Proiectarea Sistemelor Software Complexe

Curs 8 –Arhitecturi Bazate pe Servicii

8.1 Context

Arhitecturile bazate pe servicii și serviciile Web reprezentă ultimul pas în dezvoltarea tehnologiilor middleware. Aceste tehnologii rezolva problema inter-operabilității și oferă baza pentru dezvoltarea de aplicații Internet de mari dimensiuni.

Tehnologiile middleware care permit integrarea aplicațiilor sunt folosite în diferite scopuri, de la interconectarea componentelor unei aplicații desktop sau Web, până la dezvoltarea de sistem software care se întind peste Internet. Tehnologiile tradiționale precum serverele de aplicații J2EE sau cele bazate pe mesaje reprezintă soluții ideale pentru dezvoltarea de sistem software care rulează în cadrul unei singure organizații. Totuși, ele sunt destul de limitate când se pune problema interconectării sistemelor software folosite de organizații diferite, care sunt conectate prin Internet. Serviciile Web și arhitecturile bazate pe servicii sunt proiectate tocmai pentru a satisface aceste nevoi.

În multe feluri tehnologiile bazate pe servicii și serviciile Web nu reprezintă o noutate. La fel ca restul tehnologiilor și arhitecturilor care oferă suport pentru calcul distribuit, principalul scop al tehnologiilor bazate pe servicii este acela de a oferi suport pentru un sistem software de a invoca funcționalității oferite de un alt sistem software (așa cum J2EE permite clienților Java să apeleze metode implementate de componente J2EE).

Principala diferență constă în faptul că accentul în cazul arhitecturilor bazate pe servicii se pune pe interoperabilitate și rezolvarea problemelor ridicate de folosirea de platforme și limbaje diferite. Deși se poate dezvolta o arhitectură bazată pe servicii utilizând orice tehnologie care permite calcul distribuit, numai serviciile Web oferă un grad de interoperabilitate nelimitat.

Necesitatea pentru interoperabilitate izvorăște din faptul că majoritatea companiilor mari dețin un mix de sisteme software în ceea ce privește limbajele de programare și platformele utilizate. În condițiile în care reimplementarea acestor sisteme pe o singură platformă este mult prea costisitoare și prea riscant este evidentă necesitatea pentru tehnologii middleware care să permită comunicare între aceste sisteme software. Nu de puține se întâmplă ca sistemele software care trebuie să comunice să fie dezvoltate pentru platforme incompatibile, de aceea este nevoie de interoperabilitate.

Serviciile Web și arhitecturile bazate pe servicii reprezintă o soluție care permite integrarea diferitelor tehnologii.

8.2 Sisteme bazate pe servicii

Nevoie de integrare a sistemelor de business a existat din totdeauna. Această integrare realizându-se prin intermediul documentelor fizice schimbate de diferite organizații: facturi, chitanțe sau ordine. În

prezent însă datorită Internetului și a serviciilor Web se tinde tot mai mult înspre înlocuirea documentelor fizice cu cele electronice.

Principiile care stau la baza arhitecturilor bazate pe servicii nu sunt noi. Unele principii, ca de exemplu, reducerea întârzierilor introduse de comunicarea pe rețea transmițând cât mai multă informație într-un singur apel, sunt deja folosite în sistemele software distribuite care oferă performanțe ridicate.

Principiile de bază ale arhitecturilor bazate pe servicii sunt:

- granițele sunt explicite;
- serviciile sunt autonome;
- se partajează scheme și contracte, și nu implementări;
- compatibilitatea este bazată pe reguli (policy).

În continuare vor fi detaliate fiecare din cele patru principii de baza ale unei arhitecturi bazate pe servicii.

8.2.1 Granițe Explicite

Serviciile sunt aplicații de sine stătătoare și nu doar cod care poate fi apelat de o aplicație client. Accesarea unui serviciu necesită cel puțin traversarea granițelor care separă procesele, și cel mai probabil traversarea rețelei și utilizarea autentificării inter-domenii. Fiecare graniță care trebuie traversată (proces, mașină, securitate) reduce performanța, crește complexitatea și crește probabilitatea de defectare. Foarte important este faptul că ele trebuie recunoscute și tratate în faza de proiectare.

