Technológie pre webové služby

Diplomová práca

Ľuboš Bisták **2006**

Technológie pre webové služby

DIPLOMOVÁ PRÁCA

Ľuboš Bisták

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KATEDRA INFORMATIKY

Informatika

Školiteľ záverečnej práce

Mgr. Pavol Mederly

BRATISLAVA 2006

Zadanie diplomovej práce

Cieľom práce je preskúmať, analyzovať a popísať súčasný stav v oblasti webových služieb so zameraním sa na ich použitie v podnikových informačných systémoch. Ide konkrétne o oblasť bezpečnosti, podpory transakcií a čiastočne aj o technológiu Web Services Invocation Framework a prostriedky na sprístupnenie webových služieb klientom. Úlohou diplomanta je analyzovať relevantné štandardy (napríklad WS-Security, WS-Transaction a ďalšie) a preskúmať súčasný stav ich implementácie. Súčasťou práce má byť zároveň vytvorenie ukážkovej aplikácie demonštrujúcej použitie webových služieb v kontexte podnikového informačného systému.

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne, s použitím literatúry a zdrojov uvedených v závere práce.

Bratislava, máj 2006

.....

Ďakujem svojmu diplomovému vedúcemu Mgr. Pavlovi Mederlymu za odborné vedenie práce, cenné rady a pripomienky.

Ďakujem môjmu kamarátovi a spolužiakovi Danielovi Chromekovi za podnetné postrehy z oblasti bezpečnosti.

Abstrakt

Diplomová práca sa zaoberá analýzou súčasného stavu a možnosťami využitia niektorých technológií súvisiacich s webovými službami v prostredí podnikových informačných systémov. Práca vychádza zo štúdia odbornej literatúry, štúdia štandardov a z analýzy a testovania množstva produktov.

Práca analyzuje a popisuje štandardy a ich implementácie v oblasti bezpečnosti webových služieb (XML podpis, XML šifrovanie, SAML, WS-Security, XACML) a v oblasti manažovania transakcií (WS-Transaction, WS-CAF). Práca sa tiež venuje popisu a možnostiam využitia technológie Web Services Invocation Framework a analýze možných prístupov k webovým službám. V rámci práce sa podarilo vytvoriť aplikáciu VIKUK-WS, ktorá pozostáva z webovej služby poskytujúcej informácie o publikačnej činnosti Univerzity Komenského a z ukážkového klienta vytvoreného v prostredí MS Excel.

Hlavným prínosom tejto práce je sumarizácia a porovnanie možností existujúcich technológií v oblasti bezpečnosti a transakcií webových služieb, vzhľadom na to, že podľa našich informácií v súčasnosti takýto prehľad neexistuje. Ďalším prínosom je vytvorenie aplikácie ilustrujúcej sprístupnenie údajov informačného systému prostredníctvom webových služieb.

Kľúčové slová: webové služby, bezpečnosť, transakcie, Web Services Invocation Framework, klienti webových služieb

Obsah

Zoznam obrázkov a tabuliek

Zoznam obrázkov

Zoznam tabuliek

Úvod

Webové služby predstavujú v dnešnej dobe často skloňovaný pojem v oblasti informačných technológií. Môžeme ich vnímať ako ďalší prvok vhodný pri budovaní distribuovaných systémov. Webové služby sú prostriedkami a nástrojmi na sprístupnenie funkčnosti a dát aplikácií externým aplikáciám v podobe služieb, ktoré sú vystavené na webe. Veľkým pozitívom je ich nezávislosť od jazyka, technológie a platformy, ktoré sa použili na implementáciu samotnej aplikácie. Aj z tohto dôvodu sa v súčasnosti veľmi často a intenzívne využívajú v prostredí podnikových informačných systémov na realizáciu komunikácie medzi ich jednotlivými aplikáciami, ale aj ako spôsob a prostriedok integrácie samostatných aplikácií do spoločného informačného systému.

Samotné webové služby sú silným prostriedkom informačného systému na realizáciu komunikácie medzi jeho jednotlivými časťami. Napriek tomu však štandardne nerieši všetky požiadavky kladené na informačný systém. Najdôležitejšími požiadavkami v dnešnej dobe sú bezpečnosť a podpora transakcií. Bezpečnosť je množina postupov a pravidiel určených na ochranu citlivých údajov napríklad s využitím šifrovania, digitálneho podpisu, autentifikácie a autorizácie. Pod transakciami budeme rozumieť množinu operácií, ktoré sa majú vykonať atomicky (všetky operácie sa vykonajú úspešne alebo žiadna) na dosiahnutie spoločného výsledku.

Z tohto dôvodu vznikajú podporné technológie, ktoré predstavujú určitú nadstavbu resp. rozšírenie webových služieb, ktorých cieľom je zabezpečiť protokol SOAP, vytvoriť podporu transakcií vo webových službách, pridať webovým službám určité vlastnosti, ktoré im umožnia lepšie využitie pri integrácii aplikácií a podobne.

Dnes existuje veľké množstvo technológií, ktoré sa snažia riešiť rôzne potreby webových služieb. Problém je v tom, že mnohé technológie sa prekrývajú v svojej aplikačnej doméne, konkurujú si navzájom a špecifikácie k niektorým z nich nie sú v konečnej verzii. Ďalší problém predstavujú samotné implementácie týchto technológií. Pre niektoré podporné technológie ešte neexistuje implementácia, pre iné technológie je ich viac, niektoré implementácie sú komerčné, niektoré voľne použiteľné, niektoré sú už stabilné, niektoré majú ešte množstvo nedostatkov resp. chýb a sú v štádiu vývoja. Niektoré z nich majú dodatočné požiadavky, napr. vyžadujú špeciálnu implementáciu webových služieb resp. určitú verziu tejto implementácie.

Cieľom tejto práce je prieskum, analýza a popísanie súčasného stavu v oblasti webových služieb so zameraním sa na ich použitie v podnikových informačných systémoch. Ide konkrétne o oblasť bezpečnosti, podpory transakcií a čiastočne aj o technológiu Web Services Invocation Framework a prostriedky na sprístupnenie webových služieb klientom. Úlohou je analýza relevantných štandardov (napr. WS-Security, WS-Transaction a ďalšie) a prieskum súčasného stavu ich implementácie. Ďalším cieľom je vytvorenie ukážkovej aplikácie demonštrujúcej využitie niektorých preskúmaných technológií v kontexte podnikového informačného systému.

Prvá kapitola tejto práce obsahuje úvod do problematiky webových služieb a stručne popisuje základné prvky webovej služby.

Druhá a tretia kapitola sa venujú analýze súčasného stavu bezpečnosti a podpory transakcií vo webových služieb. Popisujú existujúce technológie, ich vzájomný vzťah a k nim prislúchajúce implementácie. Snažia sa tiež načrtnúť spôsob využitia týchto technológií v prostredí podnikových informačných systémov.

Štvrtá kapitola popisuje technológiu Web Services Invocation Framework a analyzuje možnosti jej využitia v kontexte podnikových informačných systémov.

Piata kapitola sa venuje možnosti klientského prístupu k funkčnosti a údajom vystaveným pomocou webových služieb. V rámci tejto práce bola vytvorená ukážková aplikácia VIKUK-WS, ktorá demonštruje klientský prístup k webovým službám a je detailne popísaná v šiestej kapitole.

Práca je doplnená o slovník pojmov za účelom zvýšenia jej čitateľnosti. Tento slovník obsahuje krátke vysvetlenie technických pojmov, ktoré sú potrebné na pochopenie jej jednotlivých častí.

1 Úvod do webových služieb

Webové služby sú založené na jazyku XML, ktorý ich robí univerzálnymi, rozšíriteľnými a napomáha prekonávať obmedzenia heterogénnych aplikácií (napr. rôzne programovacie jazyky, v ktorých boli tieto aplikácie vyvinuté). Základom webových služieb je protokol SOAP a dokument WSDL.

Skratka SOAP v minulosti znamenala „jednoduchý protokol na prístup k objektom“ (Simple Object Access Protocol) a dnes predstavuje skôr zaužívaný pojem v prostredí informačných technológií. Ďalšou možnou interpretáciou tejto skratky je výraz „protokol pre SOA[1]“ (SOA Protocol). SOAP presne definuje protokol komunikácie s webovou službou. S webovou službou je možné komunikovať prostredníctvom XML správ. Tieto správy predstavujú volanie webovej služby za účelom vykonania určitej operácie resp. žiadosť o určité dáta. Taktiež reprezentujú výsledok vykonanej operácie resp. získané dáta. Správy SOAP môžu byť prenášané viacerými transportnými protokolmi ako HTTP (resp. HTTPS) a SMTP. Tieto správy môžu byť prenášané systémami, založenými na prenose správ. Príkladom takéhoto riešenia je prenos správ SOAP prostredníctvom systému JMS. V dnešnej dobe je najrozšírenejší prenos cez protokol HTTP (resp. HTTPS).

Dokument WSDL slúži na presný popis webovej služby. WSDL je skratkou pojmu Web Service Definition Language. Definuje formát správ protokolu SOAP (vrátane vstupných a výstupných údajov), transportný protokol pre správy SOAP, presné umiestnenie služby na webe a podobne. Je možné použiť centrálny register (resp. databázu) na uloženie dokumentov WSDL, a tak uľahčiť ich následné vyhľadávanie a prístup k nim. Jeden z takýchto registrov špecifikuje technológia UDDI (Universal Description, Discovery, and Integration).

Ďalšie informácie o webových službách je možné nájsť v [1].

[1] Service Oriented Architecture (SOA) – architektonický vzor, ktorého základným prvkom sú služby (nielen webové). Systém postavený na tomto vzore sa skladá zo služieb, ktoré spolu vytvárajú jeden celok. Každá zo služieb vystavuje určitú funkčnosť aplikácie, ktorú môžu využiť iné externé aplikácie v rámci tohto systému.

2 Bezpečnosť webových služieb

Bezpečnosť predstavuje jednu zo základných požiadaviek kladených na podnikový informačný systém. Základnou ideou bezpečnosti informačného systému je ochrana informácií a funkčnosti, ktoré tento systém poskytuje alebo s ktorými operujú jednotlivé jeho časti. Pri vytváraní tejto kapitoly sme čerpali informácie a inšpiráciu z nasledovných zdrojov: [2], [3], [4], [5], [6], [7].

2.1 Základné princípy bezpečnosti

Vo svete bezpečnosti sa používajú pojmy entita (*entity*) a identita (*indetity*). Entita predstavuje človeka, aplikáciu, webovú službu, počítač a podobne. Identitu si môžeme predstaviť ako entitu v určitej bezpečnostnej doméne. Príkladom môže byť študent univerzity, ktorý popri svojom štúdiu pracuje v istej komerčnej spoločnosti. Ide o tú istú entitu (osobu resp. človeka), ktorá predstavuje dve rozdielne identity (študent a zamestnanec) v dvoch rozdielnych doménach. To znamená, že v práci je identifikovaný podľa jemu prideleného čísla zamestnanca a adresy elektronickej pošty a na univerzite je identifikovaný podľa jemu prideleného čísla študenta a adresy elektronickej pošty fakulty, ktorá je súčasťou univerzity. Bezpečnostnú doménu si môžeme predstaviť ako organizáciu, ktorá používa stanovenú množinu pravidiel, ktoré definujú identitu a podľa ktorých je táto identita vytvorená. Tieto pravidlá určujú, čo tieto identity majú povolené robiť resp. k akým informáciám majú prístup v rámci danej organizácie. Príkladom môže byť informačný systém fakulty, kde je administrátorovi tohto systému povolené vytvárať kontá pracovníkom fakulty a pracovníci môžu iba vykonávať operácie súvisiace s ich kontom.

Ak identita bola vytvorená a overená v jednej bezpečnostnej doméne a požaduje sa, aby bola uznaná ako identita s príslušnými právami, v druhej bezpečnostnej doméne hovoríme o portovaní resp. preklade tejto identity. Jedným z riešení tohto problému je vytvorenie vzťahu dôvery medzi týmito dvoma samostatnými bezpečnostnými doménami.

Jedným zo základných prvkov bezpečnosti webových služieb je utajenie a zaistenie integrity prenášaných informácií. Utajenie informácie znamená, že pri komunikácii s webovou službou sa k tejto informácii nedostala žiadna neoprávnená entita. Informácia spĺňa vlastnosť integrity, ak počas prenosu nebola zmenená žiadnou entitou, ktorá na jej zmenu nemá právo. Na zabezpečenie informácií existuje vo všeobecnosti 5 základných bezpečnostných mechanizmov:

1. **Šifrovanie** (*encryption*) – transformuje informáciu resp. komunikáciu do podoby, ktorá môže byť prečítaná iba entitami pre ktoré je určená. Hlavnou úlohou šifrovania je zabezpečiť utajenie prenášanej informácie. Dáta sa šifrujú pomocou šifrovacích algoritmov. Existujú dva typy šifrovania:

    a) symetrické – na šifrovanie a dešifrovanie sa použije ten istý šifrovací kľúč,

    b) asymetrické – existuje súkromný a verejný kľúč. Príkladom môže byť komunikácia, v ktorej odosielateľ zašifruje posielanú informáciu

verejným kľúčom prijímajúcej strany. Prijímajúca strana použije svoj súkromný kľúč na dešifrovanie prijatej, zašifrovanej informácie. Odosielateľ získal verejný kľúč prijímajúcej strany z certifikátu, ktorý získal od certifikačnej autority, ktorej dôveruje. Certifikát je podpísaný súkromným kľúčom certifikačnej autority (CA1), ktorý sa dá overiť pomocou verejného kľúča certifikačnej autority. Verejný kľúč tejto certifikačnej autority (CA1) odosielateľ získa z certifikátu jej nadradenej certifikačnej autority (CA2). Na overenie certifikátu autority CA2 odosielateľ použije certifikát certifikačnej autority CA3, ktorá je nadradená autorite CA2. Takto môže odosielateľ postupovať v hierarchii certifikačných autorít, až kým sa dostane k certifikačnej autorite CAx, ktorej certifikátu už dôveruje. Tento certifikát môže byť zabudovaný v operačnom systéme odosielateľa. Príkladom je operačný systém Microsoft Windows, ktorý obsahuje certifikát spoločnosti VeriSign.

Symetrické šifrovanie je rýchlejšie v porovnaní s asymetrickým šifrovaním. Väčšinou sa asymetrické šifrovanie využíva na dohodnutie kľúča symetrickej šifry a pri nadviazaní komunikácie medzi komunikujúcimi stranami. Na zníženie možnosti prelomenia zabezpečenej komunikácie sa odporúča pri každej komunikácii využívať iný symetrický kľúč.

2. **Digitálny podpis** (*digital signature*) – jeho hlavnou úlohou je zabezpečiť integritu komunikácie. Digitálny podpis predstavuje podpísaný heš[2] (*hash*) posielanej správy. Prijímajúca strana si vypočíta heš prijatej správy. Rovnosť prijatého a vypočítaného hešu by mala zaručovať integritu prijatej správy. Digitálny podpis sa dá využiť ako dôkaz, že odosielateľ skutočne odoslal danú správu, lebo iba on je držiteľom tohto súkromného kľúča (za predpokladu, že sa k nemu nedostala žiadna iná entita). Cieľom je, aby odosielateľ neskôr nemohol poprieť poslanie danej správy. Táto vlastnosť sa nazýva nepopierateľnosť (*non-repudiation)*.

