ПРИКАСПИЙСКИЙ ЖУРНАЛ:
управление и высокие технологии № 3 (15) 2011

УДК 004.89

ПОИСК УСТОЙЧИВЫХ СИТУАЦИОННЫХ КОМПОЗИЦИЙ СЕРВИСОВ

Ишкина Евгения Геннадиевна, *аспирант, Астраханский государственный университет, 414056, Россия, г. Астрахань, ул. Татищева, 20а, e-mail: ishkina@aspu.ru.*

В статье описана коллективная память гетерогенных сервисов, доступных в рапределенном окружении: основной компонент самоадаптирующегося промежуточного программного обеспечения для мобильных повсеместных систем в рамках проекта ASTRA (Adaptive Service Technologies for Reliable Access, Адаптивные технологии сервисов для надежного доступа). Основная идея проекта заключается в использовании коллективного разума сервисов для предоставления пользователям персонализированных сервисов, наиболее подходящих для конкретной ситуации использования в различных сферах применения (туризм, образование и др.). Ядром проекта является промежуточное программное обеспечение с поддержкой трех ключевых особенностей: коллективного управления сервисами, независимого от целевой предметной области; задачеориентированности как основного фактора взаимодействия; способности к самоадаптации, т.е. возможности находить новые композиции сервисов и критерии их надлежащего использования. В статье описана структура компонента коллективной памяти сервисов, а также потоки данных между ним и другими компонентами промежуточного программного обеспечения, а именно компонентом интеграции сервисов (поток описаний базовых сервисов), компонентом анализа использования (поток ситуационных аннотаций сервисов), компонентом поиска новых сервисов (поток полезных составных сервисов) и компонентом ситуационной композиции сервисов (поток сервисов для заданной ситуации и неявных связей между сервисами). Каждый из этих потоков данных подробно описан вместе со связанными алгоритмами обработки.

Ключевые слова: *сервисные вычисления; сервис-ориентированная архитектура; адаптивные сервисы; композиция сервисов; интеллектуальный анализ данных; ситуационно-зависимые системы; контекстно-зависимые системы; проект ASTRA; промежуточное программное обеспечение; задачеориентированные вычисления; композиция сервисов, управляемая целями.*

MINING FOR STABLE SITUATIONAL SERVICE COMPOSITIONS

Ishkina Evgeniya G.*, Post-graduate student, Astrakhan State University, 20a Tatishchev str., Astrakhan, 414056, Russia, e-mail: ishkina@aspu.ru*

This paper describes the collective services memory component inside the self-adaptive middleware for mobiquitous systems in the framework of ASTRA project (Adaptive Service Technologies for Reliable Access). The main idea of our middleware approach is to use collective service intelligence for providing users with customized services which are the most appropriate for concrete usage situations in different domains (tourism, education, etc.). The core of ASTRA approach consists of a middleware supporting three key peculiarities: collective domain-independent service management; task-awareness as main factor of interaction; and finally, ability of self-adaptation, i.e. possibility to find new service compositions and criteria of their appropriate usage. In this paper we describe the structure of collective services memory component and data flows between it and other middleware components, namely service integration component (atomic ser-

vices description flow), usage analysis component (situational service annotations flow), service mining component (useful service compositions flow), and situational service composition component (flow of appropriate services for the given situation and flow of services implicit relationships). Each data flow is described in details along with related algorithms.

***Key words**: service computing; service-oriented architecture; adaptive services; service composition; service mining; data mining; situation awareness; context awareness; ASTRA project; middleware; task-aware computing; goal-driven service composition.*

Глобальная архитектура промежуточного программного обеспечения для проекта ASTRA представлена в [3]. В данном разделе подробно описывается один из его компонентов – коллективная память сервисов. Цель проекта ASTRA состоит в улучшении взаимодействия пользователей с информационными системами (например, в сфере туризма, образования) с использованием адаптивного сервисно-ориентированного подхода. Ядром проекта является промежуточное программное обеспечение с поддержкой трех ключевых особенностей: коллективного управления сервисами, независимого от целевой предметной области; задачеориентированности как основного фактора взаимодействия; способности к самоадаптации, т.е. возможности находить новые композиции сервисов и критерии их надлежащего использования.

