INTEGRAREA APLICAȚIILOR SOFTWARE – STUDIU DE CAZ

Ciprian Iamandi **Daniela Saru** **Ștefan Mocanu**

dsaru59@yahoo.com

Universitatea „Politehnica" București

Rezumat: Integrarea aplicațiilor software continuă să reprezinte o provocare pentru specialiști și o necesitate pentru beneficiari, oferind multiple avantaje de natură tehnică, managerială și financiară. Soluțiile elaborate până în prezent se concretizează în platforme software ce permit utilizatorului să integreze în mod transparent atât date scrise în formate diferite, cât și aplicații eterogene din punct de vedere software/hardware. Această lucrare prezintă un studiu de caz al modului de integrare a aplicațiilor în domeniul bancar folosind facilitățile platformei IBM WebSphere InterChange Server.

Cuvinte cheie: integrarea aplicațiilor software, middleware, hub and spoke, obiect business, mapări de date, IBM WebSphere, Linux.

Abstract: Software application integration is still a technical challenge and a rewarding option for modern enterprises and their managers. Various systems that need to be linked together often reside on different operating systems, use different data formats and different computer languages. Specialized integration packages can manage heterogeneous systems in a transparent and efficient way, linking business processes across applications. This paper presents a software integration application case study focused on IBM WebSphere InterChange Server facilities.

Key Words: application integration, middleware, hub and spoke, business object, data mapping, IBM WebSphere, Linux.

1. Introducere

Dezvoltarea vertiginoasă a Internetului, a produselor software destinate piețelor de afaceri și a produselor *CRM (Customer Relationship Management)* a condus la apariția unor arhitecturi informatice complexe, ce permit oferirea serviciilor de tip *e-business* (comerț electronic, intreprindere virtuală etc.) utilizând tehnologii Internet pentru rezolvarea unor probleme de intreprindere (de firmă) sau pentru realizarea unor sarcini funcționale. Aceste arhitecturi integrează la nivelul întreprinderii atât aplicații software deja existente, cât și aplicații noi și asigură colaborarea lor cu alte aplicații, dispersate geografic și executate pe sisteme de tipuri diferite, folosind ca liant tehnologiile de tip *middleware* [3].

Extinderea domeniului de integrare la nivelul Internetului poartă numele generic de integrare a aplicațiilor de întreprindere, acronimul folosit în limbaj tehnic provenind din limba engleză: *EAI – Enterprise Application Integration*. EAI pune accentul pe integrarea aplicațiilor în contextul afacerilor *(business)*, ceea ce conferă tehnologiilor *middleware* tradiționale un rol foarte important la nivelul infrastructurii arhitecturii globale [1]. Pentru creșterea performanțelor sistemelor informatice și a eficienței procesului de afaceri, entitățile ce trebuie integrate nu mai sunt obiecte sau componente software ci aplicații software și date eterogene. Prin EAI, sistemele informatice ale intreprinderilor se mulează din ce în ce mai bine pe structura procesului de afaceri.

Deoarece problemele de integrare pot implica un volum mare de muncă, timp și bani, numeroase firme care inițiază un astfel de proiect apelează la pachete software specializate, create de producători de prestigiu.

Această lucrare prezintă rezultatele studierii unor aspecte legate de integrarea aplicațiilor cu ajutorul platformei IBM WebSphere InterChange Server [5], folosind ca exemplu domeniul bancar. Studiul de caz a fost gândit astfel încât să ilustreze cât mai elocvent facilitățile oferite de către produsul software utilizat, în corelație cu elementele arhitecturale și funcționale prezentate în secțiunile dedicate expunerii de principiu.

2. Modele de integrare a aplicațiilor

Într-un sistem integrat de aplicații business, apare invariabil problema comunicării între aplicații. De cele mai multe ori, structura sistemului ajunge să fie asemănătoare unui graf complet conex (figura 1) în care, la un număr de *n* noduri, numărul de muchii este de *n/2*(n-1)*. Dacă graful este și orientat, numărul de muchii se dublează, ajungând la *n*(n-1)*. Se observă astfel că numărul de conexiuni între aplicații crește exponențial, ajungându-se la ceea ce se numește

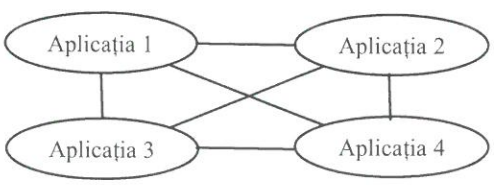

Figura 1. Integrare tip „spaghetti"

integrare de tip „spaghetti" [3].

Pentru rezolvarea acestei probleme, s-a elaborate un alt model, numit *Hub and Spoke*, care presupune existența unui element central (*hub*- centru, nucleu), la care se conectează toate celelalte elemente (*spoke*- rază). Într-o astfel de topologie (figura 2), numărul muchiilor este egal cu numărul nodurilor, ceea ce înseamnă că numărul de conexiuni creşte liniar cu numărul de aplicații, şi nu exponențial, ca în cazul anterior. Acest model arhitectural este preferat, în general, atunci când comunicarea între aplicații se face, predominant, pe bază de mesaje [4]. Mesajele pot fi stocate la nivelul *hub*-ului în şiruri de mesaje sau „fire de aşteptare".