Programatorii și furnizorii de servicii pot de asemenea să fie separați din punct de vedere geografic, ceea ce adaugă o nou graniță, care se reflectă în creșterea costului de dezvoltare și reducerea robusteții. Răspunsul la aceste provocări constă în păstrarea simplității atât în ceea ce privește definirea serviciului cât și în ceea ce privește suportul pentru standardele în domeniul serviciilor Web. Serviciile bune au o interfață simplă și partajează cât mai puține abstractizări și constrângeri posibil. Simplitatea face ca un serviciu să fie ușor de folosit și înțeles de către programatorii care vor dezvolta clienți pentru respectivul serviciu.

8.2.2 Serviciile sunt Autonome

Serviciile sunt aplicații independente și autonome și nu clase sau componente strâns legate de o anumită aplicație. Serviciile sunt proiectate pentru a fi instalate pe o rețea, posibil Internet, acolo unde ele pot fi cu ușurință integrate într-o aplicație în care este nevoie de ele. Serviciile nu trebuie să cunoască nimic despre clienți și trebuie să accepte cereri venite de oriunde, atâta vreme cât mesajele recepționate respectă formatul recunoscut de serviciu, iar cerințele în ceea ce privește securitatea sunt respectate.

Serviciile pot fi instalate și gestionate independent de alte servicii și de posibilele aplicații client, iar proprietarii serviciilor por modifica interfața și funcționalitatea unui serviciu în orice moment. Compatibilitatea cu versiunile anterioare este o problemă importantă pentru orice sistem de calcul distribuit, cu atât mai mult în cazul serviciilor Web care sunt deschise prin definiție.

Soluția la această problemă este rezolvată parțial de simplitatea și extensibilitatea unui serviciu. Tot ceea ce știu clienții despre un serviciu este cel fel de mesaje acceptă, respectivă returnează, respectivul serviciu. Astfel, un serviciu poate fi modificat atâta vreme cât mesajele anterioare sunt în continuare

acceptate ca fiind mesaje valide. Pot fi extinse chiar și mesajele care reprezintă cereri, respectiv răspunsuri, atâta vreme cât este menținută compatibilitatea cu mesajele anterioare.

Întrucât serviciile sunt aplicații autonome, ele trebuie să își asigure singure securitatea, trebuie să se protejeze împotriva apelurilor rău intenționate. Sistemele software care sunt instalate într-o rețea închisă pot să ignore în bună măsură problemele de securitate, este suficient să se bazeze pe firewall sau pe protocoale de comunicare securizate cum este protocolul SSL. În timp ce serviciile accesibile în Internet trebuie să ia măsuri de securitate mult mai stricte.

8.2.3 Sunt Partajate Scheme și Contracte, și nu implementări

Construirea de sistem software complexe robuste și fiabile este dificil de realizat. Construirea de astfel de sistem software din componente dezvoltate în limbaje de programare diferite este și mai dificilă. Totuși serviciile reușesc să ofere o soluție viabilă datorită simplității lor. Serviciile nu sunt obiecte distribuite care folosesc moștenire, nu suportă evenimente, proprietăți sau apeluri care folosesc o sesiune. Serviciile sunt doar aplicații care primesc și trimit mesaje. Clienții și serviciile nu partajează altceva decât definiția mesajelor și cu siguranță nu partajează cod.

Tot ceea ce o aplicație client trebuie să știe despre un serviciu este contractul și anume: structura mesajelor acceptate și returnate ca răspuns și ordinea în care mesajele trebuie trimise. Clienții pot folosi aceste contracte pentru a compune mesaje și a trimite cereri către un serviciu, în timp ce serviciul poate folosi contractul pentru a valida faptul că mesajul recepționat este în formatul corect.

8.2.4 Compatibilitatea este bazată pe reguli (policy)

Clienții trebuie să fie în totalitate compatibili cu serviciul pe care vor să îl folosească. Compatibilitatea în acest context nu înseamnă doar că, clienții trebui să înțeleagă formatul mesajelor și ordinea în care acestea trebuie transmise, ci și că ei trebuie să fie compatibili cu alte cerințe importante, cum ar fie faptul că informația trebuie criptată sau că trebuie verificat faptul că nu s-a pierdut informație în timpul transmisiei. În ceea ce privește serviciile, cerințele non-funcționale sunt definite prin intermediul regulilor, care nu sunt scrise ca și parte a documentației unui serviciu.