Коллективная память сервисов (КПС) хранит все базовые (атомарные) сервисы, а также найденные составные сервисы. И те, и другие описаны с использованием единой метамодели. Данный компонент позволяет также осуществлять логический вывод для обнаружения неявных связей между сервисами, основанный на выявлении схожести подходящих ситуаций использования этих сервисов.

«Коллективная» означает, что все сервисы хранятся в памяти вместе с их явными и неявными связями. Явная связь между двумя сервисами заключается в их совместном использовании в рамках составного сервиса. Это совместное использование является условным и определяется описанием подходящей для него ситуации. Неявная связь между двумя сервисами заключается в схожести ситуаций их использования. Как явные, так и неявные связи включены в процесс предварительного отбора сервисов для генерации конечного сервиса.

Далее в статье описаны потоки данных между КПС и другими компонентами промежуточного программного обеспечения: поток описаний базовых сервисов, поток ситуационных аннотаций сервисов, поток полезных составных сервисов, поток сервисов для заданной ситуации и неявных связей между сервисами. Каждый из входных потоков данных подробно описан вместе со связанными алгоритмами обработки.

Потоки данных между коллективной памятью сервисов и другими компонентами промежуточного программного обеспечения. На рисунке 1 показаны потоки данных между КПС и другими компонентами промежуточного программного обеспечения.

КПС взаимодействует с четырьмя компонентами:

- *компонентом интеграции сервисов* (слой интеграции базовых сервисов), который предоставляет описания исходных сервисов с использованием стандарта WSMO [9];
- *компонентом анализа использования* (слой анализа взаимодействия), который предоставляет аннотации сервисов на основе терминов онтологий целей, контекстов и пользовательских профилей как для базовых, так и для составных сервисов КПС;
- *компонентом поиска новых сервисов (паттернов)* (слой управления КПС), который предоставляет обнаруженные полезные композиции существующих сервисов (паттерны);
- *компонентом ситуационной композиции сервисов*, который запрашивает у КПС сервисы, наилучшим образом соответствующие заданной ситуации использования, а также неявные связи между этими сервисами.

