Декабрь, № 8 **Физико-математические науки** **2013**

УДК 004.4

АЛЕКСАНДР АНДРЕЕВИЧ ЛОМОВ
аспирант кафедры информатики и математического обеспечения математического факультета, Петрозаводский государственный университет (Петрозаводск, Российская Федерация)
lomov@cs.karelia.ru

ВЗАИМОДЕЙСТВИЕ ПРОГРАММНОГО АГЕНТА НА УРОВНЕ СЕССИИ С ИНТЕЛЛЕКТУАЛЬНЫМИ ПРОСТРАНСТВАМИ*

Рассматривается задача доступа программных агентов к интеллектуальным пространствам на платформе Smart-M3. Предлагается модель параллельных сеансов для работы с несколькими ИП. Модель определяет сессию, объединяющую несколько сеансов, операции управления и хранилище данных, оперирующее онтологическими сущностями (класс, свойство, индивид). Предлагаемая модель реализована в инструментарии SmartSlog и позволяет упростить разработку агентов при взаимодействии с несколькими ИП.

Ключевые слова: интеллектуальные пространства, Smart-M3, сессия, SmartSlog, многоагентные системы

ВВЕДЕНИЕ

На программной платформе Smart-M3 реализуются многоагентные распределенные системы, в которых агенты взаимодействуют в общем интеллектуальном пространстве (ИП, от англ. smart space) [7], [9]. Агент – программа, работающая на вычислительном устройстве. Семантический информационный брокер (SIB, от англ. semantic information broker) реализует интерфейс доступа агентов к ИП.

Информация в ИП представлена моделью RDF (от англ. resource description framework) в виде троек: субъект, предикат, объект. Это позволяет проводить машинную обработку и вывод новых понятий [5]. Таким образом, ИП рассматривается как база знаний [4], формируемая агентами и управляемая брокерами. Агент взаимодействует с брокером по протоколу SSAP (от англ. smart space access protocol) на основе сеансов клиент-сервер [8]. Для отслеживания изменений на стороне ИП в рамках сеанса организуются подписки [2].

В статье предлагается частный случай модели взаимодействия в распределенной системе [3] для поддержки параллельных сеансов агента с ИП. Модель определяет сессию, операции управления и хранилище данных. Используется онтолого-ориентированный подход к разработке агента [11] на основе языка веб-онтологий OWL. Подробнее о разработке агентов в [12], об использовании онтологий для разработки программного обеспечения и агентов в [10], [11].

Модель используется в авторском инструментарии для разработки Smart-M3 агентов SmartSlog [1] и позволяет уменьшить объем программного кода агента, использующего параллельные сеансы, и способствует разработке агента с более высоким уровнем переносимости.

УРОВЕНЬ СЕАНСОВ

Для выполнения операций агент устанавливает сеанс с брокером. Сеанс обеспечивает взаимодействие, управляет обменом данных, синхронизацией задач. Операции протокола SSAP манипулируют с данными, представленными тройками. Операция подписки реализует постоянный запрос, аналогичный запросу на выборку данных, но хранимый на стороне брокера. Брокер будет уведомлять агента при изменении данных, подходящих под запрос.

Будем представлять сеанс p как (Q_p, T_p), где Q_p – это множество подписок агента в сеансе, T_p – набор всех троек, используемых (отправляемых и получаемых) в операциях сеанса. Множество всех сеансов и подписок заданного агента обозначим как P и Q.

Агент работает с локальными копиями данных, поэтому при использовании параллельных сеансов необходимо объединять данные, так как в распределенных системах связанные между собой данные могут получаться из различных источников. Пусть $D = Y_{p \in P} T_p$ – локальное хранилище на стороне агента.

Рассмотрим пример использования параллельных сеансов на основе «умного дома», где $ИП_1$ работает внутри дома, а $ИП_2$ работает в доме и снаружи (рис. 1). В $ИП_1$ есть информация о звонках и электронных письмах, в $ИП_2$ – о звонках. В доме агент использует два параллельных сеанса, снаружи – один. При выходе из дома сеанс p_1 с подписками завершается. При возвращении в дом сеанс p_1 и подписки должны быть восстановлены. Нужно провести синхронизацию локального хранилища D с данными сеанса T_{p1}, например, добавить данные о письмах и звонках в хранилище D, учитывая дублирование.