3. **Autentifikácia** (*authentication*) – proces na identifikovanie a overenie entity, ktorá chce pristupovať k určitej informácii. Takáto entita sa nazýva autentifikovanou identitou, ak je dokázané napr. podľa mena a hesla, že je to presne ona. Príkladom môže byť prístup osoby do informačného systému. Systém ju identifikuje podľa mena a prístup povolí na základe správne zadaného hesla.

4. **Autorizácia** (*authorization*) – proces identifikujúci, ktorá identita má právo pristupovať ku ktorej informácii a akým spôsobom. Príkladom môže byť prístup k pracovným úlohám. Vedúci oddelenia má možnosť zadávať, prezerať

[2] Heš (*hash*) – reťazec znakov a čísiel stanovenej dĺžky, ktorý sa vypočíta pomocou špeciálnej matematickej (hešovacej) funkcie zo zdrojovej správy. Ak sa zmení obsah správy, zmení sa aj hodnota vypočítaného hešu.

a mazať pracovné úlohy pre pracovníkov oddelenia. Pracovníci môžu tieto úlohy iba čítať a meniť ich stav zo zadanej na vykonanú.

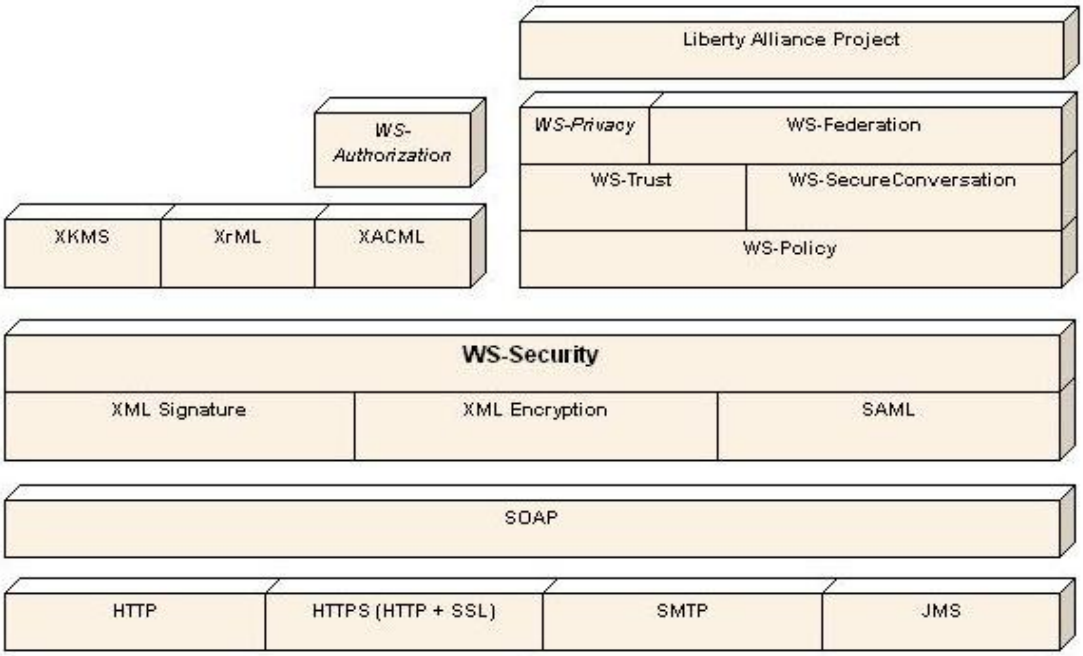
5. **Audit** (*auditing*) – uchovávanie záznamov o prístupoch (oprávnených ako aj neoprávnených) k informáciám a následná analýza týchto záznamov, ktorá môže prispieť k zvýšeniu bezpečnosti. Možnosť auditu prístupov k webovej službe závisí od samotnej implementácie webovej služby. Ak sú správy SOAP prenášané protokolom HTTP, je možné využiť audit protokolu HTTP na získanie informácií o komunikácii s webovou službou.

Zabezpečiť komunikáciu medzi klientom webovej služby a webovou službou znamená vytvoriť bezpečný kanál medzi týmito dvoma komunikujúcimi stranami. Bezpečný kanál by mal garantovať utajenie a integritu prenášaných správ, a tiež komunikujúce strany by mali byť navzájom autentifikované. Samozrejme definícia bezpečného kanálu závisí od presnej bezpečnostnej špecifikácie, ktorá je stanovená pre daný informačný systém.

Protokol SOAP je učený na realizáciu komunikácie medzi dvoma stranami, ale priamo neposkytuje mechanizmus na ochranu realizovanej komunikácie. Z tohto dôvodu vzniklo mnoho štandardov, ktoré sa snažia riešiť tento problém.

2.2 Súčasný stav v oblasti bezpečnostných štandardov webových služieb

V súčasnosti existuje viacero štandardov zaoberajúcich sa rôznymi spôsobmi zabezpečenia komunikácie s webovými službami. Obrázok 2.1 znázorňuje súčasný stav v oblasti týchto štandardov (resp. technológií) ako aj vzťah medzi nimi.



Obrázok 2.1: Súčasný stav v oblasti bezpečnostných štandardov webových služieb. Kurzíva na obrázku znázorňuje štandardy, ku ktorým ešte neexistujú konečné verzie špecifikácií.

WS-Security zastáva najvýznamnejšie miesto z pomedzi týchto štandardov. Dnes predstavuje najpoužívanejšiu technológiu na zabezpečenie komunikácie SOAP.

2.2.1 Krátky popis jednotlivých špecifikácií

- **XML Signature** – štandard definujúci, ako zabezpečiť integritu XML správ. Slúži zároveň aj ako mechanizmus na autentifikáciu komunikujúcich strán. Viac informácií o tomto štandarde sa nachádza v kapitole 2.2.2.

- **XML Encryption** - štandard definujúci, ako zabezpečiť utajenie obsahu XML správ resp. ich častí. Viac informácií o tomto štandarde je možné nájsť v kapitole 2.2.3.

- **SAML** (Security Assertion Markup Language) – technológia na prenos a overenie autentifikačných a autorizačných tvrdení (*assertions*) o určitej identite. Viac informácií o tejto technológii sa nachádza v kapitole 2.2.5.

- **WS-Security** – definuje spôsob zabezpečenia webových služieb s využitím XML Signature, XML Encyption a technológie SAML. WS-Security predstavuje jadro bezpečnosti webových služieb. Viac informácií o tomto štandarde je možné nájsť v kapitole 2.2.6.

- **WS-Policy** – technológia na popis pravidiel (*policies*) interakcie s webovými službami. Definuje vlastnosti, ktoré musí príslušná komunikácia SOAP spĺňať. Viac informácií o tejto technológii sa nachádza v kapitole 2.2.7.

- **XKMS** (XML Key Management Specification) – protokol na distribúciu a správu verejných kľúčov, ktoré sú vhodné na použitie v spojení s XML podpisom a XML šifrovaním. Viac informácií o tomto jazyku je možné nájsť v kapitole 2.2.8.

- **XACML** (eXtensible Access Control Markup Language) – je jazyk na zápis pravidiel (*policies*) a protokolu týkajúcich sa kontroly prístupu k určitému zdroju informácií. Viac informácií o tomto jazyku sa nachádza v kapitole 2.2.9.

- **XrML** (eXtensible Rights Markup Language) – je jazyk na špecifikáciu práv pre kontrolu prístupu k digitálnemu obsahu a službám. Viac informácií o tomto jazyku je možné nájsť v kapitole 2.2.10.

- **WS-SecureConversation** – technológia na vytvorenie bezpečného kanálu pri komunikácii s webovou službou. Viac informácií o tejto technológii je možné nájsť v kapitole 2.2.12.

- **WS-Trust** – definuje model na vytvorenie vzťahu dôvery medzi viacerými aplikáciami v rozdielnych bezpečnostných doménach, ktoré medzi sebou komunikujú prostredníctvom protokolu SOAP. Viac informácií o tomto štandarde sa nachádza v kapitole 2.2.13.

- **WS-Privacy** – je jazyk na zápis požiadaviek na zachovanie súkromia komunikujúcich strán pri komunikácii s webovou službou. Jazyk WS-Privacy bude postavený na technológiách WS-Policy, WS-Security a WS-Trust. Pri písaní tejto práce ešte neexistovala konečná verzia špecifikácie pre tento jazyk.

- **WS-Federation** – popisuje ako manažovať a sprostredkovať vzťah dôvery v heterogénnom prostredí. Viac informácií o tejto technológii sa nachádza v kapitole 2.2.14.

- **WS-Authorization** – definuje ako manažovať a pristupovať k pravidlám týkajúcim sa autorizácie webových služieb. Podobá sa vo veľkej miere technológii XACML, ale obmedzuje sa iba na potreby webových služieb. Špecifikácia pre túto technológiu v čase písania tejto práce ešte neexistovala.

2.2.2 XML podpis (XML Signature)

XML podpis predstavuje digitálny podpis zapísaný vo formáte XML. Slúži na podpis XML dokumentu resp. jeho častí, ale aj na podpis iných binárnych, nie XML údajov (napr. obrázku). XML podpis, tak ako každý digitálny podpis, sa dá použiť na zabezpečenie integrity komunikácie, autentifikácie komunikujúcich strán a na zabezpečenie nepopretia (*non-repudiation)* odoslania správy.

XML podpis sa skladá z dvoch základných častí:

1. **Referencia** - obsahuje popis miesta uloženia podpisovaných údajov, metódu, akou sa má vypočítať heš a hodnotu samotného hešu. Miesto uloženia dát

môže byť reprezentované viacerými spôsobmi, napr. ako odkaz na inú časť XML dokumentu, ako webová adresa na ktorej sa nachádza podpisovaný zdroj informácií a podobne. Príkladom metódy výpočtu hešu je algoritmus SHA1. V tejto časti môžeme špecifikovať transformáciu, ktorá sa má vykonať nad odkazovaným zdrojom, predtým ako je podpísaný. Príkladom je aplikovanie výrazu zapísaného pomocou jazyku XPath[3], aplikovanie transformácie XSLT[4], zakódovanie obsahu do formátu Base-64[5] a podobne.

2. **Zvyšné informácie a podpis** – obsahuje metódu kanonikalizácie (*canonicalization*), podpisovací algoritmus (napr. DSA-SHA1, RSA-SHA1), hodnotu podpisu, informáciu o použitých kľúčoch. Kanonikalizácia predstavuje proces normalizácie XML dokumentu. Dva dokumenty XML môžu byť sémanticky rovnaké, ale zapísané rozdielnym spôsobom. Môžu sa líšiť napr. znakmi konca riadkov (Windows vs. Unix), počtom a umiestnením prázdnych znakov, formátom a poradím atribútov, rôznym umiestnením špecifikátorov menného priestoru elementov XML (*namespace*) a podobne. Vypočítaný heš takýchto dvoch dokumentov XML bude odlišný, a preto sa vyžaduje normalizácia oboch dokumentov XML. K podpisu môže byť pripojená informácia o názve kľúča, referencia na kľúč, certifikát X.509 obsahujúci kľúč a podobne.

Pri generovaní XML podpisu sa najskôr získa obsah, ktorý sa má podpísať. Aplikuje sa naň definovaná transformácia (ak je definovaná aspoň jedna), vypočíta sa heš podľa zadaného hešovacieho algoritmu. Z týchto údajov sa vytvorí element XML (ďalej v tejto kapitole už iba element) `Reference` resp. viacero takýchto elementov. Vytvorí sa prázdny element `SignedInfo`, do ktorého sa uloží element `Reference` a zvyšné informácie o podpise (kanonikalizačná metóda, podpisovací algoritmus). Na element `SignedInfo` sa aplikuje zvolená kanonikalizačná metóda. Výsledok kanonikalizácie sa podpíše určeným algoritmom na podpisovanie. To znamená, že z kanonikalizovaného elementu `SignedInfo` sa vytvorí heš podľa definovaného hešovacieho algoritmu a následne sa tento heš zašifruje. Elementy `SignedInfo`, `SignatureValue` a `KeyInfo` predstavujú XML podpis reprezentovaný elementom `Signature`.

Overenie podpisu je podobný proces pri ktorom sa najskôr z prijatého obsahu vytvorí XML podpis. Hodnota prijatého XML podpisu sa rozšifruje a porovná sa s hodnotou

---

[3] XPath – XML jazyk na zápis výrazov pomocou ktorých je možné presne identifikovať určitú časť dokumentu XML.

[4] XSLT (Extensible Stylesheet Language Transformations) – technológia, pomocou ktorej je možné transformovať dokumenty XML do inej štruktúry (napr. na iný dokument XML, dokument HTML, ...). Transformácia je popísaná špeciálnym dokumentom XML (*stylesheet*), v ktorom je presne povedané na akú štruktúru sa jednotlivé časti zdrojového dokumentu majú transformovať.

[5]Kódovanie Base-64 (Base-64 encoding) – algoritmus na transformáciu binárnych dát do textovej podoby (reťazec ASCII znakov).

vygenerovaného XML podpisu. Ak sa tieto hodnoty rovnajú, prijatý obsah sa prehlási za platný.