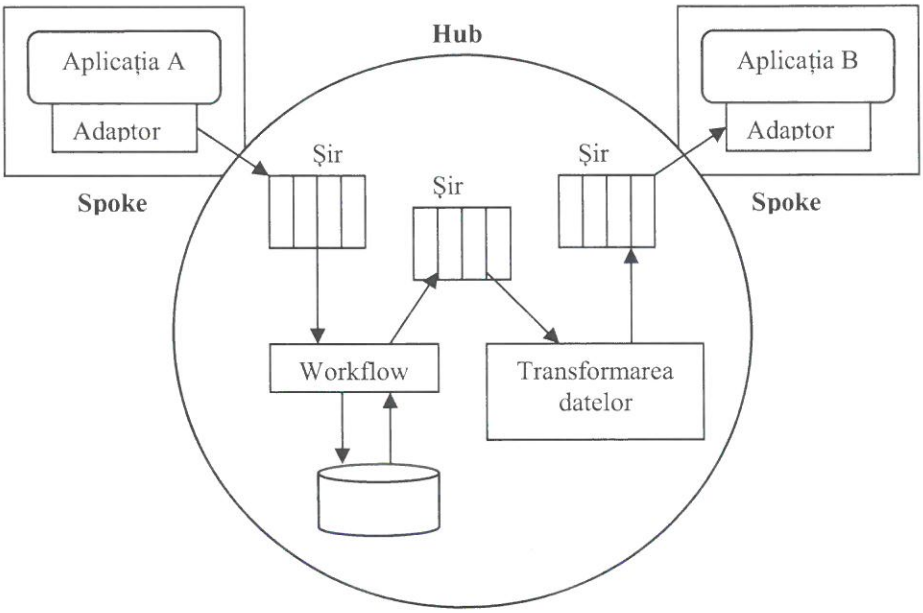

Figura 2. Arhitectura *„hub and spoke"*

Funcțiile oferite de către *hub* sunt, în general:

- *gestiunea firelor de aşteptare* – acestea sunt regrupate la nivelul *hub*-ului, nu descentralizate la nivelul fiecarei aplicații. Un modul de gestiune unică a firelor de aşteptare permite folosirea unor structuri de comunicație mai complexe: unu-la-unu (tip întrebare/răspuns) sau unu-la-mai-mulți (difuzare sau publicare/subscriere) [2];
- *înlănțuirea sarcinilor (workflow)* - această funcție este activată pentru fiecare mesaj emis în vederea stabilirii destinației sale care poate fi definită de către:
  - antetul mesajului – conține numele aplicației destinatar;
  - valoarea anumitor parametri din corpul mesajului – de exemplu, dacă mesajul reprezintă un bon de comandă, în funcție de valoarea conținută va fi trimis unor persoane diferite;
  - secvența de prelucrare de care aparține mesajul – mesajul aparține unui proces de afaceri descris intr-un tabel la care aplicația are acces; tabelul conține lista etapelor de prelucrare şi deci şi numele aplicației destinatar;
- *securitate şi drepturi de acces* – în cadrul acestui tip de arhitectură, este necesar să se verifice dacă o anumită aplicație are dreptul să trimită mesaje de un anumit tip unei alte aplicații; drepturile de acces ale fiecărei aplicații şi/sau fiecărui utilizator sunt păstrate intr-o bază de date conformă standardului LDAP *(Lightweight Data Access Protocol)* şi sunt verificate înainte de trimiterea mesajului către firul de aşteptare asociat funcției de transformare a datelor;
- *funcția de transformare a datelor* – este un program care se activează ori de câte ori apare un mesaj în firul de aşteptare ce-i este asociat; transformă datele din modelul folosit de aplicația emitentă a mesajului în modelul conform aplicației destinatar, generând un nou mesaj pe care îl depune în firul de aşteptare al celei de a doua.

O entitate *spoke (rază)* este legatura între *hub* şi una dintre aplicaţii. Are două componente: protocolul de reţea (care permite circulaţia mesajelor) şi adaptorul (o componentă software ad-hoc de conectare la aplicaţie).

Într-o arhitectură de tip *hub* şi *spoke*, se poate utiliza orice protocol de reţea care permite citirea sau stocarea mesajelor intr-un fir de aşteptare, situat cel mai adesea într-o bază de date. Structura mesajelor trebuie însă precizată [3].

Utilizarea unui *middleware* pentru integrarea a două aplicaţii ce doresc să schimbe informaţii în cursul execuţiei presupune conectarea acestor aplicaţii la *middleware* şi folosirea interfeţei specifice acestuia. Dacă aplicaţiile nu au fost concepute din start pentru a folosi un *middleware*, una dintre soluţiile posibile este crearea şi folosirea unui modul software specializat, numit adaptor [4]. *Adaptorul* asigură conectarea unei anumite aplicaţii la un anumit tip de *middleware şi* îi permite acesteia să formuleze (să iniţieze) sau să primească cereri. De aceea, adaptorul trebuie să îndeplinească, în general, următoarele funcţii (figura 3):

- comunicare cu aplicaţia prin intermediul API - fiecare dintre cele mai importante produse software comerciale dispune de una sau mai multe API bine definite; de exemplu, în cazul produsului SAP se pot menţiona BAPI şi iDOC;
- dacă o aplicaţie emite o cerere către o altă aplicaţie, adaptorul trebuie să afle prin intermediul API informaţiile legate de tipul cererii şi datele asociate; după aceea va construi mesajul în forma impusă de *middleware* (de exemplu JMS, XML etc.) şi îl va pune în firul de aşteptare situat în *hub*;
- dacă mesajul este primit de la *hub*, adaptorul trebuie să-l citească şi să-l transforme în unul sau mai multe apeluri de funcţie oferite de către API corespunzătoare aplicaţiei; în general, această transformare necesită o foarte bună cunoaştere a aplicaţiei şi constituie o sarcină deosebit de complexă pentru cel ce creează adaptorul.