Regulile sunt un set de declarații care pot fi interpretate de o aplicație și care oferă informații despre cerințe precum securitatea și fiabilitatea. Regulile pot fi înglobate în contractul unui serviciu sau pot fi stocate într-o baza de date specializată, de unde pot fi obținute dinamic în timpul rulării.

Contractele bazate pe reguli pot fi privite ca și o parte a documentației serviciului, dar în același timp ele pot fi folosite de utilitare pentru a genera cod compatibil atât pentru partea de client cât și pentru cea de serviciu. De exemplu, o regulă de securitate poate fi folosită pentru a genera cod care va verifica, că mesajele primite sunt criptate, va decripta mesajele și le va trimite către aplicația serviciu.

Separarea regulilor de contracte permite aplicațiilor client să se adapteze dinamic pentru a respecta cerințele unui anumit serviciu. Acest lucru se va dovedi foarte util pe măsură ce serviciile sunt standardizate și sunt oferite de diferiți furnizori.

8.3 Servicii Web

Serviciile Web reprezintă o tehnologie care a fost proiectată explicit pentru a îndeplinii cerințele specifice arhitecturilor bazate pe servicii. În general serviciile Web nu diferă prea mult de alte tehnologii

middleware, dar ele se disting prin simplitate și interoperabilitate ridicată. Cea mai importantă caracteristică a serviciilor Web este faptul că ele sunt suportate de toți mari producători de software.

Toate aplicațiile de tipul tehnologiilor de integrare, inclusiv serviciile Web și alternativele la acestea precum J2EE sau CORBA, oferă de fapt doar patru funcții de bază care permit următoarele:

- descoperirea serviciului potrivit (utilizând fie UDDI fie un alt serviciu de localizare);
- descoperirea interfeței unui serviciu (utilizând WSDL);
- trimiterea unei cereri către un serviciu (utilizând SOAP) ;
- utilizarea serviciilor precum securitatea (utilizând standardul WS-*).

SOAP, WSDL și UDDI au fost primele standarde publicate, dar ele nu îndeplinesc decât cerințe de bază. Nu oferă suport pentru securitate, tranzacții, fiabilitate și alte funcții importante. Acest vid a fost însă umplut prin definirea unor standarde intitulate "WS-*", primele astfel de standarde au fost definite de către IBM și Microsoft.

Serviciile Web sunt standarde bazate pe XML. Serviciile sunt definite utilizând XML, aplicațiile cer servicii trimițând mesaje XML, practic serviciile Web ca standard folosesc o serie de standarde XML existente. În ansamblu serviciile sunt cât de simple se poate, având în vedere faptul că ele trebuie să ofere suport pentru securitate, robustețe și interoperabilitate. De asemenea există o serie de utilitare și librării, care oferă suport pentru aceste standarde, astfel că programatorii trebuie să înțeleagă doar capabilitățile acestor standarde și nu detalii de sintaxa XML. În Fig. 8.1 sunt prezentate principalele tipuri de standarde definite pentru serviciile Web.

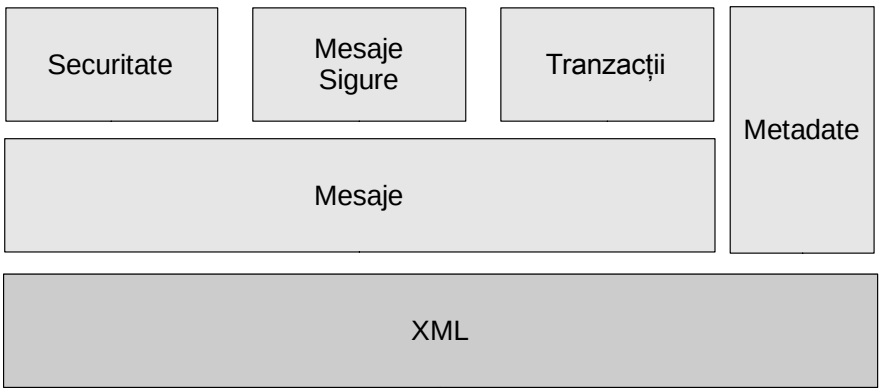

Fig. 8.1. Standarde definite pentru serviciile Web.