Príklad XML podpisu:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
          <CanonicalizationMethod
                    Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <InclusiveNamespaces
                              xmlns=http://www.w3.org/2001/10/xml-exc-c14n#PrefixList="wsse enc env ns0 xsd xsi"/>
          </CanonicalizationMethod>
          <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <Reference URI="#XWSSGID-01">
                    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <DigestValue>
                              dp1dzJDWBZhiVXyf3SO…VSvFgNUA=
                    </DigestValue>
          </Reference>
          <Reference URI="#XWSSGID-02">
                    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                    <DigestValue>
                              oKatMB3cJ/zlFyK7QB…KIQwUgFW0=
                    </DigestValue>
          </Reference>
    </SignedInfo>
    <SignatureValue>
                    FTKMLzxuf4+4MUPju/skPJsS+nQm…ajGf6v306+pos=
    </SignatureValue>
    <KeyInfo xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
          <wsse:SecurityTokenReference xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/
                                                  oasis-200401-wss-wssecurity-utility-1.0.xsd"
                                       wsu:Id="XWSSGID-1143994086585-1830266609">
                    <wsse:Reference URI="#XWSSGID-1143994083871-1836311613"
                                   ValueType="http://docs.oasis-open.org/
                                       wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
   </KeyInfo>
</Signature>
```

Tento príklad znázorňuje XML podpis dvoch XML elementov, ktoré sa nachádzajú na inom mieste v XML dokumente (tieto elementy sú označené identifikátormi XWSSGID-01 a XWSSGID-02). Tieto dva elementy sú zahešované algoritmom SHA1 a vypočítaný heš je zašifrovaný algoritmom RSA. Verejný kľúč na overenie tohto podpisu sa nachádza v certifikáte X.509, na ktorý existuje odkaz v elemente `KeyInfo` XML podpisu.

Existujú tri rôzne typy XML podpisov:

1. **Enveloping** – zabaľuje v sebe obsah, ktorý podpisuje, t.j. v rámci elementu XML podpisu sa nachádza aj podpisovaný obsah.

2. **Enveloped** – podpis je uložený v XML elemente, ktorý podpisuje.

3. **Detached** – XML element podpisu nespĺňa bod *1* ani bod *2*. Môže ukazovať na iný XML dokument, element v inom XML dokumente, alebo na ľubovoľný iný zdroj, na ktorý existuje odkaz zapísaný prostredníctvom URI (Uniform Resource Identifier).

Viac informácií k XML podpisu možno nájsť v [2] a [3].

2.2.3 XML šifrovanie (XML Encryption)

XML šifrovanie slúži na zabezpečenie utajenia prenášaných informácií. Pôvodná informácia sa nahradí zmenenou informáciou, ktorá je čitateľná len pre entity, pre

ktoré je táto informácia určená. Pomocou XML šifrovania je možné utajiť XML dokument resp. jeho časť. Je možné zašifrovať rôzne časti XML dokumentu rovnakým alebo rôznym kľúčom.

Existujú 2 spôsoby ako sa môže šifrovaný XML element (ďalej v tejto kapitole už iba element) nahradiť XML šifrou (ďalej v tejto kapitole už iba šifra):

1. šifrou sa nahradí celý obsah elementu vrátane XML značiek,

2. šifrou sa nahradí iba obsah elementu.

Pri XML šifrovaní, podobne ako pri klasickom šifrovaní, sa používa väčšinou symetrická šifra (z dôvodu rýchlosti). Na kľúči symetrickej šifry (zdieľaný kľúč) sa komunikujúce strany buď vopred dohodnú, alebo sa tento kľúč zašifruje asymetrickou šifrou pomocou verejného kľúča prijímateľa. Takto zašifrovaný zdieľaný kľúč sa priloží k vytvorenej šifre. K tejto šifre sa tiež pripojí aj informácia o verejnom kľúči asymetrickej šifry. Podobne ako pri XML podpise, šifra môže obsahovať názov verejného kľúča, referenciu na verejný kľúč, identifikátor certifikátu X.509 obsahujúceho tento verejný kľúč a podobne.

XML šifra obsahuje nasledovné časti:

- informáciu o šifrovaných dátach – napr. pri šifrovaní obrázku vo formáte Base-64 bude šifra obsahovať atribút `MimeType` s hodnotou `'image/gif'` a atribút `Encoding` s hodnotou `'base64'`,

- informáciu o kľúči – táto informácia má veľmi podobnú štruktúru ako pri XML podpise (element `KeyInfo`),

- informáciu o type šifrovacieho algoritmu – napr. 3DES, AES a podobne ,

- hodnotu šifry vo formáte Base-64.

Príklad XML šifry:

```
<EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#"
          Type="http://www.w3.org/2001/04/xmlenc#Content">
          <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <EncryptionMethod Algorithm="…/xmlenc#tripledes-cbc"/>
                              <EncryptedKey>
                                        <EncryptionMethod
                                                  Algorithm="…/xmlenc#rsa-1_5"/>
                                        <ds:KeyInfo>
                                                  <ds:KeyValue>
                                                            <ds:RSAKeyValue>
                                                                      <ds:Modulus>
                                                                                7hgvqe…Ldq+tQ9W8=
                                                                      </ds:Modulus>
                                                                      <ds:Exponent>
                                                                                AQAB
                                                                      </ds:Exponent>
                                                            </ds:RSAKeyValue>
                                                  </ds:KeyValue>
                                        </ds:KeyInfo>
                                        <CipherData>
                                                  <CipherValue>
                                                            VAZ7Lwn7zbjE4PGvUYps+WurApjNa
                                                            ZQr/wjqcbc4uyKJRpJoZUwRC
                                                            S5DmIU3fc+zoQRvYgGj5B1i
                                                  </CipherValue>
                                        </CipherData>
                              <EncryptedKey>
          </ds:KeyInfo>
          <CipherData>
                    <CipherValue>
                              C0En9u/beS3yPkALaA/F7eBRwVAFtlxqOb
                              7f0MZ0JD6r3DDWfh+WoTdLSrx31NRs2gdQR5dlpPB32QA=
                    </CipherValue>
          </CipherData>
</EncryptedData>
```

Táto šifra bola vytvorená šifrovacím algoritmom 3DES pomocou zdieľaného kľúča. Tento zdieľaný kľúč bol zašifrovaný algoritmom RSA s využitím asymetrickej šifry, a bol uložený v elemente `EncryptedKey`. Prijímajúca strana si nájde súkromný kľúč asymetrickej šifry na základe deliteľa (element `Modulus`) a exponenta (element `Exponent`) uložených v elemente `RSAKeyValue`.

Viac informácií k XML šifrovaniu možno nájsť v [2] a [4].

2.2.4 Príklad scenára použitia XML šifrovania spolu s XML podpisom v prostredí informačného systému

Popis:

Pracovník personálneho oddelenia vkladá údaje o novoprijatom zamestnancovi do informačného systému. Tieto údaje majú byť poslané do viacerých aplikácií informačného systému (napr. do aplikácie mzdového oddelenia) ako dokument XML. Tento dokument sa skladá z dvoch častí. Prvá časť obsahuje všeobecné informácie o zamestnancovi (napr. priezvisko, meno, adresa, ...) a druhá obsahuje citlivé informácie (napr. výška mzdy, číslo účtu, ...) o ňom. Podmienkou je, aby druhá časť dokumentu bola viditeľná iba mzdovej aplikácii, a aby celý dokument mohli čítať iba aplikácie informačného systému. Ďalšou podmienkou je, aby tento dokument nebol zmenený pri prenose treťou stranou.

Náčrt riešenia:

1. Časť dokumentu, ktorá obsahuje citlivé informácie o novoprijatom zamestnancovi sa zašifruje XML šifrou s použitím verejného kľúča mzdovej

aplikácie. Len táto aplikácie bude schopná tieto informácie úspešne dešifrovať.

2. Dokument doplnený o šifru citlivých dát sa podpíše XML podpisom.

3. Následne sa celý dokument (pôvodný text + zašifrované citlivé informácie) zašifruje XML šifrou. Pri tomto šifrovaní sa použije symetrická šifra, ktorej kľúč poznajú všetky aplikácie informačného systému. Tým je zabezpečená ochrana tohto dokumentu pred ľubovoľnou entitou, ktorá sa napr. náhodne dostane k tomuto dokumentu a nemá právo tento dokument čítať.

Aplikácie informačného systému musia poznať presné poradie týchto krokov.

2.2.5 SAML (Security Assertion Markup Language)

SAML je jazyk na vytváranie tvrdení (*assertions*) o identitách. Ide o autentifikačné a autorizačné tvrdenia vo formáte XML. SAML tiež definuje protokol, pomocou ktorého je možne tieto tvrdenia overiť u dôveryhodnej autority (napr. autentifikačná alebo autorizačná autorita, ktorá pozná technológiu SAML).

Tieto tvrdenia predstavujú výroky (*claims*) dôveryhodnej autority o určitej identite. Príkladom môže byť nasledovný výrok: „Som autentifikačná autorita XY a pán Ľuboš Bisták, študent FMFI UK sa u mňa úspešne autentifikoval menom a heslom dňa 01.11.2005 o 11:15.“.

Jazyk SAML definuje tri typy tvrdení:

1. autentifikačné – definuje, akým spôsobom bola identita autentifikovaná. Príkladom je autentifikácia pomocou mena a hesla, XML podpisu, certifikátu X.509, lístka (*ticket*) Kerberos[6], hardvérového tokenu, kľúča poskytovaného technológiou XKMS (viac informácií o XKMS je možné nájsť v kapitole 2.2.8).

2. autorizačné – definuje ku ktorým zdrojom údajov a služieb (napr. webových) má identita právo pristupovať a akým spôsobom (napr. právo čítania, zapisovania, mazania).

3. tvrdenie o vlastnostiach (*attributes*) identity – predstavuje dodatočnú informáciu o identite. Príkladom môže byť ročník študenta, v ktorom sa práve nachádza.

---

[6] Lístok (*ticket*) Kerberos – lístok s obmedzenou časovou platnosťou určený na autentifikáciu. Tento lístok je vytvorený protokolom Kerberos na základe tajného kľúča, ktorý je uložený v centre na distribúciu kľúčov (Key Distribution Center).

Existuje spoločná množina informácií, ktorá sa vzťahuje na všetky tri typy tvrdení. Napr. verzia tvrdenia, definícia identity, popis autority, ktorá vydala toto tvrdenie, kedy vzniklo tvrdenie a aká je jeho časová platnosť, pre koho je toto tvrdenie určené, XML podpis tvrdenia a podobne.

Príklad autentifikačného tvrdenia zapísaného pomocou jazyka SAML:

```
<saml:Assertion
          Issuer="Bistak Corporation"
          IssueInstant="2005-11-01T11:15:30Z">
          <saml:AuthentificationStatement AuthentificationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
                                          AuthentificationInstant="2005-11-01T11:15:00Z ">
                              <sampl:Subject>
                                        <saml:NameIdentifier="urn:oasis:names:tc:
                                                  SAML:1.1:assertion#emailAddress>
                                                  1bistak@st.fmph.uniba.sk
                                        </saml:NameIdentifier>
                              </sampl:Subject>
          </saml:AuthentificationStatement>
</saml:Assertion>
```

Toto tvrdenie hovorí, že identita reprezentovaná adresou elektronickej pošty 1bistak@st.fmph.uniba.sk bola overená autentifikačnou autoritou Bistak Corporation prostredníctvom mena a hesla dňa 01.11.2005 o 11:15. Toto tvrdenie bolo vytvorené v ten istý deň o niečo neskôr.

Príklad autorizačného tvrdenia zapísaného pomocou jazyka SAML:

```
<saml:Assertion
          Issuer="Bistak Corporation"
          IssueInstant="2005-11-01T11:15:30Z">
          <saml:AuthorizationStatement
                              Decision="Permit"
                              Resource="http://dipl.bistak.sk/VIKUK-WS/report.xls">
                              <sampl:Subject>
                                        <saml:NameIdentifier="urn:oasis:names:tc:
                                                  SAML:1.1:assertion#emailAddress>
                                                  1bistak@st.fmph.uniba.sk
                                        </saml:NameIdentifier>
                              </sampl:Subject>
                              <sampl:Action Namespace="urn:oasis:names:tc:SAML:1.0:am:rwedc">
                                        Read
                              </sampl:Action
          </saml:AuthorizationStatement>
</saml:Assertion>
```

Toto tvrdenie hovorí, že identita reprezentovaná adresou elektronickej pošty 1bistak@st.fmph.uniba.sk má právo čítať MS Excel súbor na webovej adrese http://dipl.bistak.sk/VIKUK-WS/report.xls. Toto tvrdenie bolo vydané autorizačnou autoritou Bistak Corporation dňa 01.11.2005 o 11:15.

Ako bolo spomínané v úvode tejto kapitoly, SAML nie je len špecifikáciou jazyka na zápis tvrdení, ale aj technológiou, ktorá definuje spôsob prenosu a overovania týchto tvrdení. Technológia SAML má nasledovnú štruktúru:

1. **Tvrdenia** o identitách zapísané vo formáte XML.

2. **Protokol** na komunikáciu s dôveryhodnými autoritami.

   S týmito autoritami sa komunikuje prostredníctvom správ, ktoré obsahujú požiadavku na zoznam možných tvrdení vzťahujúcich sa k určitej identite. Autorizačná požiadavka môže byť chápaná ako: „Má táto identita oprávnenie

pristupovať k tomuto zdroju ?“. K nej prislúchajúca odpoveď: „Táto identita má možnosť čítať tento zdroj“.

3. **Napojenie na prenosný protokol** (*binding*) – popisuje konkrétny spôsob prenosu požiadaviek na vytvorenie resp. overenie tvrdenia o identite k dôveryhodnej autorite a aj prislúchajúcich odpovedí od tejto autority.

   Tieto požiadavky a odpovede sa môžu prenášať viacerými spôsobmi, ako napr. prostredníctvom HTTP, SMTP, JMS alebo SOAP. Pri prenose protokolom SOAP sa táto informácia môže nachádzať v hlavičke, ale aj v tele správy SOAP.

Technológiu SAML je možné skombinovať s XML podpisom a XML šifrovaním, a týmto spôsobom zabezpečiť utajenie a integritu tvrdení jazyka SAML.

SAML predstavuje pokročilejšie riešenie bezpečnosti, ktoré sa dá využiť v prostredí informačného systému na autentifikáciu a autorizáciu identít, ktoré chcú pristupovať k webovým službám v rámci tohto systému. Vyžaduje existenciu centrálneho mechanizmu na vytváranie a overovanie tvrdení jazyka SAML. Snahou SAML je vytvoriť digitálnu identitu, ktorá by existovala v jednom informačnom systéme predstavujúcom samostatnú bezpečnostnú doménu a mohla by byť použitá (resp. uznaná) v inom systéme, ktorý sa nachádza v inej bezpečnostnej doméne. Predpokladom pre tento model je existencia autority, ktorej by dôverovali oba systémy a mohli ju použiť na overenie tejto digitálnej identity. Funkčnosť tejto autority by mohla byť vystavená prostredníctvom webových služieb, čoby umožňovalo univerzálny prístup k tejto autorite z oboch systémov.

Presnú špecifikáciu toho štandardu možno nájsť v [5]. Doplňujúce informácie sa nachádzajú v [2].

2.2.6 WS-Security

Hlavným cieľom WS-Security je zabezpečenie protokolu SOAP. Nepredstavuje nový štandard v oblasti bezpečnosti (v zmysle nových bezpečnostných mechanizmov), ale snaží sa zabezpečiť webové služby s využitím XML podpisu, XML šifrovania a technológie SAML. Pri správnej kombinácii týchto troch štandardov spolu s využitím ďalších bezpečnostných mechanizmov (ako napr. lístku Kerberos, certifikátu X.509) sa dá zabezpečiť integrita a utajenie obsahu správ SOAP, autentifikácia a autorizácia komunikujúcich strán. Je tiež možné zabezpečiť nepopretie (*non-repudiation)* odoslania správy SOAP.

WS-Security presne definuje umiestnenie artefaktov týchto troch štandardov v správach protokolu SOAP. Väčšina týchto zabezpečovacích artefaktov sa nachádza v hlavičke správy SOAP. WS-Security definuje tri základné typy týchto artefaktov umiestnených v hlavičke správy SOAP:

1. **Bezpečnostný token** (*security token*) – predstavuje informáciu slúžiacu hlavne na autentifikáciu a autorizáciu komunikujúcich strán. WS-Security podporuje tri typy týchto tokenov:

a) token pozostávajúci z mena a hesla (UsernameToken) – namiesto hesla sa vloží heš vypočítaný z hesla a dodatočnej informácie (aktuálny čas, náhodný reťazec pozostávajúci zo znakov a čísiel). S vytvoreným hešom sa do tokenu pribalí táto dodatočná informácia, aby na strane servera bolo možné vygenerovať rovnaký heš. Viac informácií možno nájsť v [8].

b) binárny token (BinarySecurityToken) – binárny reťazec, ktorý zakódovaný určitým spôsobom do textovej podoby, napr. pomocou kódovania Base-64. Kódovanie do textovej podoby sa vyžaduje hlavne z dôvodu, že tento token má byť uložený v správe SOAP, ktorá predstavuje XML dokument. Takýto token môže obsahovať certifikát X.509v3 alebo lístok Kerberos.

c) iný token vo formáte XML – napr. token reprezentujúci tvrdenie SAML, token reprezentujúci licenciu zapísanú jazykom XrML , ...

2. **XML podpis** – digitálny podpis tela, hlavičky alebo celej správy SOAP.

3. **XML šifra** - v hlavičke správy SOAP sa väčšinou nachádza iba popis XML šifry, ktorou sú zašifrované jednotlivé časti tela správy SOAP. Samotná šifra je uložená väčšinou v tele správy. Na šifrovanie sa používa symetrický kľúč, ktorý je chránený asymetrickou šifrou.

Viac informácií o WS-Security je možné nájsť v [2] , [6] a [7].

2.2.6.1 Príklad použitia WS-Security v prostredí informačného systému

Tento príklad predstavuje náčrt riešenia problému popísaného v kapitole 2.2.4 pomocou technológie WS-Security.

Predpokladajme, že personalista zadáva údaje o novoprijatom zamestnancovi do aplikácie personálneho oddelenia. Táto aplikácia obsahuje klienta na komunikáciu s webovými službami. S aplikáciami informačného systému, do ktorých sa má dostať táto informácia o novoprijatom zamestnancovi, je možné komunikovať prostredníctvom protokolu SOAP.

Náčrt riešenia:

1. Personálna aplikácia uloží údaje o novoprijatom zamestnancovi vo formáte XML do tela správy SOAP. Citlivá časť dát sa naradí XML šifrou, ktorá sa vytvorí pomocou verejného kľúča mzdovej aplikácie. Šifra spolu s informáciou o kľúči sa uloží do tela správy.

2. Celá správa sa podpíše XML podpisom, ktorý sa vloží do hlavičky správy. Informácia o asymetrickej šifre tohto podpisu sa nachádza v certifikáte X.509, ktorý sa uloží ako binárny token v hlavičke správy.

3. Celý obsah tela správy (obsahujúci všeobecné informácie a šifru citlivých údajov) sa nahradí jeho symetrickou šifrou, ktorej kľúč poznajú všetky aplikácie informačného systému. Názov tohto zdieľaného kľúča sa priloží k informácii o kľúčoch XML šifry.

Aplikácie informačného systému buď poznajú presné poradie týchto krokov alebo je definované pomocou WS-Policy. Viac informácií o tejto technológii je možné nájsť v kapitole 2.2.7.

Takto vytvorená správa SOAP by mohla vyzerať asi takto:

```
<env:Envelope
          xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
          ID="XWSSGID-01">
<env:Header>
<wsse:Security
          xmlns:wsse="…-wss-wssecurity-secext-1.0.xsd" env:mustUnderstand="1">
          <wsse:BinarySecurityToken EncodingType="…#Base64Binary"
                                    ValueType="…-x509-token-profile-1.0#X509v3" Id="XWSSGID-02">
                              MIIC8zC…4oRYGlNh
          </wsse:BinarySecurityToken>
          <ds:Signature xmlns:ds="…/xmldsig#">
                    <ds:SignedInfo>
                              <ds:SignatureMethod Algorithm="…/xmldsig#rsa-sha1"/>
                              <ds:Reference URI="#XWSSGID-01">
                                        <ds:DigestMethod
                                                  Algorithm="…/xmldsig#sha1"/>
                                        <ds:DigestValue>WE…ewJxg=</ds:DigestValue>
                              </ds:Reference>
                    </ds:SignedInfo>
                    <ds:SignatureValue>RIKlES…P0rU=</ds:SignatureValue>
                    <ds:KeyInfo>
                              <wsse:SecurityTokenReference Id="XWSSGID-03">
                                        <wsse:Reference URI="#XWSSGID-02"
                                                  ValueType="…x509-token-profile-1.0#X509v3"/>
                              </wsse:SecurityTokenReference>
                    </ds:KeyInfo>
          </ds:Signature>
          <Timestamp Id="XWSSGID-04">
                    <Created>2005-11-01T11:15:01Z</Created>
                    <Expires>2005-11-01T11:20:01Z</Expires>
          </wsu:Timestamp>
          <xenc:EncryptedKey xmlns:xenc="…/xmlenc#">
                    <xenc:EncryptionMethod Algorithm="…/xmlenc#rsa-1_5"/>
                    <ds:KeyInfo xmlns:ds="…/xmldsig#">
                              <ds:KeyName>AppIS-SharedKey001</ds:KeyName>
                    </ds:KeyInfo>
                    <xenc:CipherData>
                              <xenc:CipherValue>RGpF+…d4/A=</xenc:CipherValue>
                    </xenc:CipherData>
                    <xenc:ReferenceList>
                              <xenc:DataReference URI="#XWSSGID-05"/>
                    </xenc:ReferenceList>
          </xenc:EncryptedKey>
</wsse:Security>
</env:Header>
<env:Body Id="XWSSGID-05">
          <xenc:EncryptedData xmlns:xenc="…/xmlenc#" Id="XWSSGID-1144088881553169731 4664"
                              Type="http://www.w3.org/2001/04/xmlenc#Content">
                    <xenc:EncryptionMethod Algorithm="…/xmlenc#tripledes-cbc"/>
                    <xenc:CipherData>
                              <xenc:CipherValue>
                                        6aKI3KNNFe…7w29SrI=
                              </xenc:CipherValue>
                    </xenc:CipherData>
          </xenc:EncryptedData>
</env:Body>
</env:Envelope>
```

Z dôvodu zvýšenia čitateľnosti tejto správy SOAP boli odstránené niektoré XML elementy resp. ich časti.

2.2.7 WS-Policy

WS-Policy slúži na zápis požiadaviek kladených na komunikáciu s určitou webovou službou. Tieto požiadavky sú vyjadrené pomocou pravidiel (*policies*), podľa ktorých

sa má táto komunikácia riadiť. Príkladom môže byť požiadavka na šifrovanie celej komunikácie SOAP algoritmom RSA. Táto požiadavka v sebe zahŕňa potrebu šifrovania správy, ktorá predstavuje volanie príslušnej operácie webovej služby, ale aj potrebu zašifrovať odpoveď reprezentujúcu výsledok tejto operácie. WS-Policy sa neobmedzuje iba na zápis bezpečnostných požiadaviek, ale aj na zápis požiadaviek kladených na iné domény ako potreba transakcií, nutnosť spoľahlivého doručovania správ SOAP, požiadavka na istý formát adresovania správ SOAP a podobne. WS-Policy taktiež popisuje spôsob, akým sa majú tieto požiadavky previazať s konkrétnou webovou službou.

Technológia WS-Policy sa dá rozdeliť na nasledovné časti:

1. **WS-PolicyAssertions** – štandard definujúci množinu všeobecných požiadaviek, ktorá sa dá špecifikovať pre ľubovoľnú webovú službu bez obmedzenia cieľovej domény, bez ohľadu na to, či sa jedná o bezpečnostné požiadavky alebo požiadavky na podporu určitého typu transakcií. Príkladom takýchto všeobecných požiadaviek je špecifikácia určitého kódovania a jazyku pre obsah správy SOAP. Napr. obsah správy SOAP musí byť kódovaný v ISO-8859-1 a napísaný v slovenskom jazyku.

2. **WS-PolicyAttachment** – štandard definujúci spôsob uloženia a prepojenia týchto požiadaviek s webovými službami. Existujú dva typy prepojenia:

    a) požiadavky sú definované v dokumente WSDL,

    b) požiadavky sú uložené mimo definície webovej služby – príkladom je externý XML dokument popisujúci tieto požiadavky. Obsahuje zoznam webových služieb, na ktorých sa tieto požiadavky vzťahujú. Môže obsahovať aj zoznam klientov prislúchajúcich k týmto službám. Takýmto spôsobom sa dá špecifikovať, že aj správa od webovej služby určená pre konkrétneho klienta musí byť zašifrovaná. V tomto prípade definuje požiadavku klienta, že tento klient prijíma iba zabezpečené údaje. Takáto požiadavka by sa nedala špecifikovať spôsobom popísaným v bode *a)*, lebo WSDL obsahuje informácie týkajúce sa iba webovej služby (a nie jej klientov).

3. **Štandardy definujúce požiadavky pre konkrétne domény** – napr. štandard definujúci bezpečnostné požiadavky webových služieb sa nazýva WS-SecurityPolicy. Pomocou tohto štandardu je možné špecifikovať požiadavku na autentifikáciu pomocou SAML tokenu, XML podpis a XML šifrovanie prenášanej správy SOAP, uloženie verejných kľúčov v binárnom bezpečnostnom tokene (napr. v certifikáte X.509v3), v hlavičke správy SOAP a podobne. Ako vidno z týchto príkladov, WS-SecurityPolicy využíva prvky štandardu WS-Security.

Príklad definície požiadaviek pomocou WS-SecurityPolicy:

```
<wsdl:definitions …>
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
            Name="policy1">
          <wsp:All>
                    <wsp:TextEncoding wsp:Usage="Required"  Encoding="ISO-8859-1"/>
                    <wsp:ExactlyOne>
                              <wsp:All>
                                        <wsse:SecurityToken
                                                  TokenType="wsse:Kerberosv5TGT"/>
                                        <wsse:Algorithm Type="wsse:AlgSignature" URI="…/xmlenc#aes" />
                              </wsp:All>
                              <wsp:All>
                                        <wsse:SecurityToken TokenType="wsse:X509v3" />
                                        <wsse:Algorithm  Type="wsse:AlgEncryption" URI="…/xmlenc#3des-cbc"/>
                              </wsp:All>
                    </wsp:ExactlyOne>
          </wsp:All>
</wsp:Policy>
          …
<wsdl:message name="zaregistrujNovehoZamestnanca" wsp:PolicyRefs="ps:policy1">
          …
</wsdl:message>
          …
</wsdl:definitions>
```

Tento príklad špecifikuje požiadavky na operáciu „zaregistrujNovehoZamestnanca“ určitej webovej služby. Parametre (vstupné a výstupné) tejto metódy musia byť v kódovaní ISO-8859-1. Na zabezpečenie tejto operácie sa má použiť buď lístok Kerberos spolu so šifrovacím algoritmom AES alebo certifikát X.509v3 so šifrovacím algoritmom 3DES. Pri vytváraní tohto príkladu som sa inšpiroval príkladmi, ktoré sú uvedené v [2] - ôsma kapitola.

Význam použitia technológie WS-Policy v prostredí informačného systému chápeme ako spôsob definície požiadaviek kladených na komunikáciu SOAP v tomto systéme. Príkladom môže byť definícia spôsobu a miesta aplikovania technológie WS-Security v prostredí informačného systému.

Viac informácií o tejto technológii je možné nájsť v [9].

2.2.8 XML Key Management Specification (XKMS)

XKMS je určený na uloženie a manažovanie verejných kľúčov asymetrickej šifry, ktoré sa využívajú pri XML podpise a XML šifrovaní. Kľúče a k nim prislúchajúce informácie sú zapísané vo formáte XML.

Samotné uloženie a manažovanie týchto kľúčov je realizované pomocou webových služieb, ku ktorým sa pristupuje prostredníctvom správ typu SOAP. Tieto správy vo svojom tele prenášajú žiadosti a odpovede na vykonanie určitej operácie nad kľúčmi. XKMS definuje protokol pre komunikáciu s dvoma typmi webových služieb:

- služba XKISS (XML Key Information Service) je určená na získavanie kľúčov a validáciu informácie spojenej s týmito kľúčmi,
- služba XKRSS *(*XML Key Registration Service) je určená na registráciu a správu kľúčov.

Príkladom využitia XKMS (okrem uloženia a manažovania kľúčov) je operácia služby XKISS, ktorá príjme podpísanú správu a na základe dodatočnej informácie (napr. identifikátora odosielateľa, certifikátu X.509, ...) v tejto správe overí platnosť podpisu.

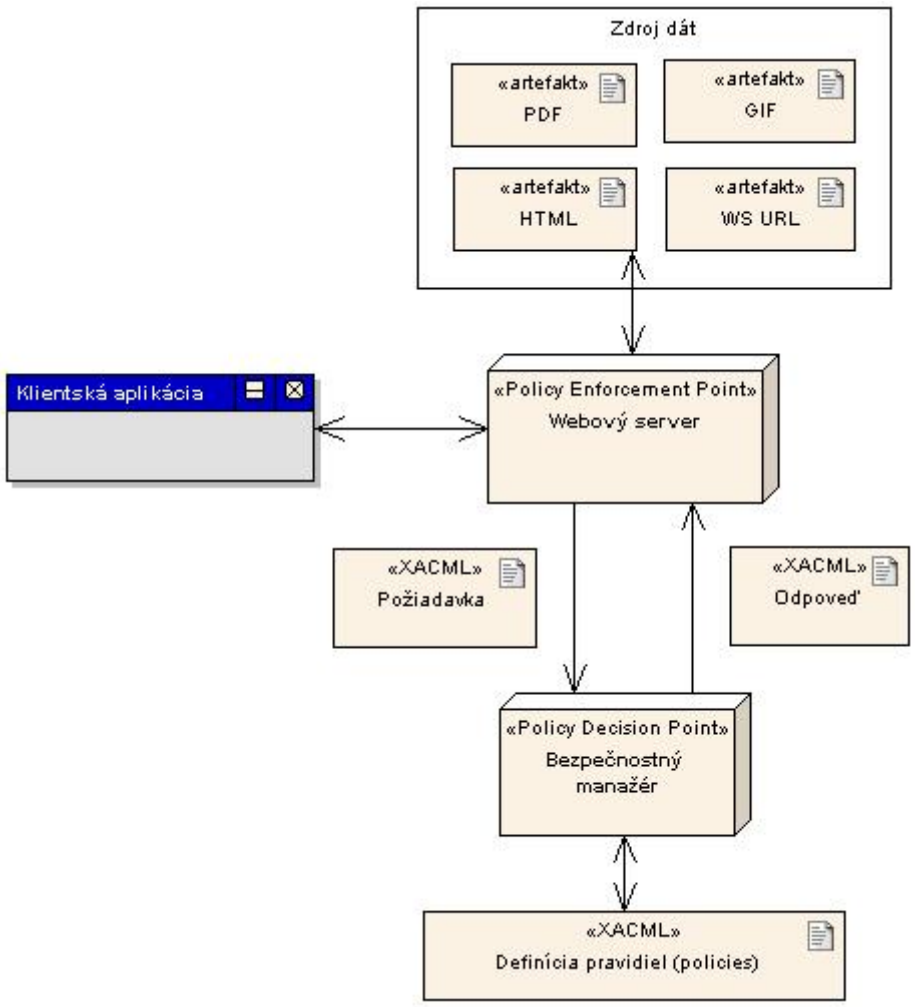
Využitie tejto technológie v prostredí informačného systému vidíme v oddelení webovej služby, ktorá vystavuje určitú funkčnosť resp. dáta od bezpečnostného mechanizmu, ktorý zabezpečuje komunikáciu SOAP. Predpokladáme, že táto technológia nájde uplatnenie hlavne v informačných systémoch, v ktorých sa intenzívne používajú zabezpečené webové služby, a ktorých zabezpečovací mechanizmus je postavený na centrálnej správe kľúčov.

Viac informácií možno nájsť v [10].

2.2.9 eXtensible Access Control Markup Language (XACML)

XACML je špecifikácia jazyka XML, pomocou ktorého je možné vytvoriť množinu pravidiel, ktoré definujú kto, kedy, za akých podmienok a akým spôsobom môže pristupovať k zdroju dát resp. informáciám. Definuje tiež formát a štruktúru správ XML protokolu, ktorý sa používa na prenos žiadostí a odpovedí týkajúcich sa povolenia resp. zamietnutia prístupu k určitému zdroju dát resp. informáciám.

Príkladom využitia jazyka XACML je informačný systém, v ktorom existuje webový server poskytujúci rôzne druhy artefaktov (napr. stránky HTML, dokumenty PDF, obrázky, webové služby, ...). V tomto systéme existuje klientská aplikácia žiadajúca o prístup k týmto artefaktom. S touto požiadavkou sa obráti na webový server (Policy Enforcement Point). Ten vygeneruje XACML požiadavku, ktorú pošle bezpečnostnému manažérovi. Pre všetky artefakty uložené na webovom serveri je v jazyku XACML zapísaná množina pravidiel (*policies*), ktoré definujú kto, za akých podmienok a s akými právami (napr. právo na čítanie, zápis), môže pristupovať k týmto artefaktom. Táto množina pravidiel je uložené ako XML dokument v bezpečnostnom manažérovi. Bezpečnostný manažér (Policy Decision Point) rozhodne na základe týchto pravidiel či povoliť, alebo zamietnuť prístup k žiadanému artefaktu. Vytvorí XACML odpoveď, ktorú pošle späť webovému serveru. Ten na základe tejto odpovede klientskej aplikácie poskytne požadovaný artefakt alebo ju informuje o zamietnutí jej požiadavky prostredníctvom chybovej správy. Tento príklad je zobrazený na obrázku 2.2 .



Obrázok 2.2: Scenár použitia jazyka XACML.

XACML predstavuje pokročilé riešenie autorizácie. Umožňuje oddeliť dáta resp. služby (napr. webové) od definície pravidiel prístupu k nim. Na druhej strane si toto riešenie vyžaduje existenciu bezpečnostného manažéra, ktorý podporuje jazyk XACML.

2.2.10 eXtensible Rights Markup Language (XrML)

Jazyk XrML je jeden zo stavebných jednotiek v snahe o vytvorenie jednotného manažmentu digitálnych práv (*digital rights management*), ktorý by kontroloval prístup k digitálnemu obsahu (napr. k dokumentu PDF, k zvukovému súboru, k obrázku) a službám (napr. webovým), ktoré sú distribuované verejnými sieťami. XrML sa snaží zohľadniť autorské práva, a teda rozlišuje medzi plateným a voľne šíriteľným obsahom a službami. Pomocou tohto jazyka sa dá vytvoriť licencia vo formáte XML, v ktorej je definovaný držiteľ licencie, na aký digitálny obsah alebo službu sa licencia vzťahuje, koľko a akým spôsobom treba zaplatiť za používanie licencie. Môžeme povedať, že táto technológia prekračuje hranice podnikového informačného systému. Jediné využitie tejto technológie vidím ako jeden zo spôsobov

poskytovania dokumentov a služieb informačného systému v platenej forme okolitému svetu.

2.2.11 Porovnanie SAML, XACML a XrML

Všetky tieto tri technológie sa venujú autorizácii, ale každá trochu z iného uhla pohľadu. SAML v porovnaní s XACML a XrML vytvára mechanizmus na šírenie informácie o autorizácii, t.j. špecifikuje tvrdenie, že daná identita bola autorizovaná určitým spôsobom na prístup k určitým informáciám. Toto tvrdenie sa dá použiť v ďalšej komunikácii. Koncová strana si môže kedykoľvek overiť platnosť tohto tvrdenia (napr. pomocou protokolu SAML alebo pomocou XACML). Na druhej strane XACML a XrML hlavne definujú pravidlá resp. práva, podľa ktorých je možné danú identitu autorizovať. XACML v porovnaní s XrML hovorí o pravidlách podľa riadenia procesu autorizácie, pričom XrML sa venuje skôr licenciám, podľa ktorých je jasne definovaný proces autorizácie identity. Existuje možnosť integrácie technológie SAML s technológiou XACML, kde XACML sa použije ako mechanizmus na overenie autorizačných tvrdení SAML. Príkladom môže byť správa SOAP, ktorá je obohatená o autorizačné tvrdenie SAML. Webový server, na ktorom je nasadená volaná webová služba, iniciuje XACML komunikáciu s bezpečnostným manažérom, ktorou si overí pravdivosť obsahu autorizačného tvrdenia SAML.

2.2.12 WS-SecureConversation

WS-SecureConversation slúži na vytvorenie bezpečného kanálu pri komunikácii s webovými službami. Jeho hlavnou ideou je vytvorenie bezpečnostného kontextu, v ktorom by sa prenášala informácia (napr. zdieľaný kľúč pre potreby XML šifrovania) potrebná na zabezpečenie komunikácie s webovými službami. Tento kontext by sa vytvoril iba raz (napr. iba raz by sa vytvoril zdieľaný kľúč XML šifrovania pomocou asymetrickej šifry), a to pri prvom volaní webovej služby a pri ďalšom prístupe k tejto službe by sa znovu využil na zabezpečenie komunikácie s touto službou. Na podobnom princípe je založený protokol SSL[7]. Na rozdiel od protokolu SSL, sa tento kontext využíva pri komunikácii s viacerými webovými službami, pričom SSL sa obmedzuje iba na dve komunikujúce strany (*peer-to-peer connection*).

Využitie tejto technológie predpokladáme v informačnom systéme, v ktorom existuje pravidelná a veľmi intenzívna komunikácia medzi jeho jednotlivými subsystémami reprezentovanými webovými službami. Táto technológia by napomohla k zrýchleniu komunikácie medzi týmito subsystémami, a tiež k zvýšeniu

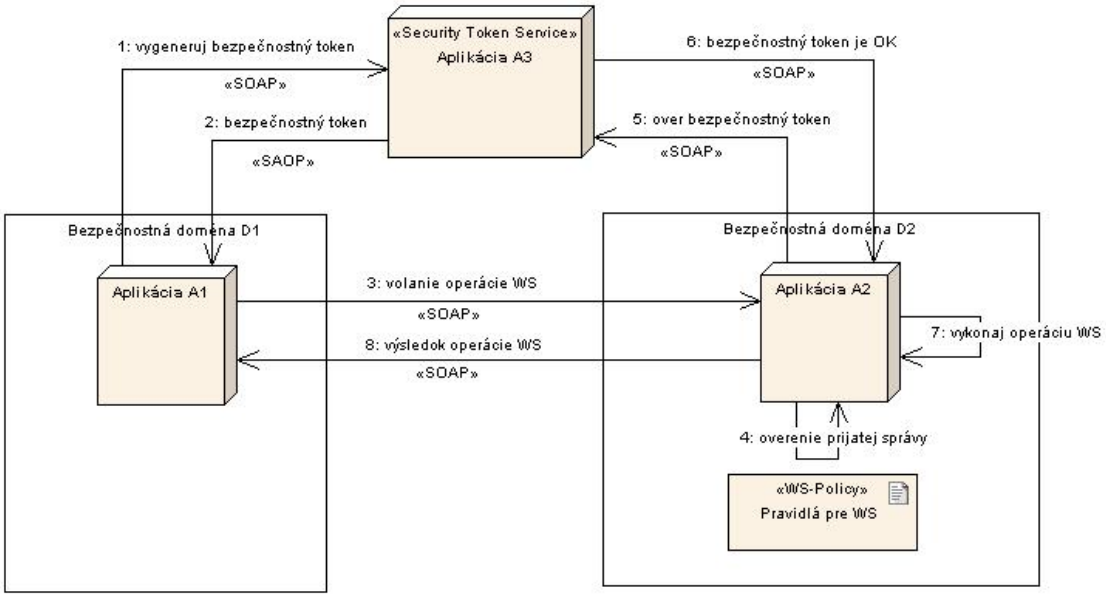
[7] SSL (*Secure Socket Layer*) – protokol na zabezpečenie spojovacej (*session*) vrstvy. Na začiatku sa pomocou asymetrickej šifry dohodne zdieľaný kľúč symetrickej šifry, ktorým sa šifruje zvyšná komunikácia medzi komunikujúcimi stranami.

bezpečnosti v tomto systéme (obmedzenie použitia verejných kľúčov vedie ku zníženie možnosti resp. rizika odhalenia kľúčov asymetrickej šifry).

2.2.13 WS-Trust

Štandard WS-Trust definuje model resp. mechanizmus na vytvorenie vzťahu dôvery medzi aplikáciami, ktoré medzi sebou vystavujú svoju funkčnosť a dáta prostredníctvom webových služieb.

Aplikácia A1 napr. pomocou WS-Policy definuje pravidlá komunikácie s jej webovými službami. Aplikácia A2 pošle aplikácii A2 požiadavku na vykonanie určitej operácie jej webovej služby. Aplikácia A2 pridá do správy SOAP dodatočnú informáciu (napr. certifikát X.509, meno a heslo), podľa ktorej by sa malo dať určiť, ku ktorým webovým službám a akým spôsobom môže aplikácia A2 vo všeobecnosti pristupovať. Problém je, že aplikácie A1 a A2 sa nachádzajú v rozdielnych bezpečnostných doménach (D1 a D2), v ktorých platia rozdielne bezpečnostné pravidlá. To znamená, že vzniká otázka, akým spôsobom si aplikácia A1 overí, že údaje, ktoré aplikácia A2 vložila do komunikácie SOAP sú pravdivé. Tento problém sa snaží riešiť technológia WS-Trust, ktorá definuje tretiu centrálnu aplikáciu A3. Tejto aplikácii dôverujú obe aplikácie (t.j. A1 aj A2). Aplikácia A3 (Security Token Service) má na starosti generovanie a overovanie bezpečnostných tokenov na základe ktorých si budú aplikácie A1 a A2 navzájom dôverovať. S aplikáciou A3 je možné komunikovať prostredníctvom protokolu SOAP. Prostredníctvom aplikácie A3 sa medzi aplikáciou A1 a A2 vytvorí vzťah dôvery. Komunikácia medzi aplikáciami A1, A2 a A3 je znázornená na obrázku 2.3. WS-Trust presne definuje tieto tokeny a protokol, akým sa dajú vygenerovať a overiť.



Obrázok 2.3: Vytvorenie vzťahu dôvery pomocou WS-Trust medzi dvoma aplikáciami z rozdielnych bezpečnostných domén.

WS-Trust stavia na bezpečnostných technológiách WS-Policy a WS-Security.

Môžeme povedať, že technológie WS-Trust a SAML sa podobajú v spôsobe overovania identít. Rozdielom je, že WS-Trust sa sústreďuje na mechanizmus akým sa tieto identity overujú, pričom SAML sa skôr sústreďuje na tvrdenia o identite a na formu ich prenosu resp. distribúcie medzi webovými službami.

Ako miesto spoločného využitia technológií WS-Trust a SAML si môžme predstaviť informačný systém univerzity, v ktorom by každá fakulta predstavovala samostatnú bezpečnostnú doménu. Vzťah dôvery medzi fakultami by sa vytvoril pomocou technológie WS-Trust. Identity jednotlivých aplikácií a používateľov fakultných informačných systémov by sa reprezentovali tvrdeniami jazyka SAML.

2.2.14 WS-Federation

WS-Federation predstavuje technológiu na preklad identít medzi webovými službami v rôznych doménach dôvery. Doména dôvery predstavuje množinu bezpečnostných domén, medzi ktorými existuje vzťah dôvery realizovaný napríklad pomocou technológie WS-Trust. Technológiu WS-Federation si môžeme predstaviť ako nadstavbu technológie WS-Trust.

Príkladom môžu byť dve domény D1 a D2. V doméne D1 je vytvorený vzťah dôvery pomocou bezpečnostného tokenu, ktorý predstavuje certifikát X.509. V doméne D2 je vytvorený vzťah dôvery pomocou bezpečnostného tokenu, ktorý predstavuje lístok Kerberos. WS-Federation definuje mechanizmus, ktorý zabezpečí preklad certifikátu X.509 na lístok Kerberos a naopak, t.j. vytvorí zjednotenú (*federated*) identitu, ktorá bude akceptovaná v doméne D1 aj v doméne D2.