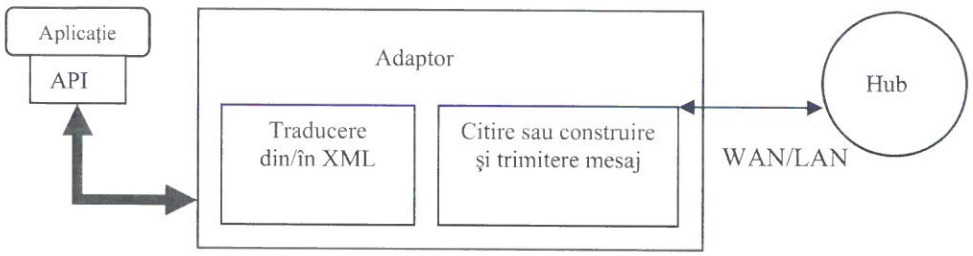

Figura 3. Structura unui adaptor

Integrarea aplicaţiilor implică dezvoltarea câte unui adaptor pentru fiecare dintre aplicaţiile ce urmează a fi integrate. În cazul în care o aplicaţie comunică nu cu una ci cu mai multe alte aplicaţii, mesajele pe care le va folosi vor varia în funcţie de destinatar. Va trebui, deci, ca adaptorul să le ia în considerare pe toate. Aceasta înseamnă ca un adaptor să scrie pentru o anumită aplicaţie, dar şi în funcţie de dialogul pe care aceasta urmează să îl poarte în cadrul sistemului.

Principalul motiv pentru care am prezentat mai multe detalii despre arhitecturile de tip *hub and spoke* în cadrul acestei secţiuni a lucrării este acela că, în studiul de caz propus, am utilizat un produs software ce se conformează acestei arhitecturi. Alegerea este justificată de specificul domeniului de integrare, aplicaţii bancare, în care logica business şi datele vehiculate sunt adecvate facilităţilor şi avantajelor oferite de arhitectura *hub and spoke*. Numite şi *tehnologii MOM (Message Oriented Middleware)* sau *middleware de tip „fir de aşteptare"*, aceste arhitecturi sunt, de multe ori, o alegere potrivită pentru realizarea sistemelor software în care aplicaţiile comunică prin schimb de mesaje. Asincrone prin însăşi natura lor, tehnologiile MOM elimină inconvenientul modului de lucru sincron impus, în general, de tehnologiile orientate obiect (CORBA, DCOM) [3], fiind preferate în cazul aplicaţiilor industriale şi de afaceri unde este esenţial ca mesajele să nu se piardă. Tehnologiile MOM decuplează, practic, aplicaţiile comunicante, mesajele fiind stocate până în momentul livrării către destinatar. Indisponibilitatea temporară a unei aplicaţii nu mai provoacă pierderea mesajelor, gradul ridicat de fiabilitate şi uşurinţa gestionării în mediu productiv recomandând de la sine acest gen de tehnologii pentru integrarea aplicaţiilor în cadrul întreprinderilor. Există, desigur, şi dezavantaje [1] [2] [5], care sunt legate mai ales de problema creşterii latenţei mesajelor dar şi de costuri. Elementul suplimentar (*hub*-ul) ce apare între două aplicaţii cooperante şi timpul necesar pentru luarea deciziei de rutare a mesajelor măresc durata realizării comunicaţiei. În plus, dacă *hub*-ul devine un punct de „gâtuitură", performanţele sistemului se vor degrada.

3. Prezentarea instrumentelor software utilizate

Pentru realizarea studiului de caz propus, a fost ales ca produs software reprezentativ IBM WebSphere InterChange Server. În cadrul acestei secțiuni, vor fi prezentate, pe scurt, câteva dintre principalele sale caracteristici, evidențiindu-se semnificația și rolul elementelor ce au fost folosite în cadrul implementării soluției de integrare.

La nivelul său cel mai înalt, un sistem de integrare WebSphere este compus dintr-un *broker* de integrare și un set de adaptori ce permit aplicațiilor business eterogene să schimbe date, sub formă de obiecte business, prin transfer coordonat de informații.

InterChange Server nu este singurul broker de integrare ce poate fi folosit ca piesă centrală a unui sistem de integrare WebSphere, alte variante fiind WebSphere MQ Integrator Broker, WebSphere Business Integration Message Broker sau WebSphere Application Server. Abordarea InterChange Server folosește o infrastructură distribuită, de tip *hub and spoke* ce permite integrarea aplicațiilor distribuite în rețeaua Internet, în rețea locală sau mixt [5]. Sistemul realizat este distribuit și flexibil, permite reutilizarea componentelor și asigură opțiuni de personalizare.

3.1. Componente funcționale de bază

Principalele componente funcționale ale IBM WebSphere InterChange Server sunt colaborările, obiectele business, conectorii și interfața de acces la server [5].

Colaborările InterChange Server sunt module software, care conțin logica de descriere a unui proces business distribuit. Colaborările formează de fapt *hub*-ul arhitecturii, coordonând și asigurând funcționarea corectă a proceselor business ce implică schimb de informații (sub formă de *obiecte business*) și colaborare între aplicații distincte. *Manipulatorii de date* transformă datele seriale produse de aplicații în obiecte business iar *mapările* pun în corespondență modelul de date specific unei aplicații cu modelul de date folosit de către colaborări la nivelul *hub*-ului.

Conectorii (cu rol de adaptor) oferă conectivitate cu aplicațiile (sau cu servere Web sau cu alte entități software) la nivelul *spoke*-urilor, putând fi configurați să interacționeze fie în rețea locală, fie prin Internet și *firewall*-uri și au structură distribută, fiind formați din două entitati: *conectorul controller* (interacționează direct cu colaborările și rulează ca o componentă în interiorul procesului InterChange Server) și *conectorul client* (un proces ce rulează separat de InterChange Server, Și, împreună cu o componentă specifică aplicației, interacționează direct cu aplicația (entitatea software) deservită. În această lucrare, vom considera conectorul client și componenta specifică aplicației drept o singură componentă, pe care o vom numi *agentul conector*. Există conectori proiectați pentru interacțiuni conforme cu standarde tehnologice specifice (de exemplu, conectorul XML) și conectori proiectați pentru a interacționa cu aplicații software specifice.