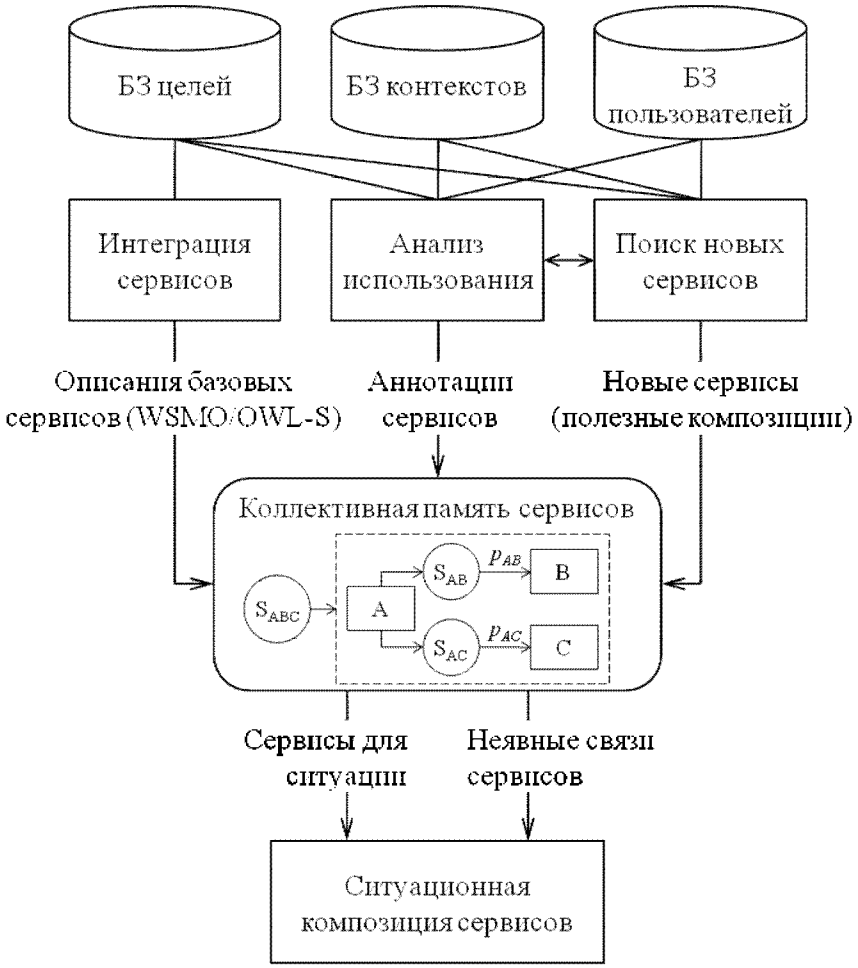

Рис. 1. Входные и выходные потоки данных для КПС

Далее подробно описан каждый из этих потоков данных.

Описания базовых сервисов. Базовые сервисы для описанного выше промежуточного программного обеспечения могут представлять собой либо веб-сервисы, либо мобильные сервисы (NFC/QR).

Для веб-сервисов существует два основных подхода к реализации:

- RESTful – подход, основанный на REST (REpresentational State Transfer) [5], архитектурном стиле для клиент-серверных приложений на базе HTTP;
- SOAP – подход, основанный главным образом на стиле RPC (удаленный вызов процедур) [7].

SOAP-сервисы автоматически описываются с использованием WSDL (Web Services Description Language, язык описания веб-сервисов) [10]. Для REST-сервисов никакие метаданные автоматически не предоставляются, однако WSDL 2.0 поддерживает веб-сервисы RESTful, кроме того, существует другой стандарт, который позволяет описать операционный синтаксис RESTful-сервисов – это WADL (Web Application Description Language, язык описания веб-приложений) [8].

Таким образом, на уровне *операционного синтаксиса* сервису предоставляется метаописание с использованием WSDL или WADL.

Следующий уровень описания – это *композиционная семантика*. WSMO [9] и OWL-S [4] являются основными стандартами для определения семантики веб-сервисов и обеспечения возможности автоматического обнаружения, композиции и вызова сервисов. WSMO лучшим образом соответствует задачам проекта ASTRA благодаря поддержке задачеориентированности через явное определение цели в описании сервиса. Однако его семантики недостаточно, поскольку требуется предоставлять ситуационное описание базовых и составных сервисов. Причем для композиций сервисов аннотация может относиться к сервису целиком или же к его части.

Третий уровень описания – ситуационная семантика – описан в следующем подразделе.

Ситуационные аннотации сервисов. Ситуационное описание сервиса определяется относительно трех измерений [3]: 1) параметры профиля пользователя (неявные знания, выведенные из истории взаимодействия пользователя с системой, а также явные знания в форме предпочтений); 2) «внешний» (физический) контекст взаимодействия (местоположение, время, устройство и т.д.); 4) «внутренний» контекст взаимодействия (пользовательская задача: цель (цели), требования к результату и ограничения).

Ситуационные аннотации сервисов предоставляются автоматически компонентом анализа использования системы для всех сервисов КПС, т.е. базовых и составных сервисов.

Компонент анализа использования системы хранит все *логи взаимодействия* (историю) и осуществляет их предварительную обработку для обнаружения семантических зависимостей сервисов от параметров ситуаций и последующего аннотирования сервисов КПС. Логи взаимодействия состоят из *пользовательских сессий*.