Unul dintre principiile care stă la baza serviciilor Web este acela ca diferitele câmpuri și atribute care sunt folosite pentru a suporta funcționalități precum securitatea sau fiabilitatea sunt total independente unul de celălalt. Astfel aplicațiile trebuie să includă doar acele atribute care sunt importante pentru funcționalitatea dorită, celelalte putând fi ignorate.

Un alt obiectiv al serviciilor Web este acela de a oferi suport pentru arhitecturi de sisteme care folosesc "intermediari". Astfel, în loc să se presupună că un client trimite întotdeauna un mesaj direct către un serviciu, modelul cu intermediari presupune că aceste mesaje pot să traverseze un lanț de aplicații până să ajungă la destinație. Aceste aplicații intermediare pot să facă o serie de operații cu mesajele primite, de exemplu, pot să logheze informația, să verifice securitatea sau chiar să modifice mesajele. Standardele existente pentru serviciile Web oferă suport pentru arhitecturi bazate pe intermediari în diverse feluri.

8.4 SOAP (Simple Object Access Protocol)

SOAP a fost primul standard definit pentru serviciile Web, el fiind și în prezent cel mai important standard și totodată cel mai răspândit. Acest standard specifică un protocol de comunicare bazat pe XML simplu și extensibil. Este echivalentul Java RMI, dar mult mai simplu și de departe mult mai ușor de implementat. SOAP este un standard suficient de simplu astfel încât el poate fi cu ușurință implementat de programatori.

Această simplitate rezultă din faptul că sunt evitate probleme complexe cum ar fi, un Colector de Memorie distribuit sau trimiterea obiectelor prin referință. Tot ceea ce standardul SOAP face, este să definească un protocol de comunicare bazat pe mesaje simplu și extensibil, care să permită invocarea serviciilor aflate la distanță utilizând protocoale precum HTML, SMTP, UDP sau orice alt protocol.

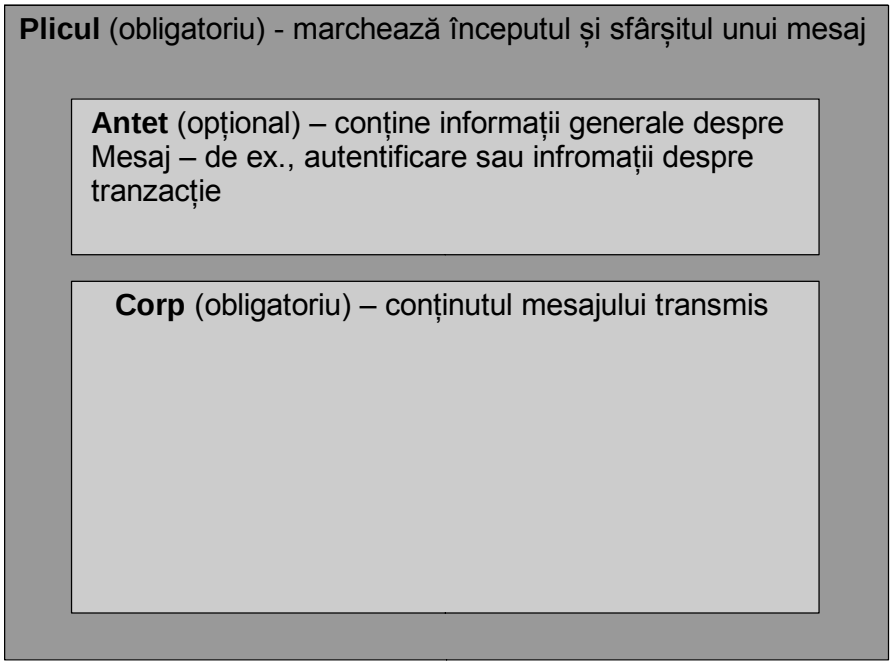

Fig. 8.2. Structura unui mesaj SOAP.