Interfața de acces la server este o API *(Application Programming Interface)* compatibilă CORBA [3] care acceptă transferuri sincrone de date atât din surse interne unei rețele, cât și din afara acesteia. Datele sunt apoi transformate în obiecte business, ce pot fi manipulate de către colaborări. Atât interfața de acces, cât și conectorii utilizează manipulatori de date ce pot fi creați dintr-un grup modular de clase de bază, *Data Handler Framework*. Soluția InterChange Server include și un *Protocol HandlerFramework*, oferind astfel suport pentru personalizarea și actualizarea soluțiilor.

O soluție tipică include una sau mai multe colaborări și un set de obiecte business ce reprezintă informație relevantă pentru o afacere. Colaborările și obiectele business sunt folosite cu conectori, cu interfața de acces la server sau cu ambele. Un conector poate interacționa cu una sau mai multe colaborări, ca urmare putând lua parte la diferite procese business. Fiecare colaborare poate interacționa cu orice număr de conectori, deci poate realiza comunicare cu oricâte aplicații.

Schimbul de informații poate însemna comunicare sincronă/asincronă între aplicații distincte situate în rețea locală/ în Internet, între aplicații din rețea locală și servere Web exterioare sau în cadrul unor procese business complexe ce pot include oricare dintre celelalte două variante. Schimbul de informații poate fi inițiat

ca urmare a unei interacțiuni de tip *publică-și-subscrie (publish and subscribe)* sau printr-o *cerere de acces* trimisă colaborării prin intermediul interfeței de acces la server. Ambele tipuri de interacțiuni oferă *trigger*-i ce pornesc execuția procesului business din colaborarea vizată. Colaborarea folosește apoi un al treilea tip de interacțiune, *cerere/răspuns*, pentru a finaliza schimbul de date cu destinația dorită.

Pentru a executa un proces business, o colaborare poate să comunice cu conectorii, cu alte colaborări, și cu procese externe de la care primește cereri de acces prin intermediul interfeței de acces la server. Toate aceste legaturi prin care se realizează comunicarea se stabilesc la momentul *configurării* colaborării prin operația de „legare" (*binding*). Pentru ca o colaborare să poată fi activată de un anumit eveniment (*trigger*), ea trebuie legată de un conector capabil să publice evenimentul respectiv. Componenta *controller* a conectorului pastrează informațiile despre legături și oferă agentului conector o listă cu colaborările care au subscris la eveniment. Atunci când aplicațiile efectuează operații relevante, agentul conector semnalează controller-ului evenimentul asociat listei iar acesta acționează ca intermediar între agentul conector și colaborare (agentul conector nu cunoaște destinația datelor). Trebuie menționat faptul că la un același eveniment pot subscrie mai multe colaborări; controller-ul conectorului poate publica evenimentul, simultan, către toți abonații. Dacă nu se dorește activarea unei colaborări prin legarea de un anumit conector, la momentul configurării se poate opta pentru varianta de legare (*binding*) la procese externe, care vor formula cereri de acces cu rol de *trigger*.

3.2. Obiecte business generice și specifice

Un obiect business este o entitate ce reprezintă o colecție de date care poate fi tratată unitar (de exemplu, toate câmpurile unui formular ce se dorește a fi folosit într-o aplicație pentru a îngloba informații despre clienți, angajați sau plăți) [1]. Obiectele business sunt preîncărcate în memoria de tip *cache* în timpul execuției colaborării, pentru a putea fi accesate rapid, și sunt stocate în stare persistentă, pentru a oferi posibilități de recuperare, *rollback*, și repetare a execuției colaborărilor în cazul repornirii serverului după o defecțiune.

Un obiect business poate fi perceput în sistemul InterChange Server ca reprezentând una dintre următoarele entități [5]:

- *eveniment* – atunci când un conector detectează apariția unui eveniment generat de o aplicație (o operație ce afectează o entitate de date prin creare, ștergere, modificare etc.) și trimite un obiect business unei colaborări, obiectul business reprezintă evenimentul (îl raportează), fiind tratat ca atare de către sistem;
- *cerere* – formularea cererilor prin intermediul obiectelor business se poate realiza astfel:
  - o colaborare trimite un obiect business unui conector, instruind conectorul să insereze, modifice, sau să șteargă date dintr-o aplicație;
  - interfața de acces la server trimite obiecte business, cu rol de cereri, unei colaborări, dacă acea colaborare a fost proiectată să accepte verbul „*retrieve*" ca *trigger;*
- *răspuns* - când conectorul încheie procesarea unei cereri, poate returna ca răspuns un obiect business (de exemplu, datele solicitate).

