ПРОВЕРКА НА ЗНАНИЯТА И АВТОМАТИЧНО ОЦЕНЯВАНЕ ЧРЕЗ ИНТЕГРИРАНЕ НА РАЗНОТИПНИ СИСТЕМИ С РАБОТНИ ПРОЦЕСИ

Георги Пашев, Евгения Алендарова, Георги Тотков

ПУ „Паисий Хилендарски",
georgepashev@gmail.com, ealendarova@gmail.com, totkov@uni-plovdiv.bg

***Резюме**: Представен е метод за интеграция на разнотипни системи чрез система за работни процеси. Предложена е методология за решаване на фундаменталните предизвикателства пред интеграцията на разнотипни системи. Описана е многослойна абстрактна архитектура, която поддържа интеграция на разнотипни системи чрез система за работни процеси. Създадена е библиотека, написана на програмния език PHP (Hypertext Preprocessor), наречена Tiny Workflow Management PHP Framework (TineWfFm). Всяко състояние на работния процес е свързано с уеб метод от разнотипни системи. Представен е прототип на работен процес за студент и преподавател.*

***Ключови думи**: работен процес, разнотипни системи, интеграция, портал.*

Въведение

Използването на работни процеси в компютърните технологии е широко разпространено. Следвайки еволюцията на софтуерните технологии, организации влагат усилия за модернизация и нови методи за разработка, с цел подобрение на обучението и работните условия. Интеграцията представлява процес, който обединява разнотипни системи посредством средства за комуникация и визуализация.

Фундаменталните проблеми пред интегацията са свързани с комуникакцията между системите, ресурсите могат да бъдат трудно достъпни заради големи географски разстояния, фирмени политики или софтуерни ограничения. За спомагане на стандартизирането на методите за интеграция на разнотипни системи и ресурси е необходимо обособяване на основни правила [4].

Бизнес процесът позволява систематизиране на ресурси в процес, който доставя услуги или обработва информация. Работният процес представлява шаблон за повтарящи се бизнес процеси [5]. Може да бъде представен като поредица от операции, извършвани от човек или група от хора [7], организация на дейности или сложни механизми на работа.

WfMS е софтуерна система за настройка, изпълнение и наблюдение на дефинирани поредици от задачи, организирани като работен процес [6].

Поставени са следните основни задачи: интеграция на разнотипни системи, интеграция на потребителски роли и права, общ достъп до множество приложения.

Направено е проучване на съществуващите решения. Представена е систематизирана информация относно резултата от проучването.

Първи подход: бизнес логиката да бъде заложена в кода на приложението. Този подход има очевидни недостатъци като сериозни затруднения при поддръжка на приложения в среда, която се променя динамично.

Втори подход: предполага използване на вече съществуващи системи за работа с бизнес процеси. Основния недостатък на този подход е необходимостта да се вложат усилия за създаване на специфични методи за комуникация между отделните системи. Ненужната функционалност на този тип системи трябва да бъде премахната с цел повишаване на производителността.

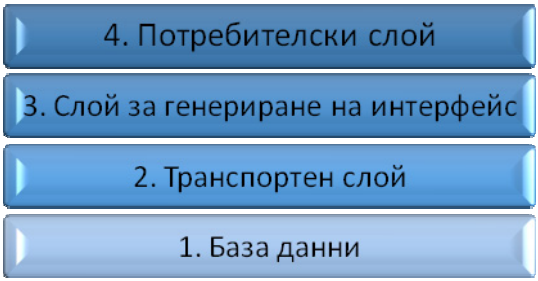
Трети подход: създаване на собствена система за управление на бизнес процеси, която да позволява гъвкави методи на работа - динамично управление (в реално време), минимални усилия при интегриране на системите, добра производителност.

Докладът предлага абстрактна архитектура (методология) за интеграция на разнотипни системи. Представената архитектура е изградена на база на третия подход. Описана е създадената система за работни процеси.

Архитектура

Предложената архитектура има 4 слоя, които са представени на фиг. 1. (от долу на горе).