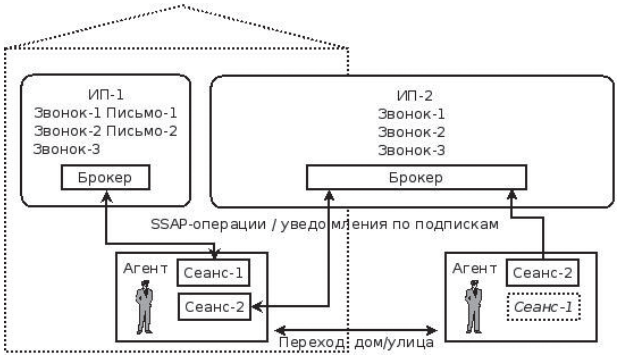

Рис. 1. Работа параллельных сеансов

Таким образом, для поддержки параллельных сеансов нужно: 1) реализовать работу сеансов и подписок, 2) организовать хранение данных с возможностью объединения и синхронизации. Для решения этих задач предлагается объединить сеансы в сессии и реализовать хранилище данных, разделенное между сеансами сессии, с возможностью синхронизации на основе подписок.

УРОВЕНЬ СЕССИЙ

Предлагаемой моделью параллельных сеансов описываются сессия, основанная на модели многоэлементной сессии [6], операции управления и хранилище данных. Данные представлены сущностями языка онтологий OWL: классами, свойствами и индивидами (экземплярами классов). Это позволяет работать с объектами (например, человек, машина) и скрыть низкоуровневые детали, упрощая разработку агента [11]. Данный подход позволяет использовать модель в инструментарии SmartSlog.

Сеансы объединяются в сессии s, реализуемой на прикладном уровне сетевой модели OSI [3] и определяемой как (P_s, Q_s, D_s), где P_s – сеансы, Q_s – подписки, D_s – хранилище данных. С точки зрения онтологического представления хранилище D_s определим как $O_c \cup O_v \cup O_i$, где O_c, O_v, O_i – множества классов, свойств и индивидов. Все объекты онтологии O представим как $O_c \cup O_v \cup O_i$.

Если сеанс $p \in P_s$, то в контексте сессии представим его как $p = (Q_p, O_p)$, где $Q_p \in Q_s$, $O_p \in D_s$. Сеанс или подписка могут находиться в активном состоянии или в состоянии ожидания. Пусть множество сеансов P_s и подписок Q_s – это дизъюнкции $P_a \cup P_w$ и $Q_a \cup Q_w$, где P_a, Q_a и P_w, Q_w – множества сеансов и подписок в активном состоянии и в состоянии ожидания. Для сеанса $p \in P_a$ установлено сетевое соединение с брокером для операций по протоколу SSAP. Для подписки $q \in Q_a$ ожидаются уведомления от брокера. Для сеанса $p \in P_w$ и подписки $q \in Q_w$ сетевое взаимодействие отсутствует.

В модели управление объектами выполняется на основе операций, каждое имеет вид (табл. 1):

$$U = \{REG, DEL, ACTIVE, WAIT\},$$

где *REG* – регистрация, *DEL* – удаление, *ACTIVE* – активация, *WAIT* – ожидание. Регистрация – это локальная операция, устанавливающая связь между объектами (сеансами, подписками, сущностями онтологий) и сессией. Операция *DEL* удаляет связь объекта с сессией. Операции *ACTIVE* и *WAIT* используются для управления состоянием всей сессии, отдельных сеансов и подписок.

УПРАВЛЕНИЕ ОБЪЕКТАМИ ДАННЫХ В РАМКАХ СЕССИИ

Все объекты сессии имеют уникальные идентификаторы, при помощи которых осуществляется регистрация. Для сессии и подписок идентификаторы генерируются при создании. Идентификаторы сеансов задаются разработчиком. Для классов и свойств используются URI (от англ. Uniform Resource Identifier, унифицированный идентификатор ресурса) [5] из онтологии. Для индивидов URI задается разработчиком либо создается на основе URI родительского класса.

