Application Development Based on Service Oriented Architectures

Prep. Cristian IONIȚĂ
Catedra de Informatică Economică, A.S.E. Bucureşti

With the growth of the Internet, and of connected networks in general, the development and deployment of large scale distributed systems has become common practice in large enterprises. This paper presents a Service Oriented Architecture (SOA) that can be used to build such systems in an efficient manner.
***Keywords**: architecture, Service Oriented Architectures, message pipeline, enterprise applications.*

Principii şi concepte fundamentale

Orientarea pe servicii reprezintă o nouă metodă pentru proiectarea sistemelor distribuite. Conform [SPRT04], conceptele principale folosite în cadrul acestei abordări sunt:

- **serviciu**: unitate independentă a unei aplicații care expune funcționalitatea către clienții din rețea prin intermediul unei interfețe bazate pe mesaje
- **client**: un modul al unei aplicații care facilitează accesul utilizatorilor la funcționalitatea oferită de servicii
- **sistem distribuit**: un sistem interconectat de servicii şi clienți

Spre deosebire de sistemele distribuite orientate obiect, unde integrarea aplicațiilor se face prin intermediul activării la distanță a obiectelor, sistemele bazate pe servicii permit o integrare bazată pe schimbul de mesaje independente de modul de implementare al clientului şi al serviciului. Apelurile de metode din modelele distribuite orientate obiect sunt considerate în modelul orientat pe servicii o tehnică privată de implementare şi nu o construcție de bază. Faptul că o interacțiune poate fi implementată ca un apel de metodă este considerat ca un detaliu implementare care nu este vizibil în exterior.

Sistemele orientate obiect sunt strâns cuplate; modificarea unei părți poate fi realizată doar în condițiile în care sunt modificate şi celelalte subsisteme, iar modificările sunt operate simultan. O asemenea abordare s-a dovedit ineficientă, în special în situația în care diferitele subsisteme sunt realizate şi operate de către organizații diferite, cum este cazul aplicațiilor de tip B2B. O altă problemă a acestor sisteme este cerința ca diferitele părți ale sistemului trebuie să fie realizate folosind aceeaşi model de obiecte şi aceeaşi platformă de dezvoltare.

Sistemele orientate pe servicii sunt sisteme slab cuplate, proiectate pentru a permite schimbarea pentru adaptarea la noi cerințe sau metode de implementare. Prin utilizarea modelului orientat pe servicii este posibilă obținerea unei interoperabilități între aplicații, indiferent de platformele şi limbajele de programare utilizate în dezvoltarea acesteia. Dezvoltatorii şi integratorii de aplicații nu au nevoie să cunoască tipul de sistem, modelul de obiecte sau protocoalele folosite de către sistemele care trebuie integrate. Expunerea funcționalității folosind protocoale standardizate şi interfețe care pot fi obținute dinamic conduce la o cuplare slabă între subsisteme care permite o conectare simplă şi asigură o flexibilitate sporită aplicației.

Dezvoltarea sistemelor orientate pe servicii se bazează pe patru principii fundamentale ([MS03]).

Granițele între subsisteme sunt explicite: funcționalitatea disponibilă în exterior este definită explicit la granița aplicației. Spre deosebire de metodele bazate pe invocarea implicită a metodelor, se foloseşte un model care impune construirea şi trimiterea explicită a mesajelor.

Deoarece sistemele orientate pe servicii sunt în general distribuite geografic şi rulează în medii sau organizații multiple, costul traversării granițelor este foarte mare din punctul

de vedere al performanţelor şi al complexităţii. Arhitecturile orientate pe servicii iau în considerare acest lucru punând un acces deosebit pe identificarea şi definirea graniţelor dintre sisteme.

Serviciile sunt independente: serviciile care compun sistemul pot fi modificate şi puse în funcţiune independent faţă de restul sistemului. Această autonomie implică şi faptul că fiecare serviciu este complet responsabil de integritatea datelor si proceselor pe care le controlează.