Structura unui obiect business evidențiază drept componente un tip (nume obiect), instrucțiuni de procesare (verb) și date (valorile atributelor). *Tipul* obiectului business este folosit pentru identificarea acestuia în cadrul sistemului InterChange Server și se definește, de obicei, ținând seama de semnificația obiectului pe care îl reprezintă (de exemplu, *Angajat, Contract* etc.). *Verbul* specifică ce acțiuni se vor realiza asupra valorilor atributelor, în funcție de rolul obiectului business respectiv (eveniment, cerere sau răspuns). Exemple de acțiuni posibile: *Create, Update, Retrieve* etc. *Valorile atributelor* reprezintă câmpuri de date (de exemplu, *Nume, Prenume, ID Angajat* sau *Stare Factură*). Unele atribute, în loc de date pot conține obiecte business „copii", sau chiar liste de obiecte business „copii". Obiectele business care conțin alte obiecte business, sau liste de alte obiecte business, se numesc *obiecte business ierarhizate*, în timp ce obiectele business care conțin numai date se numesc *obiecte business plate.*

În cadrul sistemului IBM WebSphere InterChange Server, se lucrează cu două categorii de obiecte business: *obiecte business specifice aplicației (Application Specific Object - ASBO)*, care reflectă atributele entității de date și modelul de date specifice unei anumite aplicații (entități software) și *obiecte business generice (Generic Business Object - GBO)*, care conțin un set de atribute comune pentru o mare varietate de aplicații, independente de modelul de date al vreuneia dintre ele. Agenții conectori preiau date de la aplicații și le transformă în ASBO. Controller-ul conectorului transformă ASBO în GBO și îl

trimite colaborării. Utilizarea GBO permite reutilizarea colaborării pentru versiuni diferite de aplicații, deoarece logica business nu mai este restricționată de particularitățile acestora. Cu alte cuvinte, modificările vor impune doar folosirea unui nou conector care să realizeze transformarea obiectelor din/în noul format specific aplicației în/din obiecte business generice. Colaborările interacționează folosind numai GBO. Ele trimit rezultatul controller-ului conectorului, care îl transformă în ASBO și, prin agentul conector, îl furnizează aplicației. Controller-ul conectorului transformă obiectele bussines din format ASBO în format GBO, atunci când obiectele trec de la conectori la colaborări, și din format GBO în ASBO, atunci când obiectele trec de la colaborări la conectori. Aceste transformări, numite și *mapări*, sunt descrise la momentul proiectării prin diagrame de corespondență. În cadrul studiului de caz (secțiunea 4), vor fi prezentate câteva exemple în acest sens.

3.3. Particularități ale componentei *hub*

Arhitectura sistemului IBM WebSphere InterChange Server conține la nivel de *hub* așa-numitele *colaborări*, module software ce reprezintă logica de descriere a procesului business distribuit. Acestea coordonează și asigură funcționarea corectă a schimbului de informații (sub formă de *obiecte business*) și a cooperării între toate aplicațiile distincte aflate la nivel de *spoke*. Colaborările pot fi simple sau complexe, pot include alte colaborări, pot fi distribuite peste un număr oricât de mare de aplicații, pot primi cereri de serviciu sincrone/asincrone, pot deservi procese business de durată mare. Această secțiune evidențiază doar câteva dintre particularitățile lucrului cu elemente de acest tip, mai exact pe cele care au fost folosite cu predilecție în implementarea asociată studiului de caz.

Instalarea unei colaborări în cadrul sistemului implică existența unui șablon de colaborare. Un *șablon de colaborare* [1] conține toată logica de execuție a colaborării, nu este executabil, dar permite crearea unui *obiect colaborare* care i se conformează. Obiectul colaborare devine executabil după ce este *configurat*, prin legare *(binding)* de conectori sau/și alte obiecte colaborare și prin specificarea unor proprietăți. Un același șablon de colaborare poate fi folosit pentru crearea mai multor obiecte colaborare care pot fi configurate apoi conform necesităților procesului business.

Pentru comunicare cu „lumea exterioară" (conectori sau alte colaborări), fiecărei colaborare folosește un set de interfețe, numite *porturi*. Fiecare port este asociat cu un anumit tip de obiect business, fiind reprezentat în șablonul de colaborare printr-o variabilă [5]. Pentru colaborările al căror șablon acceptă verbul *„Retrieve"* se poate configura un port ca *Extern*, astfel încât să poată primi obiecte business printr-o cerere de acces de la interfața de acces la server. Asocierea port-conector sau port-colaborare se face în momentul configurării. Administratorul poate lega un port numai la un conector ce suportă tipul de obiect business asociat sau numai la o alta colaborare ce conține un port care suportă acel tip de obiect business. Numele porturilor indică adeseori rolul pe care îl au în colaborare obiectele business ce trec prin el. De exemplu, dacă o colaborare primește informații despre client și generează informații despre contract, porturile ei pot fi numite *InCustomer* și *OutContract*.

În interiorul unei colaborări, unul sau mai multe *scenarii* conțin codul ce implementează un proces business. Fiecare scenariu specifică ce se întâmplă ca răspuns la primirea anumitor tipuri de obiecte business, ce semnificație au anumite evenimente. Relația dintre colaborări și scenarii este similară cu relația dintre programe tradiționale și rutine (subprograme, proceduri sau funcții etc.). În mod similar, pentru a proiecta și implementa colaborarea, pot fi folosite unul sau mai multe scenarii. Deși scenariile se execută independent unul față de altul, ele nu pot fi configurate și administrate independent, ci moștenesc toate configurările colaborării de care aparțin.

4. Studiu de caz

În prezent, integrarea aplicațiilor software joacă un rol major în domeniul bancar, fiind deosebit de importantă pentru buna funcționare a instituțiilor de acest tip. Deținând numeroase sisteme informatice, implementate la momente de timp diferite, pe platforme diferite și folosind tehnologii diferite, băncile rezolvă problemele legate de integrarea aplicațiilor apelând, de obicei, la soluții *Enterprise* dezvoltate de marile companii software. Deși foarte costisitoare [1], acestea au asigurat suport tehnic și beneficiază de actualizări regulate. Ținând cont de aceste aspecte, am folosit în cadrul studiului de caz facilitățile oferite de IMB WebSphere InterChange Server, urmând ca rezultatele obținute să fie valorificate în cadrul unei aplicații de integrare complexe care să stea la baza realizării disertației de absolvire a cursurilor de Master a primului autor al acestei lucrări.