На рис. 2 представлена диаграмма состояния объектов в сессии. Жизненный цикл объекта начинается при регистрации его в сессии и заканчивается при удалении. Объекты онтологий после регистрации помещаются в хранилище. Для управления состоянием сеансов и подписок применимы операции *ACTIVE* и *WAIT*. Использование нескольких сеансов с различными наборами подписок позволяет организовать группы подписок и активировать часть подписок при определенных условиях, останавливая другие, и наоборот.

Состояние сессии определяет состояние всех сеансов и подписок (см. таблицу). Выделим три способа использования нескольких сессий: 1) активация-ожидание, 2) активация-завершение, 3) смешанный. При способе 1 неиспользуемые сессии переводятся в состояние ожидания. При способе 2 сессия завершается после работы с ней. Способ 3 комбинирует два предыдущих. Например, агент использует часть сессий с постоянным набором сеансов и часть создает для разовых взаимодействий с ИП.

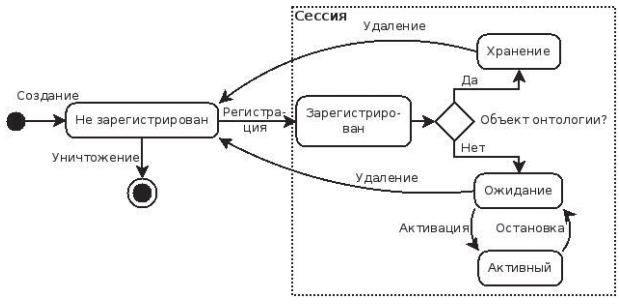

Рис. 2. Состояние объектов сессии

Операции управления объектами

Регистрация и удаление объекта из сессии; объекты онтологии помещаются или удаляются из хранилища	
$REG(s,x)=\begin{cases}(P_s\cup\{x\},Q_s,D_s),\ если\ x\in P\ и\ x\notin P_s\\(P_s,Q_s\cup\{x\},D_s),\ если\ x\in Q\ и\ x\notin Q_s\\(P_s,Q_s,D_s\cup\{x\}),\ если\ x\in O\ и\ x\notin D_s\end{cases}$	$DEL(s,x)=\begin{cases}(P_s\setminus\{x\},Q_s,D_s),\ если\ x\in P_s\\(P_s,Q_s\setminus\{x\},D_s),\ если\ x\in Q_s\\(P_s,Q_s,D_s\setminus\{x\}),\ если\ x\in D_s\end{cases}$
Активация сессии, сеанса или подписки. Активация сессии приводит к активации всех сеансов в сессии. Активация сеанса приводит к активации подписок связанных с сеансом	
$ACTIVE(s)\rightarrow ACTIVE(s,p),$ для $\forall p\in P_s$	$ACTIVE(s,x)=\begin{cases}(P_s=\{P_w\setminus\{x\}\cup\{P_a\cup\{x\}\},Q_s,D_s)\rightarrow\\\quad ACTIVE(s,q)\ для\ \forall q\in Q_x,\ если\ x\in P_s\\(P_s,Q_s=\{Q_w\setminus\{x\}\cup\{Q_a\cup\{x\}\},D_s),\\\quad если\ x\in Q_s\end{cases}$
Перевод в состояние ожидание сессии, сеанса или подписки. Переход сессии переводит в состояние ожидания все сеансы и подписки. Переход сеанса переводит все подписки в состояние ожидания	
$WAIT(s)\rightarrow WAIT(s,p),$ для $\forall p\in P_a$	$WAIT(s,x)=\begin{cases}(P_s=\{P_a\setminus\{x\}\cup\{P_w\cup\{x\}\},Q_s,D_s)\rightarrow\\\quad WAIT(s,q)\ для\ \forall q\in Q_x,\ если\ x\in P_s\\(P_s,Q_s=\{Q_a\setminus\{x\}\}\cup\{Q_w\cup\{x\}\},если\ x\in Q_s\end{cases}$

РАЗРАБОТКА АГЕНТА НА УРОВНЕ СЕССИИ

На основе модели параллельных сеансов предложена следующая схема разработки агента (рис. 3).