Sistemele distribuite orientate obiect sunt testate şi puse în funcţiune ca un ansamblu unitar. Datorită acestui fapt, orice modificare presupune o cunoaştere detaliată şi posibilitatea exercitării controlului asupra tuturor componentelor sistemului, lucru dificil de îndeplinit în cazul sistemelor complexe sau a celor distribuite în cadrul mai multor organizaţii. Sistemele orientate pe servicii consideră ca unitate de bază pentru instalare şi punere în funcţiune serviciul, nu sistemul în ansamblu. Sistemul poate fi modificat relativ uşor, prin modificarea serviciilor existente sau adăugarea de noi servicii. Descrierea mesajelor care traversează graniţa serviciului oferă toate informaţiile necesare în cazul efectuării unor modificări. În aceste condiţii nu mai este necesară o cunoaştere sau un control global asupra sistemului. Exemple în acest sens sunt serviciile puse la dispoziţie de Amazon.com şi EBay. Acestea au fost create şi puse în funcţiune înainte de a exista aplicaţii client. Folosirea tehnicilor prezentate a permis actualizarea ulterioară a acestora fără a cunoaşte modul de implementare al clienţilor şi fără a întrerupe funcţionarea acestora.

Noţiunea autonomiei serviciilor are implicaţii şi asupra modului în care sunt tratate erorile. Sistemele bazate pe servicii consideră apelul cererilor în acelaşi context cu serviciul, cum este cazul sistemelor orientate obiect, ca fiind doar un caz particular, nu o regulă. Din acest motiv, serviciile consideră că aplicaţia client poate ceda în orice moment şi fără nici o notificare. Pentru a menţine integritatea sistemului se pot folosi o serie de metode pentru tratarea unor astfel de cazuri: tranzacţii, sisteme robuste de mesagerie sau algoritmi special adaptaţi. Datorită faptului că multe servicii sunt disponibile prin reţele publice, sistemele trebuie să ia în considerare atât posibilitatea existenţei unor mesaje invalide, cât şi posibilitatea apariţiei unor mesaje trimise în scopuri malitioase. Pentru protejarea sistemelor în astfel de cazuri şi pentru a permite scalare mai bună a sistemului, mesajele trimise trebuie să conţină în afară de datele efective şi informaţiile referitoare la contextul de execuţie şi dovezile necesare autorizării cererii.

Comunicarea între servicii este descrisă prin intermediul schemelor şi al contractelor şi nu prin clase: informaţiile necesare unui client pentru a folosi un serviciu sunt disponibile prin intermediul contractului care descrie structura mesajelor pentru cererile şi răspunsurile valide.

Sistemele orientate obiect folosesc ca abstracţie de bază clasele pentru a grupa într-o singură entitate structura şi comportamentul aferent. Această grupare oferă avantaje la construirea modulelor aplicaţiei dar conduce la apariţia unor probleme legate de portabilitate între platforme şi necesită mecanisme mai complexe pentru comunicare. Arhitecturile bazate pe servicii propun o separare a strictă între structură, descrisă prin schema mesajelor, şi comportament ul care este implementat privat în cadrul serviciului/clientului. Această separare permite o implementare mai uşoară a sistemelor prin renunţarea la cerinţele specifice sistemelor orientate obiect referitoare la compatibilitatea mediului de execuţie, contextul de securitate comun sau execuţia de cod transferat de la distanţă.

Compatibilitatea între servicii este bazată pe contracte. Pentru a permite obţinerea unei cuplări slabe între servicii, acestea trebuie săşi publice cerinţele şi capabilităţile prin intermediul unor expresii prelucrabile automat. Aceste expresii definesc două tipuri de aserţiuni: condiţiile în care poate fi solicitat serviciul, cerinţe legate de forma şi conţinutul mesajului, garanţiile de securitate solicitate, şi garanţiile oferite de acesta. Aserţiunile sunt standardizate pentru a se asigura interpretare uniformă, indiferent de serviciul asupra căru-

ia se aplică. Aceste aserţiuni oferă clienţilor o modalitate de a identifica serviciile compatibile şi permit serverului să efectueze o validare automată a cererilor înainte ca acestea să fie prelucrate efectiv.

Prezentarea generală a arhitecturii

Modelul arhitectural general propus este unul orientat pe servicii, utilizabil pentru dezvoltarea aplicaţiilor distribuite. La nivel global, structurarea se poate face în trei straturi: prezentare, aplicaţie şi acces la resurse.

Stratul de prezentare este compus din clienţii sistemului care facilitează accesul utilizatorilor la serviciile oferite de sistem. Aceste aplicaţii pot avea diferite forme, aplicaţii web, aplicaţii desktop, aplicaţii mobile, în funcţie de necesităţile clienţilor. Indiferent de tipul clientului, operaţiunile sunt efectuate unitar prin intermediul serviciilor sistemului.