Exemplul proiectat şi implementat are în vedere două componente: o aplicaţie *front-end,* folosită interactiv de operatorii băncii şi o aplicaţie *back-end,* care gestionează conturilor clienţilor şi plăţile.

Aplicaţia *front-end*, numită *Procesare Cecuri,* permite clienţilor băncii să efectueze plăţi cu ajutorul cecurilor şi a fost creată ca aplicaţie Web ce poate fi accesată folosind un *browser*, pe baza autentificării cu nume de utilizator şi parolă. În cadrul studiului de caz, aplicaţia a fost dezvoltată cu ajutorul tehnologiei de *Portal* (IBM) şi instalată pe WebSphere Application Server, rulând pe sistem de operare Linux Debian. Formularul principal al aplicaţiei a fost proiectat astfel încât să poată fi folosit pentru a cere utilizatorului informaţii despre seria şi numărul cecului, codul IBAN al beneficiarului şi al plătitorului, suma de plată, data plăţii, şi alte detalii despre plată.

Aplicaţia *back-end*, numită *Core Banking System*, reprezintă chiar sistemul informatic central al băncii, fiind implementată în limbajul RPG, pe platforma iSeries şi rulând pe sisteme AS400.

Integrarea celor două aplicaţii impune construirea unui nivel *middleware* adecvat. În cadrul studiului de caz, integrarea a fost făcută demonstrativ, doar la nivelul funcţionalităţii principale a aplicaţiei *front-end*, mai exact pentru operaţia de efectuare a unei plăţi cu ajutorul unui cec. Trebuie avut în vedere faptul că procesul real de integrare a unor astfel de aplicaţii este mult mai complex, dar principiile de bază sunt aceleaşi. Mediul de dezvoltare folosit pentru implementarea demonstraţiei a fost WebSphere Business Integration System Manager, împreună cu instrumentele asociate: Business Object Designer, Map Designer, Connector Configurator, şi Process Designer.

Prima etapă ce trebuie parcursă pentru realizarea integrării este stabilirea structurii obiectelor business ce urmează a fi folosite în fluxul informaţional. În cazul prezentat, acestea sunt: *Payment_ASBO,* care conţine în principal informaţiile luate din aplicaţia *front-end*, şi *Payment_GBO,* rezultatul mapării acestuia. Structura *Payment_GBO* este ceva mai complexă, deoarece va fi folosit în toate colaborările ce implică efectuare de plăţi. *Payment_GBO* conţine nu numai valori de tip primitiv, ci şi alte obiecte business, mai exact, obiecte de tip *Cont_GBO* (figura 4.). Acestea sunt tot obiecte compuse, conţinând obiecte business de tip *Persoana_GBO.*

General | Attributes

	Pos	Name	Type	Key	Foreign Key	Required Attribute	Cardinality	Maximum Length
1	1	Payment_ID	String	☑	☐	☐		100
2	2	⊞ ContPlatitor	Cont_GBO	☐	☐	☐	1	
3	3	⊞ ContBeneficiar	Cont_GBO	☐	☐	☐	1	
4	4	Suma	Double	☐	☐	☐		
5	5	DataPlatii	Date	☐	☐	☐		
6	6	DataEfectuarii	Date	☐	☐	☐		
7	7	ComisionTranzactie	Double	☐	☐	☐		
8	8	ComisionStornare	Double	☐	☐	☐		
9	9	ComisionRefuz	Double	☐	☐	☐		
10	10	ObjectEventId	String					

Figura 4. Structura obiectului *Payment_GBO*

La nivelul conectorului cu aplicaţia *back-end* (*Core Banking System)* obiectele GBO vor fi convertite în obiecte de tip *Payment_CBS* (figura 5.), ce conţin obiecte de tip *Cont_CBS.*

General | Attributes

	Pos	Name	Type	Key	Foreign Key	Required Attribute	Cardinality	Maximum Length
1	1	PaymentID	String	☑	☐	☐		255
2	2	⊞ ContPlatitor	Cont_CBS	☐	☐	☐	1	
3	3	⊞ ContBeneficiar	Cont_CBS	☐	☐	☐	1	
4	4	SumaTotala	Double	☐	☐	☐		
5	5	DataPlatii	Date	☐	☐	☐		
6	6	DetaliiPlata	String	☐	☐	☐		255
7	7	ObjectEventId	String					

Figura 5. Structura obiectului *Payment_CBS*

Transformarea obiectelor business din ASBO în GBO şi apoi din nou în ASBO se realizează prin intermediul *mapărilor* care permit specificarea corespondenţei unu la unu între câmpurile obiectelor şi a regulilor de transformare între câmpuri, în cazul în care datele trebuie prelucrate în prealabil. Pentru exemplul ales, au fost create două mapări: *Payment_ASBO_to_GBO* (figura 6), care specifică modul de transformare a unui obiect de tip *Payment_ASBO* într-un obiect de tip *Payment_GBO* şi *Payment_GBO_to_CBS*, care specifică modul de transformare a unui obiect de tip *Payment_GBO* într-un obiect de tip *Payment_CBS*. Transformările folosite în mod predominant au fost cele de tip *Move*, prin care valoarea unui câmp aparţinând obiectului sursă este copiată ca valoare în câmpul corespunzător din obiectul destinaţie. Transformările de tip *Split* au fost aplicate acelor valori care trebuiau prelucrate înainte de stabilirea corespondenţei între câmpuri. De exemplu, pe baza caracterului spaţiu pe care îl includea, valoarea câmpului *Nume_Platitor* a fost separată în două alte valori: *Nume* şi *Prenume*. Trebuie specificat faptul că, în general, se pot efectua şi transformări mai complicate decât *Move* sau *Split,* prin alegerea opţiunii *Custom,* care permite definirea unei logici mai complexe prin scriere de cod Java.