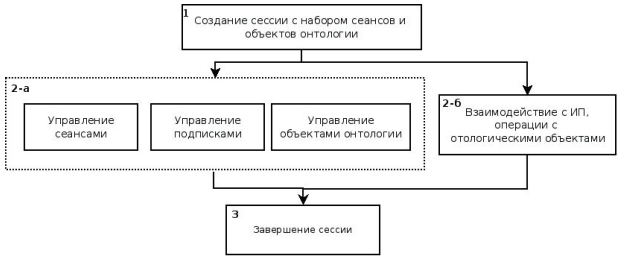

Рис. 3. Схема разработки агента

Шаг 1. Создание сессии *s*, сеансов, классов и подготовка сессии к работе операцией *REG*(*s*, *x*). Если в сессии нет сеансов, то невозможно взаимодействовать с ИП. При отсутствии классов и свойств нельзя создавать индивидов и устанавливать связи.

Шаг 2. Использование а) операций для управления состоянием (*ACTIVE*(*s*, *x*), *WAIT*(*s*, *x*)), работы с локальными данными, б) операций для взаимодействия с ИП. Операции для работы с локальными данными и взаимодействия зависят от реализации в инструментарии.

Шаг 3. Завершение сессии: перевод сессии в состояние ожидания и удаление всех зарегистрированных объектов и сущностей онтологии (операции *WAIT*(*s*) и *DEL*(*s*, *x*)).

На примере «умного дома» (см. рис. 1) рассмотрим использования сессии (см. пример ниже). В строках 1 и 2 создаются сессия, сеансы и классы. В строке 3 объекты регистрируются в сессии. В строках 4–6 создаются и регистрируются подписки. Строки 7–11 описывают логику управления состоянием сеансов в зависимости от местонахождения. Если территория «умного дома» покинута, то сессия останавливается, и из нее удаляются все объекты (строки 12–13).

Пример построения агента

1. Создать сессию, сеансы: *s*, p_1, p_2
2. Создать класс телефона и электронного письма: *t*, *e*
3. Зарегистрировать объекты: *REG*(*s*, p_1), *REG*(*s*, p_2), *REG*(*s*, *t*), *REG*(*s*, *e*)
4. Создать подписки для звонков и писем сеанса p_1: a_t, a_e
5. Создать подписку для звонков сеанса p_2: b_t
6. Зарегистрировать подписки: *REG*(*s*, a_t); *REG*(*s*, a_e); *REG* (*s*, b_t)
7. Пока В_Умном_Доме () = «Да»
8. Если (В_Доме () = «Да» И $p_1 \in P_w$) {*ACTIVE*(*s*, p_1);}
9. Если (В_Доме () = «Да» И $p_2 \in P_w$) {*ACTIVE*(*s*, p_2);}
10. Если (Снаружи () = «Да» И $p_1 \in P_a$) {*WAIT*(*s*, p_1);}
11. Конец
12. Остановить сессию *WAIT*(*s*)
13. Завершить *s* // *DEL*(*s*, *x*)

ЗАКЛЮЧЕНИЕ

Предложенная модель позволяет агенту использовать несколько сеансов в одной сессии, управлять объектами в сессии и предоставляет локальное хранилище, разделенное между сеансами сессии. Хранилище синхронизируется при помощи подписок. Подписки можно разбивать на группы и управлять их состоянием. В результате агент имеет возможность: а) работать с параллельными сеансами, б) управлять состоянием сеансов и подписок, в) объединять данные различных ИП, г) синхронизировать локальное хранилище. Модель используется в инструментарии SmartSlog, что позволяет работать с объединенной и поддерживаемой в актуальном состоянии информацией об объектах предметной области разных ИП.