Fyzicky tento preklad identít môže byť realizovaný nasledovne. Ak aplikácia A1 z domény D1 bude chcieť pristupovať k webovej službe v doméne D1, tak použije certifikát X.509, ktorý jej vygenerovala aplikácia T1, ktorá vytvára a overuje bezpečnostné tokeny v doméne D1. Ak bude chcieť aplikácia A1 pristupovať k webovej službe aplikácie A2 v doméne D2, A1 pošle svoj certifikát X.509 aplikácii T2 (aplikácia vytvárajúca a overujúca bezpečnostné tokeny v doméne D2). T2 pomocou T1 overí tento certifikát X.509 a vygeneruje k nemu „ekvivalentný“ lístok Kerberos. S týmto lístkom aplikácia A1 osloví webovú službu aplikácie A2.

WS-Federation využíva technológie WS-Security, WS-Policy, WS-Trust a WS-SecureConversation.

2.2.15 Projekt Liberty Alliance

Hlavným cieľom tohto projektu je vytvorenie spoločného manažmentu identít medzi viacerými aplikáciami resp. aplikačnými doménami. Rieši rovnaké problémy ako technológia WS-Federation, ale neobmedzuje sa iba na webové služby. Tento projekt je schopný spolupracovať s riešeniami založenými na WS-Federation. Podobne ako WS-Federation sa snaží vytvoriť zjednotenú (*federated*) identitu, ktorá by sa automaticky propagovala medzi viacerými aplikáciami, s ktorými je možné komunikovať, napr. prostredníctvom webových služieb. Tento projekt sa snaží riešiť

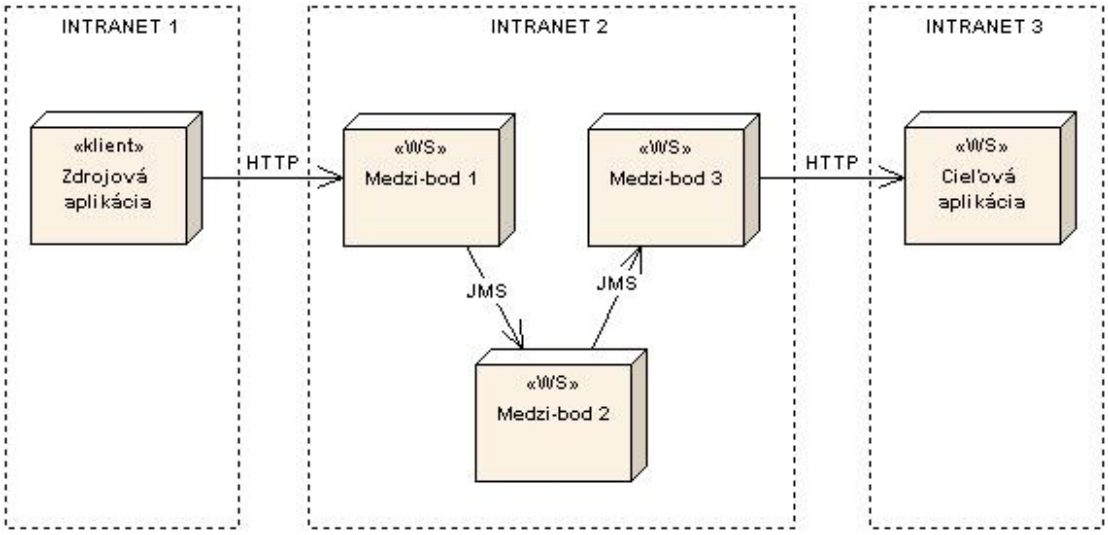
univerzálnu identitu pomocou tvrdení jazyka SAML. Tieto tvrdenia by predstavovali informáciu, pomocou ktorej by bolo možné túto zjednotenú identitu identifikovať.

Viac informácií o tomto projekte je možné nájsť v [11].

2.2.16 Zabezpečenie komunikácie na rôznych úrovniach

Komunikácia medzi webovou službou a klientom webovej služby prebieha na dvoch úrovniach – na úrovni transportnej vrstvy (napr. protokolom HTTP, SMTP, FTP alebo pomocou JMS) a na úrovni aplikačnej vrstvy (protokolom SOAP). Napriek tomu, že sieťový referenčný model ISO OSI vníma protokoly HTTP, SMTP a FTP ako protokoly aplikačnej vrstvy, z pohľadu webových služieb sú vnímané ako protokoly nižšej vrstvy, ktoré sú určené na prenos (transport) správ SOAP. V ďalšom texte tejto kapitoly budeme preto hovoriť o týchto protokoloch ako o protokoloch transportnej vrstvy. Vďaka rozdeleniu komunikácie s webovou službou na dve rôzne úrovne je možné riešiť zabezpečenie komunikácie na každej úrovni samostatne alebo súčasne na oboch úrovniach.

Vo všeobecnosti môže správa SOAP prechádzať viacerými „medzi-bodmi“, kým sa dostane k cieľovému adresátovi. Príklad takejto komunikácie je znázornený na obrázku 2.4. „Medzi-body“ nemusia hrať v tejto komunikácii iba pasívnu úlohu. Každý z nich môže mať možnosť čítať a meniť časť obsahu správy SOAP. Im určená časť obsahu správy bude zašifrovaná ich verejným kľúčom (t.j. iba oni ju budú môcť prečítať a meniť).



Obrázok 2.4: Príklad prenosu správy SOAP. (WS je skratka pre webové služby.)

Tento príklad komunikácie sa dá aplikovať na scenár v kapitole 2.2.6.1, ktorý popisuje zadanie a propagáciu údajov o novoprijatom zamestnancovi v informačnom systéme. Klientská časť aplikácie personálneho oddelenia vytvorí z údajov o tomto zamestnancovi dokument XML, ktorý vloží do správy SOAP. Túto správu najskôr pošle aplikácii oddelenia správy informačného systému (napr. na vytvorenie konta elektronickej pošty), potom sa táto správa pošle mzdovej aplikácii a nakoniec sa pošle

do ostatných aplikácií informačného systému, do ktorých sa majú dostať údaje o novoprijatom zamestnancovi.

Ďalším príkladom využitia „medzi-bodov“ v komunikácii s cieľovou webovou službou je transformácia obsahu správ SOAP. Cieľová služba môže požadovať, aby bol prenášaný obsah zapísaný v odlišnom formáte od formátu, ktorý posiela zdrojová aplikácia (napr. ako dokument v grafickom formáte SVG alebo ako dokument jazyka MathXML). „Medzi-body“ sa dajú tiež využiť pre potreby uchovávania záznamov (*logging*) o komunikácii s cieľovou webovou službou.

Na zabezpečenie protokolov transportnej vrstvy (HTTP, SMTP, FTP) sa najčastejšie používa protokol SSL. Príkladom je HTTPS (HTTP + SSL). SSL sa môže použiť aj na zabezpečenie JMS. V týchto prípadoch ide o priame zabezpečenie komunikácie medzi dvoma aplikáciami (*peer-to-peer*) na úrovni transportnej vrstvy, ide o zabezpečenie správy SOAP iba na jednej časti cesty k cieľovej webovej službe. Na obrázku 2.4 by SSL zabezpečil komunikáciu medzi zdrojovou aplikáciou a prvým „medzi-bodom“. Prvý „medzi-bod“ by po prijatí správy SOAP mohol vidieť celý obsah tejto správy. Pri použití SSL sa najčastejšie šifruje celá komunikácia rovnakým šifrovacím algoritmom s použitím rovnakého kľúča

Na zabezpečenie protokolu SOAP sa používa technológia WS-Security. Pomocou WS-Security je možné zabezpečiť komunikáciu ľubovoľných dvoch bodov na úrovni aplikačnej vrstvy, ktoré spolu komunikujú protokolom SOAP. V tomto prípade sú správy SOAP chránené počas celej svojej cesty až k cieľovej webovej službe. „Medzi-body“ môžu čítať iba tie časti správy, ktoré sú pre ne určené. Každá časť správy SOAP môže byť zašifrovaná iným algoritmom s využitím rôznych kľúčov.

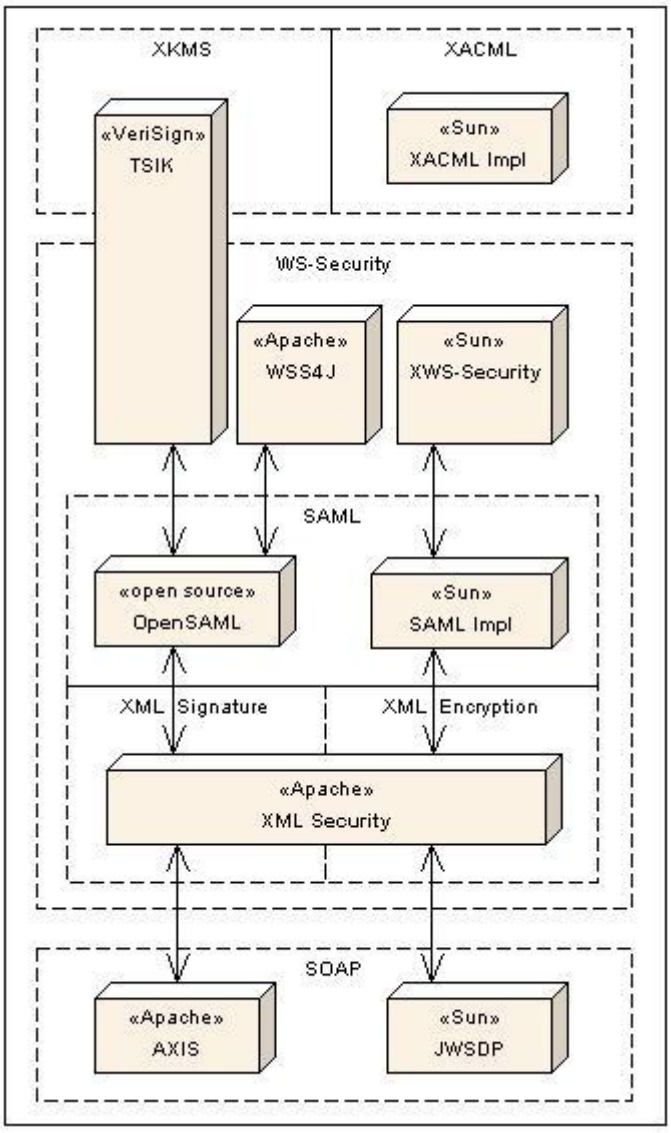
SSL ako aj WS-Security využívajú rovnaké bezpečnostné artefakty, ako je digitálny podpis, šifrovanie, certifikát X.509 a podobne. Hlavným rozdielom medzi týmito dvoma riešeniami je miesto, na ktorom sa snažia chrániť prenášanú informáciu. Použitie SSL v porovnaní s WS-Security predstavuje jednoduchšiu a rýchlejšiu alternatívu zabezpečenej komunikácie. Na druhej strane, na rozdiel od SSL umožňuje WS-Security do správ SOAP zahrnúť aj autorizačné údaje.

Dosť často sa kombinuje zabezpečenie webových služieb so zabezpečením prenosového kanálu. Konkrétne ide o prenos správ SOAP cez HTTPS, ktoré sú chránené pomocou WS-Security. Príkladom je správa SOAP šifrovaná pomocou SSL, ktorá v hlavičke obsahuje token reprezentujúci meno a heslo pre potreby autentifikácie a autorizácie.

2.3 Súčasný stav v oblasti implementácie bezpečnostných štandardov webových služieb

V dnešnej dobe existuje viacero implementácií štandardov z oblasti bezpečnosti webových služieb. Medzi ne partia uzavreté implementácie[8] rôznych spoločností ako aj implementácie založené na filozofii *open source* (viac informácií o tejto filozofii možno nájsť v [12]). Obrázok 2.5 znázorňuje implementácie, ktoré sa nám podarilo nájsť a preskúmať. Taktiež zobrazuje vzťah a závislosti medzi týmito implementáciami.

[8] Uzavretá implementácia – jej zdrojový kód je dostupný len autorovi tohto kódu a osobám, ktorej ho autor poskytol.



Obrázok 2.5: Implementácie bezpečnostných štandardov webových služieb. (Skratka „Impl“ reprezentuje slovo implementácia.)

Popis jednotlivých implementácií zobrazených na obrázku 2.5 sa nachádza vo zvyšnej časti tejto kapitoly.

2.3.1 Implementácie technológií XML podpis a XML šifrovanie

2.3.1.1 Apache XML Security

Cieľom tohto *open source* projektu je vytvorenie programových rozhraní (API) a implementácií pre bezpečnostné štandardy týkajúce sa jazyka XML. V dnešnej dobe sa zameriava na XML podpis a XML šifrovanie. Ďalším cieľom je rozšíriť tento projekt o rozhranie a implementáciu ďalšej špecifikácie W3C z oblasti manažmentu XML kľúčov (XML Key Management). V súčasnosti je tento projekt implementovaný v dvoch verziách ako knižnica Java a knižnica C++.

V budúcnosti sa plánuje nahradenie programových rozhraní tohto projektu rozhraniami špecifikovanými vo JSR-105 a JSR-106. Viac informácií o týchto špecifikáciách je možné nájsť v kapitole 2.3.3.3.1.

Súčasťou projektu nie je implementácia bezpečnostných algoritmov. Autormi projektu je odporúčaná knižnica Java Cryptography Extension (JCE) skupiny Bounty Castle (viac v [13]). Ďalšou možnosťou je použitie implementácie bezpečnostných algoritmov v rámci J2SE 5.0 (Java 2 Standard Edition, verzia 5.0).

Podarilo sa nám vyskúšať funkčnosť knižníc Java tohto projektu na jednoduchej aplikácii vytvorenej v jazyku Java, ktorá je schopná vytvoriť XML podpis (aj ho overiť) a XML šifru pre zadaný dokument XML. S rozhraniami tejto knižnice sa pracovalo veľmi jednoducho a intuitívne. Považujeme túto knižnicu za veľmi dobrý a spoľahlivý nástroj na prácu s XML podpisom a XML šifrou.

Viac informácií o tomto projekte možno nájsť v [14].

2.3.2 Implementácie technológie SAML

2.3.2.1 OpenSAML

Projekt OpenSAML predstavuje *open source* implementáciu jazyka SAML. Je vyvíjaný členmi konzorcia Internet2. Konzorcium Internet2 je vedené americkými univerzitami v spolupráci s komerčnými spoločnosťami a vládou. Jeho cieľom je vývoj technológií na zlepšenie fungovania internetu.

OpenSAML je množina Java a C++ knižníc, ktoré sa využívajú pri vytváraní, prenose a spracovaní správ SAML. OpenSAML podporuje prepravu požiadaviek a odpovedí vo formáte SAML cez protokol SOAP. Umožňuje používať správy SAML na prenos bezpečnostných informácií medzi rôznymi systémami. Je rozšíriteľný a navrhnutý tak, aby mohol podporovať rôzne bezpečnostné modely a požiadavky.

Otestovali sme knižnicu Java tohto projektu pomocou príkladov, ktoré boli priložené k tejto knižnici. Tieto príklady veľmi dobre demonštrujú ako pracovať s artefaktmi jazyka SAML pomocou tejto knižnice.

Viac informácií možno nájsť v [15] a [16].

2.3.2.2 JSR-155: Web Services Security Assertions

JSR[9]-155 je návrh špecifikácie programového rozhrania pre vytváranie a prácu s tvrdeniami jazyka SAML, v aplikáciách postavených na platforme Java. Je založený na štandarde SAML. Opiera sa o programové rozhrania pracujúce s jazykom XML a technológiami XML podpis a XML šifrovanie.

JSR-155 sa dnes nachádza iba v štádiu vývoja a nie je stanovený termín jeho ukončenia. Viac informácií možno získať v [17].

2.3.3 Implementácie technológie WS-Security

2.3.3.1 JSR-183: Web Services Message Security APIs

Hlavným cieľom tohto návrhu špecifikácie je umožniť aplikáciám postaveným na platforme Java bezpečne komunikovať pomocou správ protokolu SOAP. Snahou je vytvoriť programové rozhranie pre bezpečnostný model, ktorého základom by boli bezpečnostné tokeny vyjadrujúce výroky (*claims*) o identitách, sprevádzané digitálnym podpisom a šifrovaním. Ciele návrhu tejto špecifikácie sa podobajú cieľom štandardu WS-Security.

JSR-155 sa v dnešnej dobe nachádza iba v úvodnej fáze vývoja a nie je stanovený termín jeho ukončenia. Viac informácií možno získať v [17].

2.3.3.2 Sun JWDP a Apache Axis

Kapitoly 2.3.3.3 a 2.3.3.4 sa venujú dvom implementáciám technológie WS-Security, ktoré sú postavené na produktoch Sun JWSDP (Java Web Services Developer Pack) a Apache Axis. Pre lepšie pochopenie týchto dvoch implementácií WS-Security je potrebné mať základnú predstavu o produktoch Sun JWSDP a Apache Axis. Preto v ďalšom texte uvádzame základný popis týchto dvoch produktov.

Produkty Sun JWSDP a Apache Axis slúžia na vývoj, testovanie a nasadenie webových služieb. Webové služby sú v týchto produktoch implementované pomocou technológie Java, podľa štandardu JAX-RPC. Obsahujú nástroje pomocou ktorých je možné automaticky vygenerovať podporné artefakty webovej služby a artefakty pre klienta webovej služby.

---

[9] Špecifikácie JSR-* (Java Specification Request) sa vytvárajú v rámci otvoreného procesu Java Community Process (JCP), ktorý zoskupuje odborníkov v oblasti informačných technológií z rôznych spoločností. Cieľom tohto procesu je vytváranie špecifikácií a definícií programových rozhraní (API) pre rôzne technológie založené na platforme Java. Viac informácií o tomto procese je možné nájsť v [17].

Artefakty webovej služby predstavujú množinu konfiguračných súborov, ktoré definujú dodatočné informácie o webovej službe (napr. presný spôsob mapovania komplexného elementu XML na triedu jazyka Java). Na základe údajov uložených v týchto súboroch, webová služba dokáže mapovať volania zapísané v správach SOAP na volania metód konkrétnych objektov aplikácie.

Klientské artefakty predstavujú súbor tried jazyka Java (*stubs*), ktoré zabezpečujú preklad volaní metód klientskej aplikácie na správy SOAP a opačne. To znamená, že klient webovej služby implementovaný v jazyku Java, zavolá metódu takejto triedy, ktorá vytvorí a odošle správu SOAP príslušnej webovej službe. Po prijatí odpovede preloží výsledok popísaný v prijatej správe SOAP na návratovú hodnotu metódy.

2.3.3.2.1 JAX-RPC (Java APIs for XML based RPC)

JAX-RPC predstavuje špecifikáciu programového rozhrania pre volanie vzdialených procedúr (*remote procedure call - RPC*) vytvorených v jazyku Java pomocou dokumentov XML (v jazyku Java sú pojmy procedúra a funkcia vyjadrené pojmom metóda). Samotná požiadavka na vykonanie metódy je zapísaná dokumentom XML. JAX-RPC špecifikuje presný postup ako prekladať (*serialize*) volania metód na dokumenty XML, a tiež ako prekladať dokumenty XML na volania metód (*deserialize*). Tento preklad v sebe zahŕňa mapovanie štruktúr jazyka XML na ekvivalentné štruktúry jazyka Java a aj opačné mapovanie. Príkladom je mapovanie triedy resp. rozhrania jazyka Java na komplexnú štruktúru jazyka XML.