Элемент пользовательской сессии состоит из набора целей и цепочки сервисов, вызванных для заданного набора целей:

$< G_i, S_i >$ – элемент пользовательской сессии,

где $G_i = \{ g_{i1}, g_{i2}, \ldots, g_{in} \}$ – множество целей, g_{ij} – термин онтологии целей, j=1..n;

$S_i = (s_{i1}, s_{i2}, \ldots, s_{im})$ – вектор вызванных пользователем сервисов для заданного набора целей G_i.

Для сессии определяются также параметры пользователя и контекста. Параметры пользователя хранятся персистентно в виде профилей, контролируемых промежуточным программным обеспечением, а параметры контекста всегда динамически определяются для пользовательской сессии. Для упрощения мы будет считать, что параметры контекста остаются неизменными в течение пользовательской сессии. В противном случае, если что-то изменяется в течение сеанса работы с системой (например, используемое устройство), будем считать это началом новой сессии.

$U = \{ u_1, u_2, \ldots, u_n \}$ – параметры пользователя, u_i – логическое выражение, основанное на термине u_i' онтологии пользовательских профилей, i = 1..n;

$C = \{ c_1, c_2, \ldots, c_m \}$ – параметры контекста, c_j – логическое выражение, основанное на термине c_j' онтологии контекстов, j = 1..m;

$Session = \{ (U, C, G_1, S_1), (U, C, G_2, S_2), \ldots, (U, C, G_k, S_k) \}$ – пользовательская сессия.

Некоторые параметры пользователя могут быть предоставлены пользователем явно с использованием специального интерфейса, другие же могут быть получены, например, через оператора сотовой связи в случае использования мобильного телефона – местоположение, время и т.д. Все параметры контекста предоставляются так называемыми датчиками контекста.

Для обеих групп параметров (неявных параметров пользователя и параметров контекста) требуется разработка специальных механизмов привязки (*bindings*), которые позволяют отображать входные данные от датчиков в термины соответствующих онтологий. Например, географические координаты представляют собой контекст первого уровня – данные, полученные напрямую с датчиков. А соответствующий этим координатам городской квартал, экземпляр онтологии контекстов, добавляется к описанию контекста текущей ситуации использования, строя таким образом контекст второго уровня.

Алгоритм поиска паттернов сервисов и их семантических аннотаций. С использованием методов поиска ассоциативных правил (для анализа частоты вызова некоторого сервиса A в совокупности с параметрами $u_i/c_i/g_i$) и логического вывода на онтологиях (позволяющего вывести неявные знания на основе, главным образом, отношения включенности), рассматриваемое промежуточное программное обеспечение предоставляет семантические ситуационные аннотации для заданного сервиса A (рис. 2).

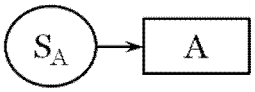

Рис. 2. Семантическая ситуационная аннотация для сервиса A

$S_A = S_{A1} \vee S_{A2} \vee \ldots \vee S_{AQ}$ – представляет собой предусловия для надлежащего использования сервиса A. Здесь $S_{AQ} = < G_{AQ}, U_{AQ}, C_{AQ} >$ – конъюнкция целей и логических выражений, основанных на терминах онтологии пользовательских профилей и контекстов.

Первым шагом данного алгоритма является фильтрация пользовательских сессий с целью получения множества векторов:

$$\text{Session}(A) = \{ (U^i, C^i, G^i, A) \},$$

где $i = 1..n_A : \exists j, k: (U_j, C_j, G_{jk}, A) \in \text{Session}_j$; n_A – общее количество фрагментов сессий, содержащих сервис A.

Затем выполняется поиск ассоциативных правил с использованием вариации классического алгоритма Apriori.

На входе алгоритма мы имеем:

- набор транзакций – Session(A);
- предметный набор – $U \cup C \cup G \cup S$ – все параметры пользовательского профиля, контекста, цели и сервисы.

На выходе алгоритма получаем ассоциативные правила, содержащие только выражения с параметрами пользовательского профиля, контекста и целями в левой части и сервис A в правой части (рис. 2).