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

СПИСОК ЛИТЕРАТУРЫ

1. Корзун Д. Ж., Ломов А. А., Ванаг П. И. Автоматизированная модельно-ориентированная разработка программных агентов для интеллектуальных пространств на платформе Smart-M3 // Теоретический и прикладной научно-технический журнал «Программная инженерия». 2012. № 5. С. 6–14.
2. Ломов А. А., Корзун Д. Ж. Операция подписки для приложений в интеллектуальных пространствах платформы Smart-M3 // Труды СПИИРАН. 2012. № 4. С. 439–458.
3. Таненбаум Э., Стеен ван М. Распределенные системы. Принципы и парадигмы. М.; СПб.: Питер, 2003.
4. Akerkar R., Sajja P. Knowledge-Based Systems // Jones & Bartlett Publishers. 2010. P. 354.
5. Allemang D., Hendler J. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL // Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2008.
6. Deni'elou P.-M., Yoshida N. Multiparty Session Types Meet Communicating Automata // Proceedings of the 21st European conference on Programming Languages and Systems. 2012. P. 194–213.
7. Honkola J., Laine H., Brown R., Tyrkkö O. Smart-M3 Information Sharing Platform // Proc. IEEE Symp. Computers and Communications (ISCC'10). 2010. P. 1041–1046.
8. Kiljander J., Morandi F., Soininen J.-P. Knowledge Sharing Protocol for Smart Spaces // International Journal of Advanced Computer Science and Applications (IJACSA). 2012. Vol. 3. № 9.
9. 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). 2011. P. 64–71.
10. Obitko M., Marík V. Adding OWL Semantics to Ontologies Used in Multi-agent Systems for Manufacturing // Lecture Notes in Computer Science. 2003. Vol. 2744. P. 189–200.
11. Pan J. Z., Staab S., Aßmann U., Ebert J., Zhao Y. Ontology-Driven Software Development // Berlin: Springer, 2013.
12. Wooldridge M. An Introduction to MultiAgent Systems // Wiley. 2009. P. 484.

Lomov A. A., Petrozavodsk State University (Petrozavodsk, Russian Federation)

AGENT' S SESSION BASED INTERACTION WITH SMART SPACE

A problem of the agents' access to smart spaces (SS) on Smart-M3 platform is considered. A model of parallel sessions' support for simultaneous interaction with several SS is offered. The model includes: a session, management operations, and data storage. The session component is based on the multiparty session model uniting a number of sessions. It allows the agent to work with several SS. The suggested storage gives the opportunity to the agent to operate with ontological elements (class, property, individual). The offered model is realized in the SmartSlog tools and allows simplifying agents' development for interactions with several SS.

Key words: smart spaces, smart-M3, session, smartSlog, multi-agent systems

REFERENCES

1. Korzun D. Zh., Lomov A. A., Vanag P. I. Automated model-based development of smart space agents for Smart-M3 platform [Avtomatizirovannaya model'no-orientirovannaya razrabotka programmnykh agentov dlya intellektual'nykh prostranstv na platforme Smart-M3]. *Teoreticheskiy i prikladnoy nauchno-tekhnicheskiy zhurnal "Programmnaya inzheneriya"*. 2012. № 5. P. 6–14.
2. Lomov A. A., Korzun D. Zh. Subscription operation for applications in smart spaces of Smart-M3 platform [Operatsiya podpiski dlya prilozheniy v intellektual'nykh prostranstvakh platformy Smart-M3]. *SPIIRAS Proc.* 2012. № 4. P. 439–458.
3. Tanenbaum E., Steen van M. *Raspredelennye sistemy: Printsypy i paradigmy* [Distributed Systems: Principles and Paradigms]. Moscow; SPb., Piter Publ., 2003.
4. Akerkar R., Sajja P. Knowledge-Based Systems. Jones & Bartlett Publishers. 2010. P. 354.
5. Allemang D., Hendler J. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2008.
6. Deni'elou P.-M.. Yoshida N. Multiparty Session Types Meet Communicating Automata. Proceedings of the 21st European conference on Programming Languages and Systems. 2012. P. 194–213.
7. Honkola J., Laine H., Brown R., Tyrkkö O. Smart-M3 Information Sharing Platform. Proc. IEEE Symp. Computers and Communications (ISCC'10). 2010. P. 1041–1046.
8. Kiljander J., Morandi F., Soininen J.-P. Knowledge Sharing Protocol for Smart Spaces. International Journal of Advanced Computer Science and Applications (IJACSA). 2012. Vol. 3, № 9.
9. 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). 2011. P. 64–71.
10. Obitko M., Marík V. Adding OWL Semantics to Ontologies Used in Multi-agent Systems for Manufacturing. Lecture Notes in Computer Science. 2003. Vol. 2744. P. 189–200.
11. Pan J. Z., Staab S., Aßmann U., Ebert J., Zhao Y. Ontology-Driven Software Development. Berlin: Springer. 2013.
12. Wooldridge M. An Introduction to MultiAgent Systems. Wiley. 2009. P. 484.

Поступила в редакцию 07.03.2013