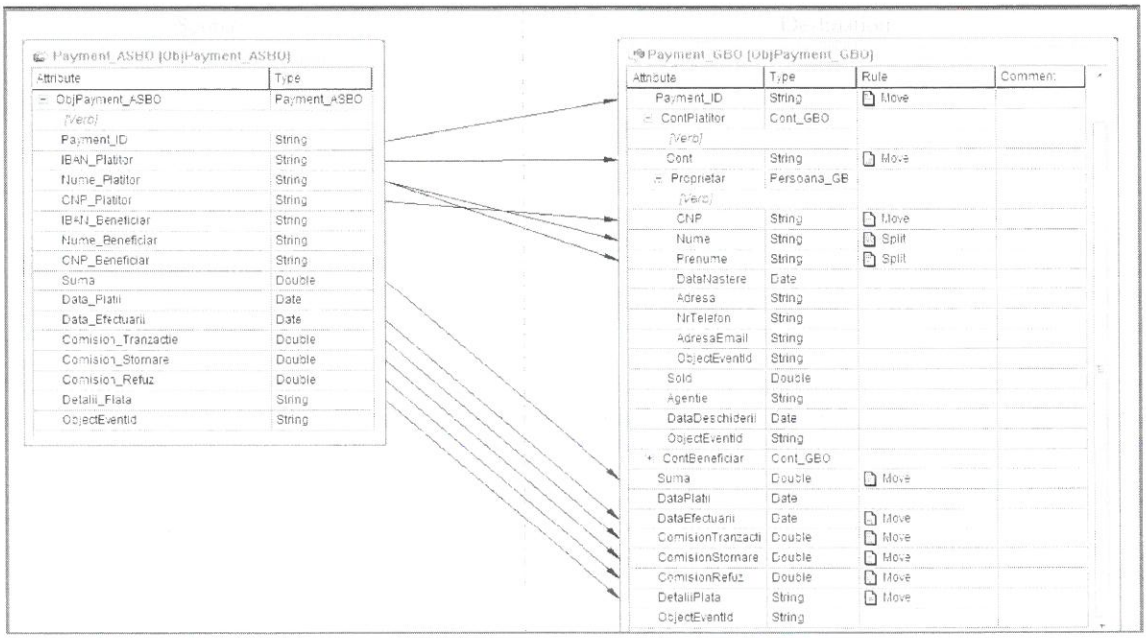

Figura 6. Maparea Payment_ASBO în Payment_GBO

A două etapă ce trebuie parcursă în cadrul procesului de integrare este crearea şi configurarea *conectorilor*. Pentru exemplul ales, a fost folosită în acest scop componenta *Connector Configurator,* din cadrul mediului de dezvoltare InterChange Server, care permite atât alegerea obiectelor ce vor fi acceptate, cât şi a mapărilor ce vor fi realizate între acestea. Conectorii vor putea rula numai pe WebSphere InterChange Server. În exemplul ales, pentru colaborarea *Payment* au fost creaţi doi conectori: unul care face legătura cu aplicaţia *front-end*, numit *APC_Connector* şi unul care face legătura cu aplicaţia *back-end*, numit *CBS_Connector*.

Cea de-a treia etapă a procesului de integrare este crearea *colaborărilor*. În exemplul ales, s-a lucrat cu o singură colaborare, numită *PaymentObject*, al cărui Şablon a fost creat cu ajutorul utilitarului *Process Designer*. Se reaminteşte faptul că *şablonul colaborării* conţine logica (grupată pe *scenarii*) şi *porturile* colaborării, împreună cu *tipul* de obiecte acceptate şi *verbele* lor. În plus, orice şablon de colaborare trebuie să aibă cel puţin un port de comunicare cu entităţi exterioare. Pentru şablonul proiectat, au fost definite două porturi (figura 7): un port numit *From*, pentru comunicaţia cu aplicaţia *front-end* şi un port numit *To*, pentru aplicaţia *back-end*. Au fost, de asemenea, definite două scenarii: scenariul unei plăţi simple, numit *Efectuare*, şi scenariul unei stornări (revocarea unei plăţi efectuate), numit *Stornare*. Cele două scenarii corespund verbelor *Create*, respectiv *Delete*. Colaborarea primeşte un obiect business de tip *Payment_ASBO* şi produce un obiect de tip *Payment_CBS*. Când obiectul primit are verbul *Create*, se va executa scenariul *Efectuare*; când verbul este *Delete*, se va executa scenariul *Stornare*.

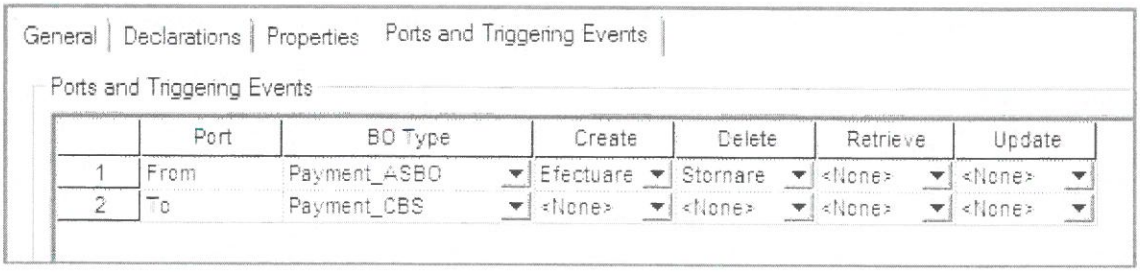

Figura 7. Definirea porturilor Şablonului de colaborare

Scenariile şablonului sunt foarte asemănătoare cu schemele logice, având drept elemente componente: puncte de pornire, blocuri de acţiune, blocuri de decizie, sub-scenarii, puncte de ieşire cu succes, şi puncte de ieşire cu eroare. Fiecare dintre aceste blocuri poate fi personalizat prin scriere de cod Java.