Полезные композиции сервисов. С использованием методов поиска ассоциативных правил и последовательных шаблонов логи использования сервисов системы предварительно обрабатываются с целью получения двух типов условных зависимостей между сервисами:

- совместное существование в пользовательской сессии (рис. 3);
- последовательный вызов в пользовательской сессии, когда данные, полученные на выходе первого сервиса, используются на входе второго (рис. 4).

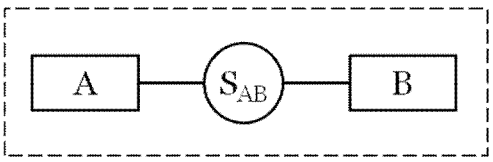

Рис. 3. Совместное существование сервисов A и B в пользовательской сессии

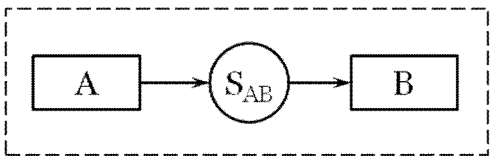

Рис. 4. Последовательный вызов A → B в пользовательской сессии

Здесь S_{AB} представляет собой описание ситуации, влияющее на заданную зависимость между сервисами, $S_{AB} = < G_{AB}, U_{AB}, C_{AB} >$.

Первый шаг состоит в поиске симметричных правил (рис. 3). Он осуществляется путем применения алгоритма поиска ассоциативных правил со следующими входными параметрами:

- набор транзакций, каждая из которых соответствует отдельной пользовательской сессии: $T_i = \{ S_{1i}, S_{2i}, \ldots, S_{ki} \}$, где $\exists i$: $Session_i = \{ (U_i, C_i, G_{1i}, S_{1i}), (U_i, C_i, G_{2i}, S_{2i}), \ldots, (U_i, C_i, G_{ki}, S_{ki}) \}$;
- предметный набор I – множество доступных сервисов.

На выходе алгоритма на этом шаге получаем множество правил вида $i_n \rightarrow i_m$, где $i_j \in I$. Правила $i_n \rightarrow i_m$ и $i_m \rightarrow i_n$ объединяются в одно правило, поддержка которого больше.

На следующем шаге применяется алгоритм для поиска ситуационных зависимостей для найденных симметричных ассоциативных правил. На входе имеем:

- набор транзакций, каждая транзакция соответствует найденному правилу:

$T_k = (U_k, C_k, G_{ik}, S_{ik}, G_{jk}, S_{jk})$ – транзакция, такая, что

$\exists p$: $(U_k, C_k, G_{ik}, S_{ik}) \in Session_p$ и $(U_k, C_k, G_{jk}, S_{jk}) \in Session_p$,

$S_{ik} \rightarrow S_{jk}$ or $S_{jk} \rightarrow S_{ik}$ – правило, найденное на предыдущем этапе;

- предметный набор – $U \cup C \cup G \cup S$ – все параметры пользовательского профиля, контекста, цели и сервисы.

На выходе алгоритма получаем правила, содержащие для каждого правила только цели и логические выражения на основе параметров пользовательского профиля и контекста в левой части и (A, B) в правой части для каждого симметричного правила A → B. Графическое представление каждой такой пары ассоциативных правил показано на рисунке 3.

В пользовательской сессии два сервиса могут быть вызваны непоследовательно, однако потоки данных с выхода одного и входа другого могут быть при этом сильно коррелированы. Анализ таких корреляций для правил, найденных на предыдущем этапе (рис. 3), позволяет получить правила, представленные графически на рисунке 4.

Затем такие правила интегрируются в КПС.

Для поиска более сложных сервисов (рис. 5) используется похожая процедура. На входе алгоритма имеем:

- набор транзакций, каждая транзакция соответствует пользовательской сессии;
- предметный набор, каждый предмет соответствует найденному шаблону последовательного вызова двух сервисов.