- База данни - релационен модел на база данни, съдържащ референции, съхранени процедури, изгледи и функции. Заявките към базата данни, изпратени от по-горните слоеве, са разположени в този слой с цел постигане на най-висока скорост на изпълнение на заявка. Приложен е принципът за най-близкото разположение между данните и тяхната обработка, който се използва при алгоритми за оптимизация [2];
- Транспортен слой - представлява връзка между слоя с базата данни и слоя за генериране на интерфейс чрез REST или SOAP протокол;
- Слой за генериране на интерфейс - дефиниция на работните процеси. За дефиницията на възлите на бизнес процеса се използват URL шаблони;
- Потребителски слой - предоставя достъп до процеси от системата за работни процеси в зависимост от потребителски роли, права и лични настройки.



Фигура 5. Многослойна архитектура за интеграция на разнотипни системи

Слой 1. База Данни

Има утвърдени практики при изграждането на бази данни. Първият аспект е избора на модел. В текущата разработка е избран релационен модел. Структурата е в трета нормална форма [1].

За улеснение на процеса на генериране на интерфейс в трети слой е приета конвенция за именуване на обектите в базата данни. Името на полето, което представлява външен ключ към дадена таблица започва с „ID_" и след това се изписва името на свързаната таблица.

Потребителските действия, свързани с данните в базата данни, са капсулирани в съхранени процедури, изгледи и функции (фиг. 2.).

Голяма част от бизнес логиката е съхранена по подобен начин. Това позволява на по-горните слоеве да се фокусират върху типовете дейности, които са по-подходящи за тяхното разположение.

Фигура 6. Обекти на слоя База Данни

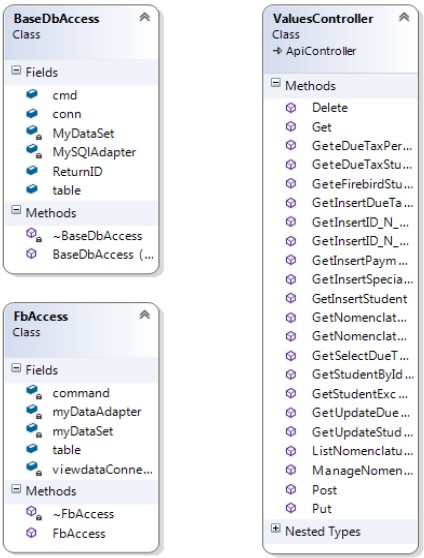
Слой 2. Транспортен слой

Транспортния слой представлява доставчик на услуги. Уеб услугата е метод за комуникация между две електронни устройства в мрежа. Представлява софтуерна функционалност, достъпна на мрежов адрес в уеб пространството. W3C дефинира уеб услугата като: софтуерна система, създадена да подпомага комуникацията между оперативно съвместими машини по мрежата [10].

Поддържат се протоколи като REST, SOAP и др.

Този слой представлява важна част от интеграцията, защото предоставя на третия слой уеб услуги за консумация. Те могат да бъдат от разнотипни приложения, което стои в основата на идеята за интеграция на разнотипни приложения.

Фиг. 3 представя UML диаграми, използвани в конкретен експеримент.



Фигура 7. UML клас диаграма на експериментални класове в транспортния слой

Слой 3. Слой за генериране на интерфейс

Крайни машини, крайни автомати или просто граф е математически модел за изчисление, използван за създаването на компютърни програми и последователни логически схеми. Замислен е като абстрактен механизъм, който може да бъде в едно от краен брой състояния. При този механизъм състояние в даден момент е само едно и се нарича текущо състояние. Може

да се преминава от едно състояние в друго при възникване на дадено събитие или условие. Това се нарича преход. Крайният автомат се дефинира като списък от неговите състояния. Основните термини в този контекст са възли и ребра. Те са логически свързани и се използват в текущата работа.

