операций машинно-ориентированной обработки RDF-триплетов и возможность использования онтологических объектов в программном

коде снижают сложность разработки и снижают затраты при программировании взаимодействия агентов в ИП, включая отслеживание появления/выхода объектов в/из ИП.

Литература

1. *Honkola J., Laine H., Brown R., Tyrkko O.* Smart-M3 Information Sharing Platform // Proc. IEEE Symp. Computers and Communications (ISCC'10). 2010. P. 1041–1046.

2. *Korzun D.G., Balandin S.I., Luukkala V., Liuha P., Gurtov A.V.* Overview of Smart-M3 Principles for Application Development // Proc. Congress on Information Systems and Technologies (IS&IT'11), Conf. Artificial Intelligence and Systems (AIS'11). Moscow: Physmathlit, 2011. Vol. 4. P. 64–71.

3. *Oliver I., Honkola J., Ziegler J.* Dynamic, localized space based semantic webs // Proc. IADIS Int'l Conf. WWW/Internet. 2008. P. 426–431.

4. *Horrocks I.* Ontologies and the semantic web // Commun. ACM. 2008. Vol. 51. No 12. P. 58–67.

5. *Корзун Д.Ж., Ломов А.А., Ванаг П.И.* Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе Smart-M3 // Программная инженерия. 2012. № 5. С. 6–14.

6. *Bechhofer S.* OWL Web Ontology Language Reference / S. Bechhofer, F. van Harmelen, J. Hendler et al. — Электрон. дан. — 2004. — Режим доступа: `http://www.w3.org/TR/owl-ref/`, свободный. — Загл. с экрана.

7. *Lomov A.A., Korzun D.G.* Subscription Operation in Smart-M3 // Proc. 10th Conf. Open Innovations Association FRUCT and 2nd Finnish-Russian Mobile Linux Summit. 2011. P. 83–94.

8. *Гамма Э., Хельм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. Паттерны проектирования // СПб.: Питер, 2011. — 268 С.

9. *Eugster P., Felber P., Kenmarrec A.M., Guerrout R.* The many faces of publish/subscribe // ACM Computing Surveys. 2003. Vol. 35. P. 114–131.

10. *Смирнов А.В., Левашова Т.В., Пашкин М.П., Шилов Н.Г.* Онтолого-ориентированный многоагентный подход к построению систем интеграции знаний из распределённых источников // Информационные технологии и вычислительные системы. 2002. № 1. С. 62–82.

11. *Lynch D., Keeney J., Lewis D., O'Sullivan D.* A Proactive approach to Semantically Oriented Service Discovery // Proc. 2nd Workshop on Innovations in Web Infrastructure (IWI 2006). Co-located with 15th Int'l World-Wide Web Conf. 2006. — Режим доступа: `http://www.tara.tcd.ie/jspui/handle/2262/30806`, свободный.

12. *Вальченко Ю., Кашевник А.М.* Современные подходы к построению контекстно-ориентированных систем в интеллектуальных пространствах // Труды СПИИРАН. 2011. Вып. 19. С. 102–127.

13. *D'Elia A., Manzaroli D., Honkola J., Cinotti T.S.* Access Control at Triple Level: Specification and Enforcement of a Simple RDF Model to Support Concurrent Applications in Smart Environments // Proc. 11th Int'l Conf. Next Generation Wired/Wireless Networking (NEW2AN'11) and 4th Conf. Smart Spaces (ruSMART'11). 2011. P. 63–74.

Ломов Александр Андреевич — аспирант, кафедра информатики и математического обеспечения, математический факультет, Петрозаводский государственный университет (ПетрГУ). Область научных интересов: семантический веб, онтологическое моделирование, автоматизированная разработка ПО. Число научных публикаций — 8. lomov@cs.karelia.ru, www.cs.petrsu.ru; пр. Ленина, д. 33, г. Петрозаводск, 185910, РФ; р.т. +7(8142)711015, факс +7(8142)711000. Научный руководитель — Д.Ж. Корзун.

Aleksandr A. Lomov — PhD student; Department of Computer Science, Faculty of Mathematics, Petrozavodsk State University (PetrSU). Research interests: Semantic Web, Ontological modeling, automated software engineering. The number of publications — 8. lomov@cs.karelia.ru, www.cs.petrsu.ru; Lenin St. 33, Petrozavodsk, 185910, Russia; office phone +7(8142)711015, fax +7(8142)711000. Scientific advisor — D.G. Korzun.

Корзун Дмитрий Жоржевич — канд.физ.-мат.наук, доцент, кафедра информатики и математического обеспечения, математический факультет, Петрозаводский государственный университет (ПетрГУ). Область научных интересов: анализ распределенных систем, дискретное моделирование, повсеместные вычисления и интеллектуальные пространства, Интернет вещей, технологии разработки ПО, проектирование алгоритмов и вычислительная сложность, линейный диофантов анализ и его приложения, теория формальных языков и методы трансляции. Число научных публикаций — 100. dkorzun@cs.karelia.ru, www.cs.petrsu.ru; пр. Ленина, д. 33, г. Петрозаводск, 185910, РФ; р.т. +7(8142)711084, факс +7(8142)711000.