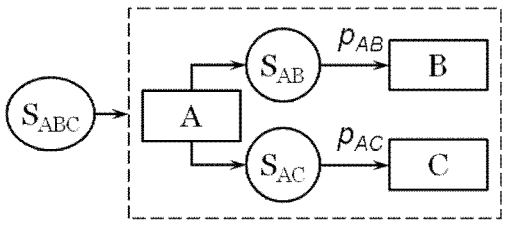

Рис. 5. Составной сервис с семантическими аннотациями

- S_{ABC}, S_{AB}, S_{AC} – описания ситуаций, $S_{ABC} \cap S_{AB} = \varnothing$ и $S_{ABC} \cap S_{AC} = \varnothing$, в свою очередь, S_{AB} и S_{AC} могут пересекаться;
- p_{AB} и p_{AC} – вероятности переходов A→B и A→C соответственно.

На выходе алгоритма поиска ассоциативных правил получаем правила, содержащие в левой и правой частях по одному шаблону. Возможны два варианта: когда эти шаблоны имеют один и тот же сервис в левых частях импликаций (A→B и A→C) и когда один и тот же сервис является посылкой одной импликации и заключением другой (A→B и B→C).

Граф, представленный выше (рис. 5), является комбинацией контекстуального графа, интенционального графа и байесовской сети.

• Контекстуальный граф [2] является контекстно-зависимым представлением выполнения задачи. Путь на графе (от входных вершин графа до выходных) представляет собой практику (процедуру), способ решения задачи с использованием выбранных методов-вершин. Таких способов столько, сколько путей на контекстуальном графе.

• Интенциональный граф или карта намерений/стратегий [6] – это ориентированный граф, вершины которого представляют намерения, а дуги – стратегии. Входная дуга вершины определяет стратегию, которая может быть использована для достижения намерения, определяемого этой вершиной. Карта (граф), таким образом, показывает, какие намерения могут быть достигнуты с помощью каких стратегий при условии, что предыдущее намерение достигнуто.

• Байесовская сеть [1] – это вероятностная графическая модель, которая представляет собой множество переменных и их условных зависимостей в виде ориентированного ациклического графа.

Список литературы

1. Bayesian network. – Режим доступа: http://en.wikipedia.org/wiki/Bayesian_ network, свободный. – Заглавие с экрана. – Яз. англ.

2. Brezillon P. Task Realization Models in Contextual Graphs / P. Brezillon // 5th International and Interdisciplinary conference CONTEXT – 2005. – Verlag, 2005. – Vol. 3554, Springer. – P. 55–68.

3. Ishkina E. Collective Service Intelligence Management in Mobiquitous Systems / E. Ishkina // The Sixth International Conference on Internet and Web Applications and Services (ICIW, 2011). – St. Maarten, The Netherlands Antilles, 2011.

4. OWL-S: Semantic Markup for Web Services. – Режим доступа: http://www.w3.org/Submission/OWL-S, свободный. – Заглавие с экрана. – Яз. англ.

5. RESTful Web services: The basics. – Режим доступа: https://www.ibm.com/ developerworks/webservices/library/ws-restful, свободный. – Заглавие с экрана. – Яз. англ.

6. Rolland C. Capturing System Intentionality with Maps / C. Rolland // Conceptual Modelling in Information Systems Engineering. – 2007, Springer. – P. 141–158.

7. SOAP W3C specification. – Режим доступа: http://www.w3.org/TR/soap, свободный. – Заглавие с экрана. – Яз. англ.

8. Web Application Description Language. – Режим доступа: http://www.w3.org/Submission/wadl, свободный. – Заглавие с экрана. – Яз. англ.

9. Web Service Modeling Ontology. – Режим доступа: http://www.wsmo.org, свободный. – Заглавие с экрана. – Яз. англ.

10. WSDL W3C specification. – Режим доступа: http://www.w3.org/TR/wsdl20, свободный. – Заглавие с экрана. – Яз. англ.

References

1. Bayesian network. – Rezhim dostupa: http://en.wikipedia.org/wiki/Bayesian_ network, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

2. Brezillon P. Task Realization Models in Contextual Graphs / P. Brezillon // 5th International and Interdisciplinary conference CONTEXT – 2005. – Verlag, 2005. – Vol. 3554, Springer. – P. 55–68.