Работен процес W_f е наредена двойка $W_f = (\mathrm{N}, \mathrm{E})$, където:

$N = \cup N_i;\ i \in \overline{1,\ n}$ е множество от възли, $N_i = (R, F, F_r, r, r_r, D)$, $R = \cup R_i;\ i \in \overline{1,\ s};\ i, s \in \mathbb{N}; \mathrm{s} \geq \mathrm{i}$, където R е множество от входни данни, $D = \cup D_j;\ j \in \overline{1,\ k};\ j, k \in \mathbb{N}; \mathrm{k} \geq \mathrm{j}$, където D е множество от изходни данни, $F = \cup F_p;\ p \in \overline{1,\ t};\ p, t \in \mathbb{N}; \mathrm{t} \geq \mathrm{p}$, където F е множество от релации за създаване на изходни данни, $\varphi = a\ F_p\ b$ – Релация: $a \subseteq R; b \subseteq R$, $\varphi = \{(\alpha_1, \dots, \alpha_n)\};\ \alpha_i \in D; i \in \overline{1,\ n}$, $F_R = \cup F_{R_l}$, където F_Rе множество от релации за създаване на входни данни, $\beta = a_R\ F_{R_l}\ b_R: a_R \subseteq D;\ b_R \subseteq D;\ \beta = \{(\gamma_1, \dots, \gamma_t)\}: \gamma_i \in R; i \in \overline{1,\ t},\ r = f(d, m); d \in [0, 1] -$ посока, $m \in [0, n-1]: n = |F|$; r: код за избор на релация, $r \in \cup F_p,\ r_R = f_1(d_R, m_R);\ d_R \in [0, 1];\ m_R \in [0, q-1]: q = |F_R|;\ r_R$: код за избор на обратна релация,
$E_v = (D, \mu, R);\ \mu: D \longmapsto R;\ \mu$: биекция, $E = \cup E_v; v \in \overline{1,\ w}; v, w \in \mathbb{N}; w \geq v$.

Работен процес в текущата реализация се дефинира като ориентиран мулти-граф, дефиниран като двойка

$$\boldsymbol{Wf} = (\boldsymbol{NWf}, \boldsymbol{EWf}) \tag{1}$$

където NWf е множеството от възли $\boldsymbol{NWf}_{\boldsymbol{i}}$ на работния процес, а EWf е множеството от ребра $\boldsymbol{EWf}_{\boldsymbol{i}}$.

Ребро е дефинирано като n-торка по следния начин:

$$\boldsymbol{NWf}_{\boldsymbol{i}} = \begin{pmatrix} \boldsymbol{Name}, \boldsymbol{Caption}, \boldsymbol{urlTemplate}, \boldsymbol{extraSettings}, \boldsymbol{webAPIType}, \\ \boldsymbol{nodeTrigger}, \boldsymbol{webAPIHttpMethodType} \end{pmatrix} \tag{2}$$

Name е текстов атрибут, съдържащ името на желания уеб метод, който се извиква, когато крайният автомат влезне в текущо състояние (възел). В контекста на TinyWfFrm влизането в състояние на краен автомат е придружено от извикване на поне един уеб метод.

Caption е текстов атрибут, който се показва на потребителя по време на визуализация на формата на текущото състояние.

urlTemplate е съставен текстов атрибут, който съдържа низ URL, представляващ заявката към уеб метода и наредена n-торка от шаблонни полета tF.

$$\boldsymbol{tF = (fieldName, fieldCapion, fieldRegularExpression, fieldDefaultValue, fieldTrigger, isHidden, hierarchyParent)} \quad (3)$$

fieldName е името на полето. Генерира се автоматично от генератора на API (Application Program Interface – програмен интерфейс) в транспортния слой.

fieldCaption е заглавието (етикета) на полето.

fieldRegularExpression е регулярен израз от тип RegExp, който, ако е дефиниран, се използва за проста валидация.

fieldDefaultValue е стойността по подразбиране, която е попълнена по време на зареждане на формата, ако е попълнена.

fieldTrigger е тригер за сложна валидация, дефиниран като анонимна PHP функция.

isHidden е Булев атрибут, който управлява видимостта на потребителя.

hierarchyParent е текстов низ, който съдържа името на родителското поле, ако е налична йерархия. Атрибутът има незадължителен характер.

$$\boldsymbol{extraSettings = (rememberLastState, autoSubmit, autoSubmit_1)} \quad (4)$$

където rememberLastState е Булев атрибут, който запазва текущо състояние като сесийна променлива; autoSubmit е Булев атрибут, който задейства автоматично предаване на входните на формата; autoSubmit1 е Булев атрибут, който задейства автоматично предаване на изходните данни чрез първото ребро, участващо в списъка на изходящите ребра (преходи) за текущия възел (състояние).