În Fig. 8.2 este prezentată structura unui mesaj SOAP. Antetul mesajului conține informație despre încărcarea mesajului, poate să includă informații de securitate sau informații referitoare la o tranzacție. Corpul mesajului reprezintă de fapt conținutul mesajului. Standardul SOAP nu impune ce informație să fie inclusă în antet, astfel că el poate fi extins de noi standarde, ca de exemplu, WS-Security, care pot să definească noi elemente pentru antet fără să fie nevoie să modifice standardul SOAP.

Există o serie de standarde care intră în categoria de standarde Mesaje, de exemplu, WS-Addressing și WS-Eventing. WS-Addressing oferă un mecanism de adresare independent de canalul de transport. WS-Eventing oferă suport pentru modelul publică-subscrie, el definește formatul de mesaj care reprezintă o cerere de subscriere și pe care clienții îl pot trimite pentru a subscrie la un anumit topic. Mesajele publicate care respectă un anumit filtru sunt trimise către clienții subscriși utilizând mesaje SOAP simple.

8.5 UDDI, WSDL și Metadate

Serviciile bazate pe standardul SOAP sunt descrise de documente WSDL (Web Service Description Language) și pot fi localizate căutând într-un director UDDI (Universal Description, Discovery and Integration). Serviciile pot să specifice cerințe precum securitatea și fiabilitatea prin așa numitele declarații *policy*, definite prin intermediul platformei WS-Policy și prin declarații *policy* specializate, cum ar fi de exemplu WS-SecurityPolicy. Aceste declarații pot fi atașate unei definiții de serviciu WSDL sau pot fi stocate într-o baza de date separată de unde pot fi accesate utilizând WS-MetadataExchange.

UDDI-ul este cel mai puțin folosit dintre cele trei standarde. Majoritatea serviciilor existente în momentul de față nu folosesc UDDI pentru a permite descoperirea, ci folosesc alte metode de localizare, de exemplu liste de servicii publicate pe site-uri Web. În viitor însă acest lucru s-ar putea să se schimbe.

WSDL-ul este folosit pentru a descrie serviciile Web, și anume interfața, metodele și parametrii. WSDL-ul este suportat de medii de dezvoltare precum Visual Studio sau WebSphere. Aceste utilitare pot genera o descriere WSDL direct din codul sursă, respectiv pot genereze cod care apelează serviciul pe baza unei astfel de descrieri.

8.6 Securitate, Tranzacții și Garantare

Având în vedere faptul că principalul protocol folosit ca și nivel transport în cazul serviciilor Web este protocolul HTTP, este evidentă nevoia de a securiza un serviciu Web. Problema securității unui serviciu Web este abordată de standardul WS-Security și standardele asociate acestuia. Acest standard specifică mecanisme criptografice puternice care să permită autentificarea celui care a inițiat un apel, protejarea conținutului și care să asigure integritatea informației transmise. Aceste standarde sunt proiectate astfel încât să permită adaptarea lor la noi tehnologii de securitate.

Standardul WS-Security oferă suport pentru arhitecturi de sisteme software bazate pe intermediari permițând folosirea mai multor elemente de securitate în antet, fiecare element fiind etichetat cu rolul lui și destinatarul căruia îi este adresat din lanțul de intermediari. Totodată standardul WS-Security suportă și folosirea criptării parțiale și a semnăturii parțiale.

În ceea ce privește suportul pentru tranzacții au fost definite două standarde. Standardul WS-AtomicTransactions oferă suport pentru tranzacții clasice distribuite de tipul ACID. AL doilea standard este WS-BusinessActivity, care oferă suport pentru atomicitate prin invocarea compensatorilor atunci când o tranzacție distribuită trebuie anulată.

Suportul pentru garantarea mesajelor transmise către și dinspre un serviciu Web asigură faptul că mesajele transmise între client și serviciu ajung la destinație fără erori și în ordinea în care au fost transmise. WS-ReliableMessaging nu garantează recepționarea mesajului în cazul în care serviciul eșuează, dar definește mecanisme care asigură faptul că mesajele vor ajunge la destinație în aceeași ordine în care ele au fost transmise chiar și atunci când comunicarea se face peste un nivel transport nesigur cum ar fi nivelul UDP.

Bibliografie

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