Stratul de acces la resurse încapsulează resursele folosite de servicii pentru îndeplinirea operaţiilor. Aceste resurse pot fi surse de date, sisteme existente sau alte servicii.

Prelucrările care nu ţin de funcţionalitatea de bază a aplicaţiei (configurare, securitatea, instrumentarea, monitorizarea, auditul, validarea cererilor) au fost separate sub forma aspectelor. Implementarea acestora este realizată folosind tehnica de interceptare a mesajelor care circulă între module. Această abordare permite simplificarea codului aplicaţiei şi permite dezvoltarea separată a elementelor care ţin de cerinţele secundare ale aplicaţiei.

Nucleul sistemului este constituit din stratul aplicaţiei. Acesta are două roluri principale: implementarea componentelor care asigură execuţia operaţiilor folosind resursele disponibile şi expunerea funcţionalităţii aplicaţiei către clienţi sub forma unui set stabil şi consistent de interfeţe. Separarea între interfeţele stabile şi componentele care le implementează permit sistemului să evolueze fără a necesita modificări în cadrul clienţilor sau a celorlalte sisteme care folosesc serviciile oferite.

Obiectivele generare urmărite în proiectarea arhitecturii au fost:

- asigurarea independenţei interfeţelor stabile de componentele care le implementează şi de mecanismele de transport ale mesajelor
- să ofere un mecanism configurabil pentru rutarea cererilor primite către componentele care execută operaţiile şi a răspunsurilor primite de la acestea folosind mai multe canale de transport
- să permită inserarea configurabilă în lanţul de procesare al mesajelor a codului corespunzător aspectelor

În afară de obiectivele generale mai există şi o serie de cerinţe care trebuie îndeplinite de către serviciile individuale pentru a se asigura calitatea sistemului. Pentru ca serviciile să fie utile, durabile şi să ofere o performanţă satisfăcătoare în condiţiile unui număr mare de utilizatori trebuie să îndeplinească funcţii utile organizaţiei, să aibă o granularitate corespunzătoare, să fie atomice şi să nu menţină informaţii de stare.

Serviciile trebuie de asemenea să nu menţină informaţii de stare între apelurile clienţilor. Toate informaţiile necesare sunt incluse în cadrul mesajului; contextul circulă împreună cu cererea şi răspunsul. Această abordare permite o distribuire eficientă a cererilor pe mai multe noduri de prelucrare şi permite implementarea separată a serviciilor secundare prin intermediul aspectelor.

Modelul conceptual

Interacţiunea dintre aplicaţia client şi acţiunea executată de către sistem necesită rezolvarea mai multor probleme: specificarea modului de transfer al mesajelor, autorizarea cererilor, înregistrarea şi tratarea erorilor şi auditul acţiunilor întreprinse.

Arhitectura prezentată se interpune între client şi acţiunea executată, interceptând mesajele şi procesându-le înaintea trimiterii către serviciu.

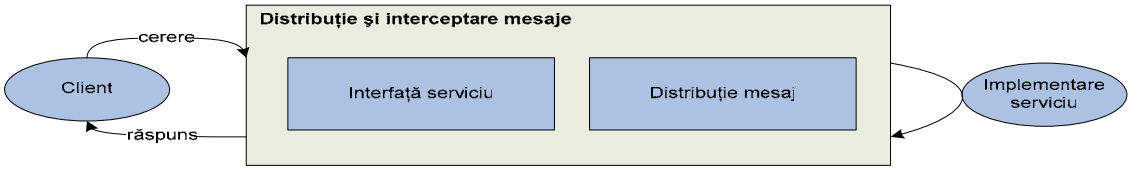

Fig.1. Modelul general

Mecanismul de interceptare conține două componente principale, prezentate în figura 1: interfața serviciului (granița dintre client şi sistem şi are rolul de a respinge cererile invalide sau neautorizate) şi distribuție mesaj (rutează mesajele şi invocă acțiunea cerută).
Structura detaliată a arhitecturii se poate observa în figura 2. Pentru îndeplinirea primului obiectiv enunțat, acțiunile sistemului trebuie să fie invocabile folosind mai multe modalități de transport. Canalele de transport permit primirea/trimiterea mesajelor folosind diverse protocoale de comunicație.
Mesajelor primite de către canalul de transport le este asociat un context de către adaptorul specific canalului de comunicație. Contextul este folosit pentru stocarea informațiilor asociate mesajului pe durata parcurgerii întregului lanț de procesare. Adaptorul are rolul de a inițializa lanțul de prelucrare (pipeline).