$webAPIType$ е текстов атрибут, който съдържа информация за текущо използвания протокол на транспортно ниво за извикването на текущия уеб метод, например SOAP, REST и др.

$webAPIHttpMethodType$ е текстов атрибут, съдържащ HTTP метода, използван за извикването на уеб метода (GET, POST, PUT, DELETE).

$nodeTrigger$ е тригер за сложна валидация, дефиниран като анонимна PHP функция със следния прототип function($userRolesList, $currentState, $lastStateTransition, $previousState). Ако функцията върне отрицателен резултат, следващото състояние се изключва от списъка. $\boldsymbol{EWf_i}$ е ребро в граф топологията, формално дефинирано като наредената тройка:

$$\boldsymbol{EWf_i = (startNode, endNode, mappingsList)} \quad (5)$$

Значението на $\boldsymbol{startNode}$ е целочислен идентификатор на входящ възел на реброто. Значението на $\boldsymbol{endNode}$ е целочислен идентификатор на изходящ възел на реброто на насочения мулти граф $\boldsymbol{TinyWf}$. $\boldsymbol{mappingsList}$ е множество от

$$\boldsymbol{mapping = (inputFieldName, outputFieldName)} \quad (6)$$

Всяко свързване логически съчетава стойността на изходно поле от предишно състояние към стойност на следващо състояние на прехода. По този начин се осъществява автоматично транфериране на стойности между състояния.

Слой 4. Потребителски слой

Основната задача изисква решение, което позволява едновременен достъп на потребител до разнотипни системи, на база на неговите роли и права. Списъкът от тези роли частично дефинира електронното портфолио на потребителя. Множеството от функционалности, респективно работни процеси, за дадено електронно портфолио трябва да бъде показано на потребителя по гъвкав и удобен начин.

Проучени са различни проекти и решения, които трябва да отговарят на тези изисквания. uPortal [8] е избран за прототипа заради следните си характеристики:

- базиран на отворени стандарати и Java, XML, XSLT, JSP and J2EE;
- използвана от милиони потребители;
- възможност за използване на API's;
- различни възможности за идентификация на потребители, включително база данни и LDAP;
- връзка с библиотеки и подобна конфигурация за интеграция като CAS;
- интегрира single sign-on;
- предоставя потребителски услуги от различни източници;
- поддръжка заявки към LDAP и релационни бази данни;
- предлага компоненти и фреймуърк за свързване на по-сложни идентификационни системи;
- мощна и гъвкава система за управление на потребителски групи;
- динамично базирани групи на потребителски атрибути;
- могат да се имплементират вложени групи;
- възможност за представяне на потребителя на фрагменти и портлети спрямо атрибутите на потребителя, връзки, участие в групи.

Прототип

Върху предложената архитектура е разработен прототип на система за проверка на знанията и автоматично оценяване. За реализацията на задачите за оценяване са използвани предоставените от Moodle REST APIs (уеб услуги) на тестов модул, модул за автоматично c++ оценяване и модул за изпращане на съобщения. За документиране на резултатите е използвана системата Jasper. JasperReports Server е сървър за репорти и справки, който осигурява отчетност и анализи, които могат да бъдат вградени в уеб или мобилно

приложение. Jasper предоставя REST и SOAP базирани уеб услуги и HTTP APIs за опростена интеграция с приложения и работни потоци.