Dmitry G. Korzun — Ph.D, Adjunct Professor; Department of Computer Science, Faculty of Mathematics, Petrozavodsk State University (PetrSU). Research interests: analysis and evaluation of distributed systems, discrete modeling, ubiquitous computing in smart spaces, Internet of Things, software engineering, algorithm design and complexity, linear Diophantine analysis and its applications, theory of formal languages and parsing. The number of publications — 100. dkorzun@cs.karelia.ru, www.cs.petrsu.ru; Lenin St. 33, Petrozavodsk, 185910, Russia; office phone +7(8142)711084, fax +7(8142)711000.

Поддержка исследований. Работа выполнена при поддержке Программы стратегического развития ПетрГУ на 2012–2016 годы в рамках реализации комплекса мероприятий по развитию научно-исследовательской деятельности.

Рекомендовано СПИИРАН, лабораторией интегрированных систем автоматизации, заведующий лабораторией Смирнов А.В., д-р техн. наук, проф.

Статья поступила в редакцию 12.09.2012.

РЕФЕРАТ

Ломов А.А., Корзун Д.Ж. **Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3**

Платформа Smart-M3 предназначена для создания многоагентных распределенных приложений, взаимодействующих в общем "интеллектуальном пространстве" (ИП). Создается разделяемое хранилище динамической информации, реализующее среду "повсеместных вычислений". Программные агенты работают на разнообразных устройствах, представленных в текущем физическом окружении (сенсоры, камеры и т.п.) и сети (персональные компьютеры, ноутбуки и т.п.). Помимо традиционных разовых запросов к ИП взаимодействие агентов может применять операцию подписки — постоянный запрос на определенное подмножество данных в ИП. При изменении каким-либо участником этих данных агенту доставляется уведомление. Базовая операция подписки платформы Smart-M3 работает на уровне RDF-модели представления данных, где базовым элементом выступают триплеты (субъект–предикат–объект). Такая модель является низкоуровневой, ориентируясь на машинную обработку больших массивов информации.

Целью работы является построение и реализация высокоуровневых типов подписки для автоматизированной модельно-ориентированной разработки программных агентов для интеллектуальных пространств платформы Smart-M3. Высокоуровневая подписка работает с OWL-объектами, каждый представлен в ИП как RDF-граф, а агент хранит локальную копию. Разработка программного кода агента выполняется в терминах онтологических классов, свойств и индивидов. Такой подход реализуется в инструментарии SmartSlog.

В статье, во-первых, систематизируются алгоритмы базовой операции подписки для уровня RDF. Рассматривается классификация способов обработки уведомлений на стороне агента, включая синхронные и асинхронные варианты. Во-вторых, предлагаются высокоуровневые типы подписки: подписка на свойства индивида (включая свойства данных и объектные свойства) и подписка на класс (отслеживание появления и удаления индивида в ИП). Высокоуровневые типы подписки сводятся к базовой подписке. В инструментарии SmartSlog это сведéние выполняется автоматически, скрывая от программиста низкоуровневые детали. Последние включают преобразование OWL-объектов в RDF-триплеты и обратно, вызовы базовой подписки и синхронизацию локальных копий OWL-объектов с содержимым ИП.

SUMMARY

Lomov A.A., Korzun D.G. **Subscription operation for applications in smart spaces of Smart-M3 platform.**

The Smart-M3 platform aims at constructing distributed multi-agent applications that operate in a common smart space. It creates a shared storage of dynamic information, implementing a ubiquitous computing environment. Agents run on various devices available in the surrounding physical environment (sensors, cameras, etc.) and in the network (personal computers, laptops, etc.). In addition to instant queries to smart space, agents can use the subscription operation. It is a persistent query to a given data set in the smart space. Whenever the data are changed by other participants the agent receives the notification. Basic Smart-M3 subscription operation operates on the level of RDF representation model. Elementary units are triples (subject–predicate–object). The model is low-level, being oriented to machine processing of large data sets.

The goal of this paper is the construction and implementation of high-level subscription types; they are needed in automated model-driven development of Smart-M3 application agents. High-level subscription operates with OWL-objects, each is represented as an RDF-graph in the smart space, and the agent keeps a local copy. Programming is held out in terms of ontological classes, properties and individuals. This approach is realized in SmartSlog SDK.

This paper, first of all, systematizes the algorithms of the basic Smart-M3 subscription operation on the RDF level. We consider classification of subscription notification processing at the agent side, including synchronous and asynchronous variants. Secondly, we contribute high-level subscription types: subscription to properties of individuals (data and object properties) and subscription to class (tracking appearance and departure of individuals). The high-level subscription is reduced to the basic one. In SmartSlog SDK, this reduction is automated, hiding low-level details from the programmer. The details include bidirectional transformation between OWL and RDF representations, calling the basic subscription and synchronization of local copies of OWL objects.