Pe baza şablonului de colaborare proiectat, a fost creat un obiect de tip colaborare, numit *PaymentObject*. La configurare, dintre cele trei posibile variante de entităţi ce puteau fi selectate (conector, altă colaborare, serviciu Web), pentru a fi asociate celor două porturi definite, s-a stabilit următoarea corespondenţă: pentru portul *From* s-a utilizat conectorul *APC_Connector*, iar pentru portul *To* s-a utilizat conectorul *CBS_Connector*. Mediul software folosit permite vizualizarea rezultatului atât sub formă de „arbore", cât şi din punctul de vedere al interacţiunii funcţionale (figura 8).

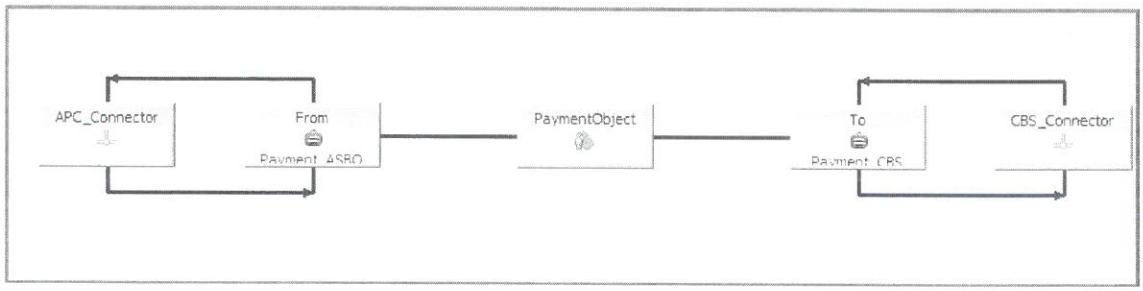

Figura 8. Vizualizarea interacţiunii funcţionale a colaborării

5. Concluzii

Integrarea aplicaţiilor software constituie, în continuare, un subiect de mare actualitate, având un rol major în eficientizarea activităţii la nivel de business în multe domenii. În cadrul acestei lucrări, au fost prezentate doar câteva dintre modelele şi soluţiile de integrare elaborate până în prezent, insistându-se asupra caracteristicilor şi facilităţilor evidenţiate de arhitecturile tip *hub and spoke*. Opţiunea este motivată de faptul că domeniul ales pentru realizarea implementării practice este cel bancar, în care specificul activităţii este adecvat utilizării tehnologiilor *MOM - Message Oriented Middleware* (*middleware* de tip *"fir de aşteptare"*).

În cadrul studiului de caz, integrarea a fost făcuta demonstrativ, doar la nivelul funcţionalităţii principale a aplicaţiei *front-end,* urmând ca în a două etapă de cercetare să se mărească gradul de complexitate. Finalitatea avută în vedere este realizarea lucrării de disertaţie (Master) a primului autor al acestui studiu. Mediul de dezvoltare folosit pentru implementarea demonstraţiei a fost *WebSphere Business Integration System Manager*, împreună cu instrumentele asociate: *Business Object Designer, Map Designer, Connector Configurator* şi *Process Designer*. Aplicaţia *front-end* (creată ca aplicaţie *Web*), numită *Procesare Cecuri,* permite clienţilor băncii să efectueze plăţi cu ajutorul cecurilor, pe baza autentificării cu nume de utilizator şi parolă. Aplicaţia a fost dezvoltată cu ajutorul tehnologiei de *Portal* (IBM) şi instalată pe WebSphere Application Server, rulând pe sistem de operare Linux Debian. Aplicaţia *back-end*, numită *Core Banking System*, reprezintă chiar sistemul informatic central al băncii, fiind implementată în limbajul RPG, pe platforma iSeries şi rulând pe sisteme AS400. Folosirea platformei IBM WebSphere InterChange Server a fost justificată şi de faptul că, pe lângă compatibilitatea sa cu sistemele de aplicaţii IBM, oferă şi interoperabilitate CORBA cu produse ale altor companii, prin interfaţa de acces la server.

Deoarece activitatea de cercetare are drept obiectiv elaborarea unei lucrări de disertație (Master), următoarele etape vor include studierea şi folosirea cât mai multor metode şi instrumente de integrare a aplicațiilor software, realizarea unui studiu comparativ bine fundamentat şi rezolvarea practică a unei situații complexe.

Bibliografie

1. **HOHPE, G., B. WOOLF:** Enterprise Integration Patterns - Designing, Building, and Deploying Messaging Solutions, Addison Wesley, 2003.
2. **CHAPPELL, D.:** Enterprise.Service.Bus, O'Reilly, 2004.
3. **SERAIN, D.:** Enterprise Application Integration. L'architecture des solutions e-business, Dunod, Paris, 2001.
4. **HOHPE, G.:** Hub and Spoke [or] Zen and the Art of Message Broker Maintenance (2007), documentație Internet, http://www.eaipatterns.com/ramblings/03_hubandspoke.html
5. ***: Technical Introduction to IBM WebSphere InterChange Server Version 4.3.0, IBM Corporation, 2004, documentație Internet, ftp://ftp.software.ibm.com/