На фиг. 4. е представен примерен работен процес с участието на студент. Студент получава оторизиран достъп до uPortal чрез автентификация през Аткивна директория. TinyWfFrm изпълнява работен процес, който генерира графичен интерфейс на база предоставените от Moodle REST APIs за достъп до електронни курсове на база на учебния план на студента (получен от REST услуги на студентска информационна система). Студентът избира дисциплина от списъка с дисциплини и получава списък с инстанции на изпитни модули, актуални към текущия момент. След решаването на изпитния модул, автоматично се изпраща съобщение към преподавател(ите), водещи съответната дисциплина.

Фигура 8. Работен процес на студент (пример)

Примерният работен процес с участието на преподавател започва с преминаване през автентификация и оторизация аналогично на студента. На преподавател се генерира списък с преподаваните от него дисциплини.

При избор на конкретна дисциплина излиза списък с актуалните към момента инстанции на изпитни модули. При избор на конкретна инстанция на изпитен модул, получава списък с изпратените решения на студенти. След преглеждане на конкретно решение, попълва рецензия (бележки) по желание. Системата автоматично генерира изпитен протокол (чрез Jasper REST API) и изпраща съобщение на студента (чрез модул за изпращане на съобщение).

Заключение

Представената многослойна абстрактна архитектура поддържа интеграция на хетерогенни системи с помощта на работни процеси. Дефинирането на

процеси чрез TinyWfFrm позволява лесна и бърза разработка. Поставените задачи са изпълнени.

Следват някои бъдещи цели за реализация:

- Машинно обучение в Слой 3, което да позволи умно преподреждане на възможните следващи преходи.
- Добавяне на абстактен подслой към слоя на БД (Бази от данни), който да позволи крос компилиране за различни СУБД (Системи за управление на Бази от Данни) чрез дефиниране на обекти на БД в платформено независим език и автоматично генериране на DDL, DML, DCL за целеви СУБД.
- Създаване на графичен редактор за дефиниране на работни процеси.
- Провеждане на експерименти за известни системи като Moodle, Jasper, DSpace и др.

Литература

1. Kroenke, D., Database Processing: Fundamentals, Design, and Implementation, Prentice-Hall, Inc., pages 130–144
2. Wolf, M., M. Lam, A data locality optimizing algorithm, Proceeding PLDI '91 Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation, ISBN:0-89791-428-7, Pages 30 – 44
3. Wright, David R. (2005). "Finite State Machines". CSC215 Class Notes. Prof. David R. Wright website, N. Carolina State Univ. Retrieved July 14, 2012.
4. Алендарова,Е., А.Трайков, Г.Тотков. За интегрирането на разнотипни информационни системи и ресурси (или от потребителския профил към потребителското е-портфолио), Сборник доклади, V национална конференция за студенти, докторанти и млади учени, гр. Пловдив, ISSN: 1314-9547, стр. 203-207
5. Business Process Management (BPM) Center of Excellence (CoE) Glossary https://www.ftb.ca.gov/aboutFTB/Projects/ITSP/BPM_Glossary.pdf, 2009
6. Introduction to workflow management systems "What is a workflow management system?" http://www.taverna.org.uk/introduction/what-is-a-workflow-management-system/
7. See e.g., ISO 12052:2006, ISO.org
8. uPortal wiki, https://wiki.jasig.org/display/UPM41/Home
9. uPortal Rendering Process, https://wiki.jasig.org/display/UPM40/User+Interface
10. "Web Services Glossary". W3C. 2004-02-11. Retrieved 2011-04-22.

AUTOMATED ASSESSMENT THROUGH INTEGRATION OF HETEROGENEOUS SYSTEMS WITH A WORKFLOW ENGINE

Abstract: *The paper presents a method for integration of heterogeneous systems through workflows. Fundamental problems concerning the problem are reviewed and methodology is*

proposed. The suggested multilayer abstract architecture supports integration of heterogeneous systems with a workflow engine. We have created a custom workflow PHP framework, named Tiny Workflow Management PHP Framework (TinyWfFrm). Each state node in the workflow is logically connected to a web method in heterogeneous systems. A prototype containing work-flow definitions for student and teacher users is presented.