Často sa JAX-RPC používa ako prostriedok na vytváranie webových služieb v platforme Java. Existujú dva spôsoby ako vystaviť funkčnosť napísanú v jazyku Java formou webových služieb pomocou JAX-RPC:

- Ako upravené triedy technológie Java Servlet - aplikácia vytvorená v tejto technológii sa skladá z množiny tried a konfiguračných súborov, ktoré sú umiestnené vo webom kontajneri. Tieto triedy sledujú komunikáciu HTTP na určitej adrese a porte, ktorá prenáša správy SOAP. Prijatú správu SOAP transformujú na príslušné volanie metódy (triedy implementujúcej vystavenú funkčnosť). Výsledok tejto metódy zapíšu do správy SOAP, ktorú pošlú späť klientovi webovej služby.
- Ako komponenty technológie Enterprise Java Beans (EJB) – tieto komponenty sa nachádzajú v kontajneri EJB. Kontajner EJB preloží správy SOAP na príslušné volania metód týchto komponentov.

Technológie Java Servlet a Enterprise Java Beans sú súčasťou Java 2 Eneterprise Edition (J2EE). Viac informácií o týchto technológiách v spojení s webovými službami je možné nájsť v [18].

Špecifikácia JAX-RPC bola vyvinutá ako JSR-101. JAX-RPC 1.1 bola rozšírená a premenovaná na JAX-WS 2.0. Špecifikácia JAX-WS 2.0 je v dnešnej dobe vyvíjaná ako JSR-224. Nástroj JWSDP využíva časť *open source* projektu GlassFish[10], ktorá implementuje JAX-RPC.

Viac informácií možno nájsť v [1].

2.3.3.2.2 Rozdiely v produktoch

Oba produkty majú veľa spoločného, ale aj napriek tomu existujú vlastnosti, ktorými sa líšia. Tabuľka 2.1 znázorňuje rozdiely medzi týmito dvoma produktmi.

	JWSDP	**Axis**
Autor	Sun Microsystems Corporation	Apache Software Foundation
Dostupnosť a cena	bezplatný prístup	open source
Cieľová platforma	Sun Java System Application Server Platform Edition, Sun Java Web Server, Tomcat	ľubovoľný kontajner resp. server založený na J2EE
Spôsob inštalácie	nainštaluje sa na cieľovú platformu	nasadí sa na cieľovú platformu ako webová aplikácia J2EE
Možnosť vystavenia funkčnosti	pomocou technológie Java Servlet	pomocou technológie Java Servlet a Enterprise Java Beans

Tabuľka 2.1: Rozdiely medzi produktmi JWSDP a Axis.

Pri skúmaní implementácií WS-Security, popísaných v kapitolách 2.3.3.3 a 2.3.3.4, sme získali skúsenosti s JWSDP a Axis. Na základe týchto skúseností sme prišli k záveru, že v prípade Axis sa webové služby vyvíjajú a nasadzujú o niečo jednoduchšie a zároveň tento produkt poskytuje väčšiu kontrolu nad vyvinutými

[10] Projekt GlassFish predstavuje referenčnú, *open source* implementáciu platformy J2EE 1.5. Viac informácií o tomto projekte je možné nájsť v [19].

webovými službami. Podľa našich informácií sa Apache Axis v praxi používa častejšie než Sun JWSDP.

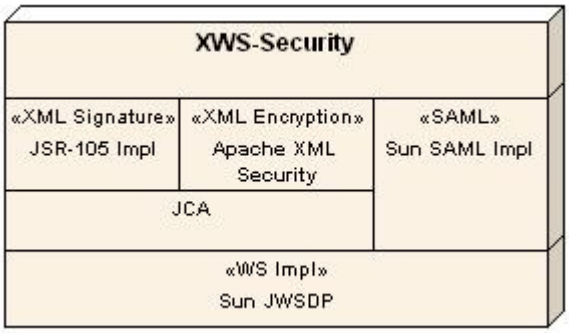
Viac informácií o týchto produktoch je možné nájsť v [20] a v [21].

2.3.3.3 XWS-Security (XML and Web Services Security)

XWS-Security je framework na vytváranie zabezpečených webových služieb vyvinutých na platforme Java. Implementuje bezpečnostný štandard WS-Security, t.j. komunikáciu SOAP je možné zabezpečiť pomocou XML podpisu, XML šifrovania a pomocou tvrdení jazyka SAML. Je súčasťou produktu Sun JWSDP. Ako bolo spomínané pri popise produktu JWSDP, implementácia webových služieb v rámci tohto produktu je postavená na technológii JAX-RPC. XWS-Security obsahuje programové rozhrania aj na zabezpečenie webových služieb, ktoré nie sú postavené na JAX-RPC. XWS-Security je možné integrovať so systémom JAAS (Java Authentication and Authorization Service), ktorý slúži na zabezpečenie aplikácií napísaných v jazyku Java.

2.3.3.3.1 Architektúra XWS-Security

Architektúra frameworku XWS-Security je znázornená na obrázku 2.6.



Obrázok 2.6: Architektúra frameworku XWS-Security. („Impl“ reprezentuje slovo implementácia a WS je skratka pre webové služby.)

Implementácia XML podpisu v XWS-Security je založená na referenčnej implementácii špecifikácie JSR-105 (Java Specification Request for XML Signature). JSR-105 špecifikuje programové rozhrania, ktoré slúži na generovanie a validáciu XML podpisu v aplikáciách napísaných v jazyku Java.

Pre potreby XML šifrovania v rámci XWS-Security sa použila časť projektu Apache XML Security, ktorá sa venuje XML šifrovaniu. V budúcnosti je snaha nahradiť túto časť referenčnou implementáciou špecifikácie JSR-106 (Java Specification Request for XML Encryption). JSR-106 predstavuje špecifikáciu programového rozhrania na prácu s XML šifrovaní v aplikáciách vyvinutých v jazyku Java.

Cieľom je, aby všetky aplikácie vyvinuté na platforme Java v budúcnosti pri práci s XML podpisom a XML šifrovaním používali rozhrania JSR-105 a JSR-106.

Samotné bezpečnostné a šifrovacie algoritmy sú implementované v rámci Java Cryptography Architecture (JCA), ktoré sú súčasťou J2SE 5.0. Pre nižšie verzie je možné doinštalovať bezpečnostný modul, ktorý poskytuje implementáciu týchto bezpečnostných algoritmov v rámci Java Cryptography Extension (JCE).

XWS-Security implementuje mechanizmus na prenos bezpečnostných tokenov v rámci správ SOAP na jednoduchú identifikáciu komunikujúcich identít. Podporované sú bezpečnostné tokeny:

- token pozostávajúci z mena a hesla (UsernameToken),
- binárny token obsahujúci certifikát X.509 (BinarySecurityToken),
- token jazyka SAML – XWS-Security obsahuje vlastnú implementáciu SAML. Sú podporované 2 typy autentitfikačných tvrdení na overenie identít:
    - **Sender-Vouches (SV)** – autorita ktorá vydala toto tvrdenie sa zaručuje za identitu komunikujúcej strany.
    - **Holder-of-Key (HOK)** – tvrdenie obsahuje kľúč, ktorý slúži na overenie identity komunikujúcej strany.

  Tieto tvrdenia boli vydané treťou stranou, ktorej dôveruje prijímajúca strana. Presný popis týchto tvrdení sa nachádza v špecifikácii zaoberajúcej sa tokenmi SAML v rámci správ SOAP. Vyžaduje sa, aby oba typy tvrdení boli podpísané XML podpisom na zaručenie integrity tokenu SAML. Viac informácii o týchto tvrdeniach je možné nájsť v [22].

2.3.3.3.2 Popis fungovania XWS-Security

XWS-Security zabezpečuje automatické vkladanie a kontrolu bezpečnostných artefaktov (XML podpis, XML šifrovanie a bezpečnostné tokeny), ktoré sa nachádzajú v správach protokolu SOAP.

JWSDP obsahuje podporný nástroj (wscompile) na vytváranie webových služieb z rozhraní jazyka Java a na generovanie klientských artefaktov. Tento nástroj obsahuje prepínač –security, pomocou ktorého sme schopní zadefinovať bezpečnosť na strane webovej služby a aj na strane jej klienta. Spôsob zabezpečenia komunikácie SOAP sa definuje v špeciálnom konfiguračnom súbore XML.

V tomto súbore je možné definovať, aké bezpečnostné artefakty musí správa SOAP obsahovať predtým, ako je odoslaná na strane klienta. Ďalej je možné pomocou tohto súboru definovať, ktoré bezpečnostné artefakty musí správa SOAP obsahovať pri volaní operácie na strane webovej služby. Stupeň zabezpečenia je možné definovať na úrovni operácie webovej služby. Napr. element `xwss:Sign` v konfiguračnom súbore špecifikuje, že odosielaná správa musí byť digitálne podpísaná. V tomto elemente je

možné špecifikovať informáciu o kľúči, ktorý sa má použiť pri overovaní tohto podpisu (napr. kľúč sa nachádza v certifikáteX.509 uloženom v hlavičke správy) a zoznam častí správy, ktoré sa majú podpísať. Ďalším príkladom je element `xwss:RequireSignature` definujúci, že prijatá správa musí obsahovať XML podpis.

Samotné podpisovanie, overovanie podpisu, šifrovanie, dešifrovanie, vytváranie a overovanie bezpečnostných tokenov automaticky zabezpečuje framework XWS-Security. XWS-Security generuje spätné volania (*callbacks*), ktoré vyjadrujú skutočnosť, že určité bezpečnostné artefakty sa našli v prijatej správe, alebo že sa majú pridať do odchádzajúcej správy. Tieto spätné volania obsluhuje trieda jazyka Java implementujúca rozhranie `CallbackHandler` uložené v balíku `javax.security.auth.callback`. Toto rozhranie je treba naimplementovať a zadefinovať v konfiguračnom súbore XML (dokumentácia k XWS-Security obsahuje príklady implementácie tejto triedy). Táto trieda dodáva XWS-Security informácie, ktoré sú potrebné pri vytváraní resp. overovaní samotných bezpečnostných artefaktov. Napr. táto trieda získa súkromný kľúč, ktorý je popísaný v konfiguračnom súbore. Tento kľúč poskytne XWS-Security na podpísanie častí odchádzajúcej správy. Ďalším príkladom je overenie bezpečnostného tokenu pozostávajúceho z mena a hesla, ktorý sa nachádza v prijatej správe. XWS-Security vygeneruje spätné volanie informujúce o tomto tokene, ktoré je zachytené a obslúžené touto triedou. Táto trieda overí, či zadané heslo prislúcha ku zadanému menu (napr. vyhľadá heslo pre zadané meno v databáze).

Ďalšou možnosťou je špecifikovať bezpečnostné artefakty komunikácie SOAP priamo v obslužnej triede. V konfiguračnom súbore sa uloží informácia, že bezpečnostné artefakty budú špecifikované pomocou obslužnej triedy za behu aplikácie. Táto možnosť sa využíva, ak je potrebné zadefinovať spôsob zabezpečenia dynamicky za behu aplikácie (napr. aplikácia si zistí, kde presne beží a na základe toho sa rozhodne, či je nutné šifrovať komunikáciu SOAP).

Ako bolo vyššie spomínané, XWS-Security sa môže použiť aj pri zabezpečení komunikácie s webovou službou, ktorá nie je implementovaná podľa JAX-RPC. V tomto prípade obsluhujúca trieda môže implementovať rozhranie `com.sun.xml.wss.SecurityEnvironment`. Konfiguračný súbor XML je pri tomto type riešení mierne upravený.

Taktiež existuje mechanizmus na integráciu XWS-Security so systémom JAAS. Príkladom je využitie tohto systému na overenie autentifikačného tokenu s využitím autentifikačných údajov, ktoré sú uložené priamo v bezpečnostných konfiguračných súboroch aplikačného servera, na ktorom je webová služba nasadená spolu s XWS-Security.

Viac informácií o XWS-Security je možné nájsť v [20].

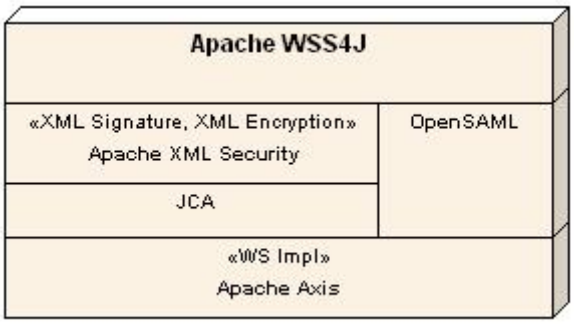
2.3.3.4 Apache WSS4J (Web Services Security For Java)

WSS4J je *open source* projekt, ktorého hlavným cieľom je zabezpečenie webových služieb implementovaných na platforme Java. Predstavuje implementáciu štandardu WS-Security. Implementuje XML podpis, XML šifrovanie a nasledovné typy bezpečnostných tokenov (rovnaké ako pri XWS-Security) :

- token pozostávajúci z mena a hesla (UsernameToken),
- binárny token obsahujúci certifikát X.509 (BinarySecurityToken),
- token jazyka SAML.

2.3.3.4.1 Architektúra WSS4J

Architektúra projektu WSS4J je znázornená na obrázku 2.7.



Obrázok 2.7: Architektúra frameworku WSS4J. („Impl“ reprezentuje slovo implementácia a WS je skratka pre webové služby.)

WSS4J je postavený na Apache Axis ako implementácii webových služieb. Pri práci s XML podpisom a XML šifrovaním využíva Apache XML Security. Na implementáciu bezpečnostných algoritmov sa používajú buď knižnice tretích strán alebo bezpečnostné knižnice J2SE 5.0. WSS4J operuje s tvrdeniami SAML pomocou OpenSAML.

2.3.3.4.2 Popis fungovania WSS4J

Ako bolo spomínané vyššie, WSS4J je založený na frameworku Apache Axis. Základom tohto frameworku sú obslužné programy technológie JAX-RPC (*handlers*), ktoré sú implementované ako triedy jazyka Java. Sú volané frameworkom Axis pri prijatí a odosielaní správy SOAP. Ich úlohou je vykonať dodatočné operácie nad týmito správami. Je možné ich zadefinovať na strane webovej služby, ale aj na strane jej klienta. Sú definované v konfiguračnom súbore XML (súbor WSDD). V ňom je možné zadefinovať aj poradie obslužných programov, v akom majú byť aplikované na tieto správy. Príkladom takéhoto obslužného programu je trieda produktu WSS4J,

ktorá pribalí do správy bezpečnostné artefakty (napr. XML podpis), pred tým ako je odoslaná produktom Axis.

WSS4J je pomocou týchto obslužných programov schopný automaticky vytvárať a vkladať (resp. vyberať a overovať) bezpečnostné artefakty do (resp. z) komunikácie SOAP. WSS4J obsahuje dve takéto obslužné programy resp. triedy. Trieda `org.apache.ws.axis.security.WSDoAllReceiver` zabezpečuje overovanie bezpečnostných artefaktov nachádzajúcich sa v prijatej správe a trieda `org.apache.ws.axis.security.WSDoAllSender` je určená na vkladanie bezpečnostných artefaktov do odchádzajúcej správy. Spolu s týmito triedami sa v konfiguračnom súbore (WSDD) špecifikujú bezpečnostné artefakty, ktorými sa má táto komunikácia zabezpečiť. Tieto dve triedy generujú volania (*callbacks*), ktoré obsluhuje trieda jazyka Java implementujúca rozhranie `CallbackHandler` uložené v balíku `javax.security.auth.callback`. Túto triedu treba naimplementovať a zadefinovať v konfiguračnom súbore (WSS4J obsahuje príklady implementácie tejto triedy). Tieto spätné volania slúžia na získavanie dodatočných informácií potrebných pri generovaní alebo validácii bezpečnostných artefaktov. Takéto spätné volanie môže byť vyvolané pri výskyte XML podpisu v správe. Táto trieda zabezpečí získanie kľúča potrebného pre validáciu tohto podpisu a poskytne ho triede `WSDoAllReceiver`. Trieda `WSDoAllReceiver` s využitím tohto kľúča automaticky overí nájdený podpis.

