Proiectarea Sistemelor Software Complexe

Curs 7 –Brocări de Mesaje. Modelarea Proceselor Business. Integrarea Arhitecturilor Eterogene

7.1 Brocări de Mesaje

Tehnologiile middleware bazate pe cozi sau pe principiul publică-subscrie pot fi folosite pentru proiectarea multor modele de comunicare. Aceste două tehnologii sunt simple, eficiente și oferă un grad ridicat de performanță și fiabilitate. Apar însă o serie de probleme atunci când trebuie integrate aplicații care folosesc tipuri de mesaje diferite, acest lucru se întâmplă frecvent atunci când se dorește construirea de sisteme software complexe prin interconectarea unor sisteme software mai vechi. Pentru rezolvarea problemelor care apar la integrarea sistemelor software complexe se poate folosi tehnologia middleware cunoscută sub numele de brocăr de mesaje.

Pentru a înțelege mai bine tipul de probleme care pot sa apară la integrarea unor sisteme software complexe în continuare se va prezenta un exemplu de integrare. Se consideră o companie care are patru sisteme software diferite în care se introduce date despre clienți. Fiecare din cele patru sisteme stochează date comune despre clienți dar și date unice care nu sunt stocate de celelalte sisteme. În plus fiecare sistem folosește un format de înregistrări diferit față de formatul folosit de către celelalte sisteme, numele câmpurilor fiind diferit de la un sistem la altul (ex.: un sistem folosește pentru adresa clientului un câmp numit ADRESĂ, în timp ce altul folosește pentru aceeași informație un câmp numit LOCAȚIE). Pentru actualizarea datelor clienților fiecare sistem pune la dispoziție o librărie API.

Compania care deține cele patru sistem software dorește să dezvolte o aplicație web care să permită clienților să își actualizeze datele online. Astfel, datele introduse în aplicația web trebuie să fie transmise către toate cele patru sisteme.

Comunicarea folosește cozi de mesaje pentru a comunica între sistemele software. Astfel, aplicația web pentru fiecare actualizare va trebui să creeze un mesaj care să fie înțeles de toate cele patru sisteme software și să trimită acest mesaj către cele patru sisteme prin intermediul unei cozi de mesaje. Fiecare din cele patru sisteme software a căror date trebuie actualizate vor interfața cu coada de mesaje printr-o componentă software dedicată. Rolul componentei care realizează interfațarea fiind acela de a citi mesajul din coadă si de-al formata corespunzător astfel încât el să poată fi salvat in baza de date utilizând librăria API corespunzătoare sistemului software pentru care se realizează interfațarea.

Soluția prezentată mai sus are următoarele implicații:

- Dacă formatul de mesaj comun (folosit de aplicația Web) se modifică, atunci toate cele patru componente care realizează transformarea mesajului trebuie modificate pentru a înțelege noul format de mesaje.
- Dacă se modifică librăria API a unuia din cele patru sisteme, atunci doar componenta corespunzătoare acelui sistem va trebui modificată.
- Modificarea unei transformări necesită coordonarea cu echipa de dezvoltare care se ocupă de sistemul software corespunzător transformării care se modifică.

S-a obținut astfel o arhitectură în care componentele sunt puternic interconectate. Această interconectare este determinată de nevoia de a se conveni asupra formatului de mesaj prin intermediul căruia se realizează comunicarea.

O altă soluție ar putea fi ca aplicația Web să trimită mesajele gata formatate către cele patru sisteme software. În acest fel sistemele software vor putea folosi direct mesajul transmis fără a mai fi necesară o formatare a lui. Dezavantajul principal al acestei soluții constă în faptul că aplicația Web va trebui să conțină toată logica de formatare a mesajelor, fiind astfel predispusă la modificări atunci când unul din sistemele cu care ea comunică schimbă formatul de mesaj.