3. Ishkina E. Collective Service Intelligence Management in Mobiquitous Systems / E. Ishkina // The Sixth International Conference on Internet and Web Applications and Services (ICIW, 2011). – St. Maarten, The Netherlands Antilles, 2011.

4. OWL-S: Semantic Markup for Web Services. – Rezhim dostupa: http://www.w3.org/Submission/OWL-S, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

5. RESTful Web services: The basics. – Rezhim dostupa: https://www.ibm.com/ developerworks/webservices/library/ws-restful, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

6. Rolland C. Capturing System Intentionality with Maps / C. Rolland // Conceptual Modelling in Information Systems Engineering. – 2007, Springer. – P. 141–158.

7. SOAP W3C specification. – Rezhim dostupa: http://www.w3.org/TR/soap, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

8. Web Application Description Language. – Rezhim dostupa: http://www.w3.org/ Submission/wadl, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

9. Web Service Modeling Ontology. – Rezhim dostupa: http://www.wsmo.org, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

10. WSDL W3C specification. – Rezhim dostupa: http://www.w3.org/TR/wsdl20, svobodniy. – Zaglavie s ekrana. – Yaz. angl.

УДК 004.031.2:37+004.424

СПОСОБЫ ПРОВЕРКИ РЕШЕНИЙ ЗАДАНИЙ ПО ПРОГРАММИРОВАНИЮ В ОБУЧАЮЩИХ СИСТЕМАХ

***Катаев Александр Вадимович**, ассистент, Волгоградский государственный технический университет, 400131, Россия, г. Волгоград, пр. Ленина, 28, e-mail: alexander.kataev@gmail.com.*

***Шабалина Ольга Аркадьевна**, кандидат технических наук, доцент, Волгоградский государственный технический университет, 400131, Россия, г. Волгоград, пр. Ленина, 28, e-mail: O.A.Shabalina@gmail.com.*

***Камаев Валерий Анатольевич**, доктор технических наук, профессор, Волгоградский государственный технический университет, 400131, Россия, г. Волгоград, пр. Ленина, 28, e-mail: kamaev@cad.vstu.ru.*

При реализации обучающих систем одна из наиболее сложных возникающих задач – задача тестирования и оценки результатов обучения. В системах, обучающих программированию, решения заданий представляют собой фрагменты программного кода, записанные на некотором языке программирования. Для верификации, анализа и тестирования программ существует специализированные инструменты, однако их использование для проверки решений в обучающих системах не всегда приемлемо, поскольку они ориентированы на анализ промышленного кода. В статье изложены особенности программного кода как вида решений заданий и предложены способы проверки такого вида решений. Рассмотрены применение способов проверки текста (исходного кода) решений и результаты работы кода решения. Описаны особенности программного кода, которые снижают эффективность применения методов обработки текста, традиционно применяемых для проверки решений в обучающих системах. Предложены методы предварительной обработки исходного кода (нормализации текста решения), позволяющие повысить эффективность работы алгоритмов проверки текста решения. Предложено применять регулярные выражения для хранения множества эталонных решений и выполнения проверки текста решения. Рассмотрены варианты реализации проверки результата работы программного кода – выполнение кода с использованием существующего интерпретатора или же с использованием специально созданного интерпретатора, ориентированного на применение в обучающей системе. Предложен способ дополнения фрагментов программного кода до полноценной программы с использованием заготовленных фрагментов кода (контекста трансляции).

***Ключевые слова:** автоматизированные обучающие системы, проверка заданий, оценка уровня знаний, программный код, верификация программного обеспечения, модульное тестирование, регулярные выражения, нормализация программного кода, контекст трансляции, выполнение программного кода.*