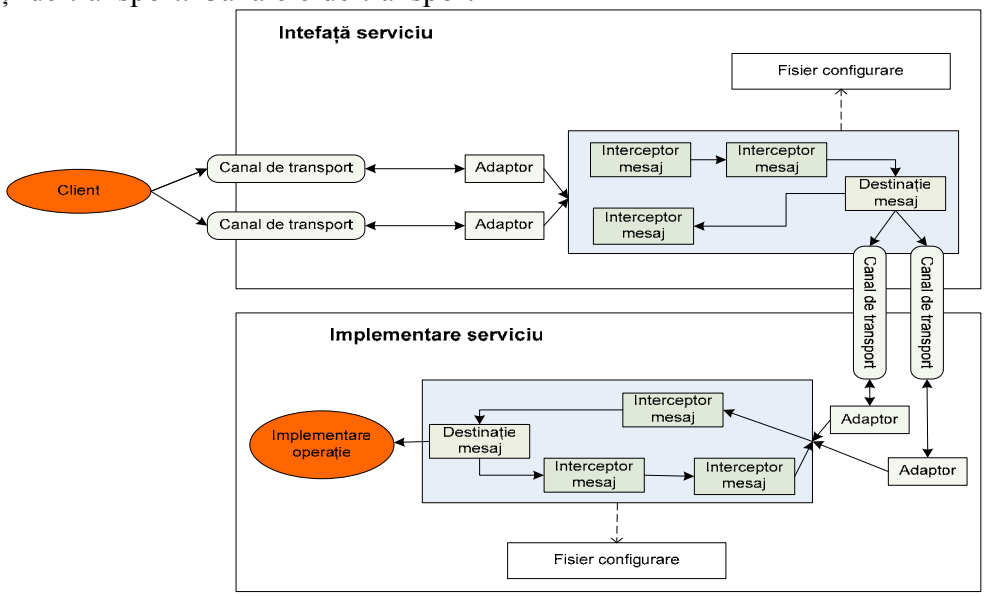

Fig.2. Detalierea modelului

Rolul unui lanț de prelucrare este de a apela interceptoarele definite în fişierul de configurare, de a trimite mesajul la destinație şi de a prelua răspunsul. În cadrul sistemului sunt folosite două lanțuri de prelucrare:

- interfață serviciu: este specific canalului de transport şi conține interceptoarele referitoare la autentificare şi autorizare; destinația mesajelor este cel de-al doilea lanț de prelucrare
- implementare serviciu: este specifică acțiunii şi conține interceptoarele legate de acțiunea întreprinsă (audit, monitorizare); destinația mesajelor este operația invocată

Implementarea operației extrage din context mesajul clientului, efectuează prelucrările necesare şi depune rezultatul în context.
După completarea execuției operației, contextul traversează în ordine inversă cele două lanțuri de prelucrare. Pe parcursul traversării sunt executate interceptoarele de retur specificate în fişierele de configurare.
Răspunsul este preluat de adaptor, translatat în formatul corespunzător şi trimis clientului prin intermediul canalului de comunicație.

Bibliografie

[INT04], Ioniță C., "Service Oriented Architectures for Enterprise Applications", Information Systems and Operations Management, Ed. Print Grup, Bucharest 2004
[MAC04], Macaulay A., "Enterprise Architecture Design and the Integrated Architecture Framework", Microsoft Architects Journal EMEA Edition, Vol. 1, Nr. 1, Ian. 2004
[MCK02], Mackenzie D., "Reliable Messaging with MSMQ and .NET", Microsoft Developer Network, February 2002
[MS03], "Microsoft .NET Solution Architectures", Microsoft Press, Redmond, 2003
[MS04], "Reference Solution for Service Development", Microsoft, Redmond 2004, în curs de publicare
[SPRT04], Sprott D., Wilkers L., "Understanding Service Oriented Architecture", Microsoft Architects Journal EMEA Edition, Volumul 1, Numarul 1, Ianuarie 2004
[VAS03], Vasters C., Swartz S., "Enterprise Architectures", Microsoft EMEA Architects Tour 2003