O a treia soluție presupune folosirea unui brocăr de mesaj pentru a comunica între aplicația web și cele patru sisteme software ale căror date trebuie actualizate. Din punct de vedere arhitectural un brocăr este un model de proiectare care conține o componentă care decuplează clienții de servere mediind comunicare dintre ei. Similar, un brocăr de mesaje extinde capabilitățile unei cozi de mesaje astfel încât să permită încorporarea logicii sistemului software. În cazul exemplului considerat logica încorporată în brocăr se referă la convertirea mesajelor de la formatul utilizat de aplicația web la formatele recunoscute de cele patru sisteme software (Fig. 7.1).

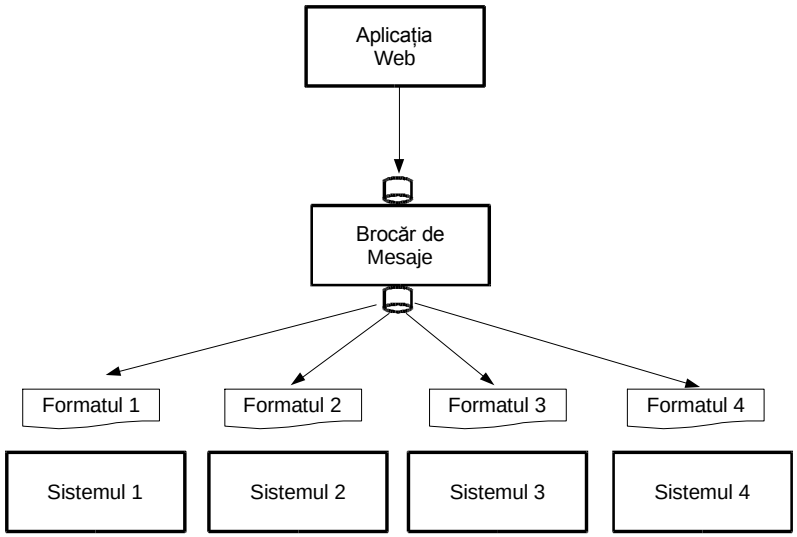

Fig. 7.1. Utilizarea unui brocăr de mesaje pentru a comunica între aplicația web și cele patru sisteme.

Utilizarea brocărului de mesaje asigură o decuplare totală a aplicației web de cele patru sisteme cărora trebuie să comunice schimbarea datelor unui client. Aplicația web creează și transmite un singur mesaj către brocăr, care transformă mesajul astfel încât să fie înțeles de fiecare din cele patru sisteme. Un alt avantaj al acestei soluții îl constituie faptul că toate componentele software sunt simplificate întrucât transformarea mesajelor se face într-un singur loc și anume pe brocărul de mesaje.

În prezent există o serie de implementări de brocări de mesaje, în mod normal o astfel de implementare conține următoarele facilități:

- Descrierea grafică a transformărilor complexe a mesajelor de la formatul de intrare la formatul de ieșire. Transformările pot fi simple (mutarea de informații între câmpurile mesajului de intrare și câmpurile mesajului de ieșire) sau complexe (conversie de date sau aplicarea unor funcții matematice).
- Dispune de un motor de transformare performant care permite transformarea simultană a mai multor mesaje.

- Descrierea unui flux de execuție în care un mesaj poate fi direcționat către o anumită transformare și către o anumită ieșire în funcție conținutul acestuia.

Avantajele folosirii brocărilor de mesaje sunt:

- Transformarea mesajelor este ușor de realizat și înțeles, atunci când se modifică un format de mesaj nu este necesară modificarea aplicațiilor care comunică prin intermediul brocărului.
- Transformările între mesaje sunt gestionate central, fiind suficientă o singură echipă de dezvoltatori pentru a realiza și testa eventualele modificări.
- Dispune de nu motor de transformare multi-fir foarte performant.

Brocării de mesaje recepționează un mesaj îl transformă conform regulilor de transformare și îl transmit către destinație. Ei funcționează cel mai bine atunci când aceste transformări se pot executa rapid (câteva milisecunde). Dacă un brocăr sau mașina pe care el se află se defectează atunci transformarea care se executa în timp ce a apărut defecțiunea se va executa de la început, ceea ce înseamnă că nu este necesară menținerea unei stări costisitoare și nici a unui context tranzacțional.

Ca orice altă tehnologie middleware și brocării de mesaje au anumite dezavantaje: sunt tehnologii dezvoltate de anumite companii software ceea ce inevitabil duce la o dependență de respectiva companie. Atunci când volumul de mesaje care trebuie procesat este foarte mare se poate ajunge la situația în care brocărul de mesaje introduce o gâtuire în sistem. Majoritatea tehnologiilor existente la ora actuală pe piață oferă soluții de clustering, dar evident aceste soluții duc la creșterea complexității și a costurilor.

7.2 Modelarea Proceselor Business

Procesul business într-o companie modernă poate fi complex în ceea ce privește numărul de aplicații (sisteme software) care trebuie accesate și actualizate pentru a finaliza un anumit serviciu. De exemplu, dacă se consideră procesul business al unui ordin de vânzare se poate să fie nevoie de realizarea următoarelor operații. Un client trimite un ordin prin intermediul unui apel telefonic. Odată ordinul transmis, cardul de credit al clientului este verificat utilizând un serviciu extern, iar baza de date este actualizată pentru a înregistra ordinul și pentru a trimite factura clientului. De asemenea este transmis un mesaj către departamentul care se ocupă cu livrarea pentru a se actualiza stocul și pentru a se trimite produsul comandat către client. Când clientul primește produsul, acesta va plăti, plată care de asemenea se va înregistra în baza de date. Periodic toate datele financiare sunt extrase din baza de date pentru a se genera rapoarte și pentru a fi arhivate într-o altă baza de date.

Implementarea unui astfel de proces prezintă două mari provocări. În primul rând, din momentul în care s-a trimis comanda și până când s-a realizat plata poate să dureze câteva zile, poate chiar săptămâni. De aceea este nevoie să se poată stoca starea procesului pentru o perioadă lungă de timp. Pierderea acestei informații este inacceptabilă. În al doilea rând, apariția unor excepții pot să necesite anularea comenzii. Anularea unei comenzii trebuind să șteargă informația din baza de date și se returneze eventuala suma plătită de client drept garanție.

Anularea ordinului s-ar putea realiza foarte ușor dacă acesta ar fi gestionat într-o singură tranzacție. În acest caz utilizarea unei tranzacții în sens clasic ar însemna blocarea bazei de date pentru o perioada de timp mult prea mare (zile). Soluția constă în folosirea tranzacțiilor de tipul “long-running”. O tranzacție de tipul “long-running” constă dintr-o serie de activități care nu blochează datele pe care le modifică în diferitele sisteme software. Modificările sunt realizate local pentru fiecare sistem software. Totuși, trebuie specificată o funcție compensator, care, dacă una din activitățile cuprinse în tranzacție eșuează,

va restaura orice modificare care a fost realizată de tranzacție, lăsând datele în aceeași stare în care se aflau înainte de începerea tranzacției.

Tranzacțiile de tipul "long-running", sunt dificil de implementat, în plus unele activități pot fi imposibil de compensat, de exemplu, cum se poate compensa pentru un e-mail trimis către client sau o factură trimisă prin poștă. Așadar, tehnologiile care permit compensare tranzacțiilor nu rezolvă toate problemele, dar ele oferă un mecanism prin care tranzacțiile de tipul "long-running" pot fi proiectate și executate.

Platformele care permit modelarea proceselor business sunt implementate de obicei peste brocări de mesaje. Acestea extind brocării de mesaje adăugând:

- Gestionarea stării – starea un proces business aflat în execuție este salvată într-o bază de date. Acest lucru face ca starea să nu se piardă în cazul în care apar unele defecțiuni. De asemenea odată ce starea a fost salvată în baza de date memoria folosită pentru stocarea ei poate fi eliberată.
- Utilitare de dezvoltare – permite definirea procesului business prin intermediul unor diagrame vizuale.
- Utilitare de instalare – permit programatorilor să mapeze pașii procesului business peste sistemele software, utilizând diferite tipuri de conexiuni, de exemplu: cozi de mesaje, protocoale web sau sisteme de fișiere.

Tehnologiile care permit modelarea proceselor business au apărut relativ recent. Necesitatea pentru aceste tehnologii a izvorât din dorința de a automatiza cât mai mult din procesul business al companiilor, care de cele mai multe ori necesită accesarea mai multor aplicații.

7.3 Integrarea Arhitecturilor Eterogene

Integrarea aplicațiilor eterogene este o problemă reală cu care se confruntă majoritatea companiilor mari. Deși sunt multe probleme care trebuie rezolvate atunci când se dorește integrarea unor aplicații eterogene, în esență problema este una de arhitectură și se referă la toleranța la modificări.

În cazul general dacă se dorește integrarea a **N** aplicații se ajunge la proiectarea a **(N^2-N)** interfețe de comunicare. Astfel pe măsură ce **N** crește numărul de interfețe care trebuie realizate crește polinomial, făcând ca o astfel de arhitectură să nu fie scalabilă în ceea ce privește modificările. O astfel de arhitectura este cunoscută sub denumirea de arhitectură *spaghetti*.

Soluția pentru o astfel de integrare este, așa cum s-a văzut în Secțiunea 7.1, utilizarea unui brocăr de mesaje. Complexitatea în ceea ce privește numărul de interfețe necesare se reduce foarte mult întrucât fiecare aplicație poate să trimită mesajele către brocăr în format propriu, brocărul având sarcina de al prelucra și al transforma în formatul destinație. Dacă se dorește modificarea formatului de mesaj utilizat de una din aplicații este suficient să se modifice doar modul în care brocărul prelucrează acel tip de mesaj, restul de aplicații nu vor observa această modificare.

În ciuda avantajelor rezultate în urma utilizării brocărului de mesaje există și o serie de dezavantaje:

- Arhitectura de tipul *spaghetti* nu a fost eliminată, ea fiind implementată în logica dezvoltată pe brocărul de mesaje, fiind necesare transformării pentru toate tipurile.
- Brocării pot să introducă limitări în ceea ce privește performanța întrucât toate mesajele comunicate între aplicații trebuie să treacă prin brocăr. Ca și o soluție la această problemă majoritatea tehnologiilor existente pe piață oferă suport pentru replicare și clustering pentru a

permite scalarea performanței. Totuși această scalare crește complexitatea sistemului și duce la creșterea costurilor.

Soluția la această problemă constă în definirea unui model de date (cunoscut sub denumirea de model de date canonic), care va reprezenta formatul în care vor fi transmise toate datele între diferitele aplicații ale unei companii. Utilizând un model de date canonic schimbul de mesaje se reduce la următorii pași:

- Aplicația sursă trebuie să transforme datele locale astfel încât să respecte formatul canonic de date.
- Aplicația sursă trimite mesajul către destinație.
- Aplicația destinație recepționează mesajul în formatul canonic și îl transformă în reprezentarea internă.

Acest lucru înseamnă că fiecare aplicație trebuie să știe:

- cum să transforme mesajele în format canonic în mesaje în format intern;
- cum să transforme mesajele pe care le trimite din formatul intern în formatul canonic.

Utilizarea mesajelor în format canonic reduce complexitatea, sunt necesare doar **2N** interfețe. În plus nu mai este nevoie de un brocăr de mesaje ceea ce înseamnă, că arhitectura este scalabilă, de asemenea toleranța la modificări este îmbunătățită.

Principalul impediment în calea unei astfel de arhitecturi constă în faptul că de cele mai multe ori definirea unui format de mesaj canonic cu care să fie de acord toată lumea dintr-o companie este de cele mai multe ori greu de realizat, de aceea se preferă utilizarea unui brocăr de mesaje.

Un alt impediment este reprezentat de interfațarea cu o aplicație a unei companii partenere asupra căreia nu există nici un control. În această situație definirea unui format canonic este și mai dificil de realizat.

Bibliografie

[1] Ian Gorton. Essential Software Architecture. Editura Springer. 2006.