Bezpečnostné artefakty je možné definovať aj za behu aplikácie v triede, ktorá obsluhuje spätné volania (podobne ako pri frameworku XWS-Security).

Ako vidno, WSS4J a JWSDP fungujú na veľmi podobných princípoch.

Viac informácií o Apache WSS4J je možno nájsť v [23].

2.3.3.5 XWS-Security vs. WSS4J

XWS-Security a WSS4J zabezpečujú komunikáciu s webovými službami veľmi podobným spôsobom. Napriek tomu existujú prvky, v ktorých sa líšia. Tabuľka 2.2 znázorňuje rozdiely medzi týmito dvoma produktmi.

	XWS-Security	WSS4J
Autor	Sun Microsystems Corporation	Apache Software Foundation
Dostupnosť a cena	bezplatný prístup	open source
Implementácia webových služieb	Sun JWSDP	Apache Axis
Implementácia XML podpisu	referenčná implementácia JSR-105	Apache XML Security
Konfigurácia	pomocou špeciálneho konfiguračného súboru	pomocou konfiguračného súboru produktu Axis (WSDD)

Tabuľka 2.2: Rozdiely medzi frameworkami XWS-Security a WSS4J.

Funkčnosť XWS-Security a WSS4J sme otestovali na jednoduchých aplikáciách založených na príkladoch, ktoré boli súčasťou týchto produktov. Podľa nášho názoru, sú oba tieto frameworky vhodné na zabezpečenie webových služieb informačného systému. S WSS4J sa nám pracovalo o niečo ľahšie v porovnaní s XWS-Security najmä z dôvodu jednoduchšej manipulácie s produktom Apache Axis v porovnaní s produktom Sun JWSDP (rozdiely medzi týmito produktmi sú popísané v kapitole 2.3.3.2.2).

2.3.4 Implementácie technológie SAML

2.3.4.1 Implementácia XACML od spoločnosti Sun

Tento produkt predstavuje *open source* Java implementáciu jazyka XACML. Slúži na vytváranie a spracovanie všetkých artefaktov jazyka XACML. Programové rozhranie tejto implementácie je určené na vytváranie a spracovanie autorizačných pravidiel (*policies*) ako aj správ, ktoré slúžia na vyhodnotenie autorizačných požiadaviek. Táto implementácia bola navrhnutá tak, aby sa dala rozšíriť o novú funkčnosť. Konkrétne môže ísť o nové miesta uloženia pravidiel (napr. v adresári LDAP[11]), o nové typy atribútov v rámci žiadostí, o vyhodnotenie autorizačnej požiadavky, ktoré sa týkajú dotknutej identity (napr. atribút popisujúci, že pán Ľuboš Bisták patrí do skupiny s názvom študenti FMFI), o nové funkcie, ktoré sa používajú pri vyhodnocovaní autorizačných požiadaviek (príkladom môže byť nová funkcia, ktorá overí, či dátum

[11] LDAP (Lightweight Directory Access Protocol) – protokol na prístup k adresárom prostredníctvom protokolu TCP/IP. Tieto adresáre slúžia na uloženie rôznych údajov v stromovej štruktúre. Každý záznam takéhoto adresára obsahuje údaje vo forme množiny pomenovaných atribútov.

vytvorenia autorizačnej požiadavky sa nachádza v zadanom časovom intervale) a podobne.

Podarilo sa nám odskúšať funkčnosť tohto produktu na príkladoch, ktoré sú jeho súčasťou. Pomocou týchto príkladov sme vytvorili aplikáciu, ktorá generuje autorizačné požiadavky (Policy Enforcement Point). Tiež sa nám podarilo vytvoriť jednoduchý bezpečnostný manažér (Policy Decision Point), ktorý tieto autorizačné požiadavky vyhodnocuje. Tento manažér je možné nasadiť ako samostatný modul na ľubovoľný webový server, ktorý je založený na platforme Java.

Tento produkt bol vyvinutý vývojovými laboratóriami spoločnosti Sun Microsystems Corporation. Viac informácií možno nájsť v [24].

2.3.4.2 OpenXACML

Tento projekt by mal predstavovať ďalšiu *open source* implementáciu jazyka XACML. V dnešnej dobe je iba v procese vývoja členmi konzorcia Internet2. Viac informácií o tomto konzorciu a jeho aktivitách bolo uvedených pri popise projektu OpenSAML v kapitole 2.3.2.1.

2.3.5 Trust Service Integration Kit (TSIK)

TSIK predstavuje integračný nástroj na vytváranie a podporu dôveryhodných a bezpečných služieb. Obsahuje programové rozhrania vytvorené v jazyku Java určené pre rôzne domény zamerané hlavne na bezpečnosť, webové služby a technológiu XML. Bol vytvorený a vyvíjaný 5 rokov spoločnosťou VeriSign. V auguste 2005 bol uvoľnený pre skupinu Apache a zaradený ako „inkubačný“ projekt medzi Apache *open source* projekty. Inkubačný proces predstavuje proces transformácie existujúceho projektu na Apache *open source* projekt. Výsledkom by mal byť projekt Apache TSIK.

Produkt TSIK od spoločnosti VeriSign v dnešnej dobe obsahuje implementáciu štandardu WS-Security. Umožňuje zabezpečiť komunikáciu SOAP s využitím XML podpisu, XML šifry a pomocou tvrdení jazyka SAML. Obsahuje aj programové rozhrania na overenie týchto tvrdení. Ďalej z oblasti bezpečnosti obsahuje implementáciu technológie XKMS (XML Key Management Service) na manažovanie a prácu s kľúčmi. Konkrétne ide o rozhrania umožňujúce registrovanie klientom vygenerovaných kľúčov, mazanie, vyhľadávanie a validáciu verejných kľúčov. Súčasťou nástroja je aj programové rozhranie na vytvorenie mechanizmu dôvery medzi vzájomne komunikujúcimi aplikáciami, spočíva v možnosti overiť si kľúče, certifikáty a ostatné bezpečnostné artefakty, ktoré sa používajú v tejto komunikácii. Ďalej obsahuje implementáciu autentifikačných služieb z oblasti zdravotníctva. Obsahuje tiež vlastnú implementáciu webových služieb s podporou pre zabezpečenú komunikáciu (WS-Security), podporné programové rozhrania pre prácu s jazykom XML (rozhrania pre prácu s jazykom XPath a dokumentmi XML).

Ako bolo spomínané v predchádzajúcich kapitolách týkajúcich sa projektov skupiny Apache, v dnešnej dobe už existujú implementácie webových služieb (Apache Axis), XML šifrovania a XML podpisu (Apache XML Security) ako aj WS-Security (Apache WSS4J). Z tohto dôvodu je v procese transformácie tohto produktu na Apache TSIK snaha nahradiť príslušnú časť implementácie TSIK už existujúcimi implementáciami. Cieľom projektu Apache TSIK je vytvorenie *open source* produktu, ktorý bude implementovať nasledujúce štandardy: SAML 2.0, WS-Security, WS-Trust, WS-SecureConversation, WS-SecurityPolicy, WS-ReliableMessaging a WS-Federation. Po dokončení programových rozhraní JSR (105, 106, 155 a 183) týkajúcich sa bezpečnosti a webových služieb, je snaha ich zapracovať do tohto produktu.

Funkčnosť TSIK sme otestovali na ukážkových aplikáciách, ktoré sú súčasťou inštalácie tohto produktu.

Viac informácií o TSIK je možné nájsť v [25] a v [26].

2.4 Zhrnutie dnešného stavu v oblasti zabezpečených webových služieb

V súčasnosti existujú technológie zabezpečujúce komunikáciu s webovými službami pomocou šifrovania, digitálneho podpisu, autentifikácie a autorizácie komunikujúcich strán. Existujú tiež technológie, ktoré definujú mechanizmy nad webovými službami na realizáciu zabezpečenej komunikácie medzi aplikáciami z rôznych bezpečnostných domén. Jadrom bezpečnosti vo webových službách je technológia WS-Security. K tejto technológii sa nám podarilo nájsť a preskúmať viacero implementácií - Sun XWS-Security, Apache Axis a VeriSign/Apache TSIK. Podľa nášho názoru, každá z týchto implementácií sa môže použiť pre reálne bezpečnostné požiadavky podnikového informačného systému.

3 Podpora transakcií vo webových službách

Jednou zo základných požiadaviek kladených na aplikácie pri ich integrácii do informačného systému je schopnosť kompozície. Táto vlastnosť znamená, že tieto aplikácie sa môžu začleniť do väčšieho celku. Tieto aplikácie sú samostatné jednotky, ktoré sa môžu podieľať na dosiahnutí spoločného výsledku distribuovanej operácie. Lokálne operácie týchto aplikácií sú spravidla manažované ich lokálnymi transakčnými systémami. Tieto systémy umožňujú aplikovanie častí takejto operácie ako jeden celok. Buď sa vykoná celá operácia úspešne, alebo sa nevykoná vôbec (t.j. ani jedna jej časť). Jedným z prostriedkov integrácie aplikácií informačného systému sú webové služby. Aby sa tieto aplikácie mohli stať súčasťou väčšieho celku, vzniká potreba vytvorenia mechanizmu, ktorý by pomocou webových služieb prepojil transakčné systémy týchto aplikácií. A tak vzniká požiadavka na podporu transakcií vo webových službách.

Predstavme si registráciu nového študenta v informačnom systéme Univerzity Komenského (UK). Registrácia je úspešná, iba ak je študent úspešne zaregistrovaný v aplikácii Študent a sú mu vytvorené kurzy v aplikácii ELEA (systém pre elektronické vzdelávanie – *e-learning*). Predpokladajme, že aplikácie Študent a ELEA vystavujú svoju funkčnosť prostredníctvom webových služieb a operácie v každej z nich sú manažované lokálnym transakčným systémom. Vzniká potreba vytvorenia globálneho transakčného mechanizmu, ktorý by tieto dva transakčné systémy prepojil. Ak by jedna z týchto dvoch operácií zlyhala, tento mechanizmus by zabezpečil zrušenie druhej operácie a obe aplikácie by previedol do stavu pred začatím registrácie.

3.1 Popis transakcií

Transakcia zoskupuje množinu operácií, ktoré sa majú vykonať na dosiahnutie spoločného výsledku. Hovoríme, že tieto operácie sú vykonávané resp. bežia v rámci transakcie. Príkladom môže byť registrácia nového študenta v aplikácii Študent, pri ktorej sa informácie o tomto študentovi majú uložiť do databázy. Registrácia je realizovaná transakciou, v ktorej sa vytvorí spojenie s databázou, do databázy sa vložia údaje o študentovi a spojenie sa uzavrie. Transakcia obsahuje identifikátor, pomocou ktorého je možné určiť, ktorá operácia patrí do ktorej transakcie. Vo všeobecnosti existujú dva typy transakcií:

1. **Transakcie typu ACID** (v ďalšom texte už iba transakcie ACID) – transakcia tohto typu spĺňa nasledovné vlastnosti:

    a) je atomická (***A**tomic*) – operácie v rámci nej sa vykonávajú ako jeden celok, a ak aspoň jedna operácia zlyhá, je zrušená celá transakcia,

    b) je konzistentná (***C**onsistent*) – systém sa nachádza v konzistentnom stave pred aj po vykonaní operácií transakcie,

    c) je izolovaná (***I**solated*) – operácie transakcie neovplyvňujú operácie vykonávané v ďalších transakciách (dá sa napr. zabezpečiť zámkami v databáze),

d) má trvalý výsledok (***D**urable*) – výsledok transakcie sa uloží na trvalo, napr. je prenesený z pamäte na disk a nemal by sa už zmeniť ani pri páde databázy.

Tieto transakcie majú krátkodobý charakter. Operácie v rámci nich sa vykonávajú krátky čas, a preto je pri týchto transakciách možné aplikovať mechanizmus uzamykania údajov (napr. uzamknutia databázovej tabuľky) s ktorými pracujú tieto operácie.

Ak transakcia prebehne úspešne (t.j. všetky operácie v rámci nej sa vykonajú úspešne), všetky zmeny v nej vykonané sa potvrdia (*commit*). Príkladom potvrdenia zmien, je prenesenie zmenených údajov z pamäti na disk. Ak transakcia skončí neúspešne (t.j. aspoň jedna z operácií zlyhá), aplikácia sa prevedie do stavu, v akom bola pred začatím tejto transakcie (*rollback*).

2. **Dlhotrvajúce transakcie** – transakcia tohto typu je vykonávaná dlhší čas a nespĺňa všetky vlastnosti ACID. Spĺňa vlastnosť „všetky operácie alebo žiadna“ (t.j. je atomická). Tieto transakcie nemôžu zamknúť údaje na dlhší čas, na rozdiel od transakcií ACID, a preto sa zmeny vytvorené operáciami tejto transakcie potvrdia (*commit*) skôr, ako skončí celá transakcia. Z tohto dôvodu sa pre prípad zlyhania definujú kompenzačné mechanizmy, ktoré sa snažia priviesť aplikáciu do stavu, v ktorom bola na začiatku tejto transakcie (príkladom je vymazanie údajov v tabuľke, zaúčtovanie storno poplatkov, ...).

V distribuovanom systéme, v ktorom medzi sebou komunikuje viacero aplikácií, existuje globálna transakcia, ktorá zoskupuje a manažuje lokálne transakcie jednotlivých aplikácií. Existujú protokoly pomocou ktorých sa na základe výsledku lokálnych transakcií určí výsledok tejto globálnej transakcie. Príkladom takéhoto protokolu je dvojfázový potvrdzovací protokol (*two-phase commit protocol – 2PC*). Potvrdenie globálnej transakcie (*commit*) je realizované v dvoch fázach. Prvá slúži na zistenie či sú všetky aplikácie pripravené na aplikovanie resp. potvrdenie zmien v dôsledku vykonaných operácií v rámci lokálnej transakcie. Ak sú všetky pripravené, nasleduje druhá fáza, v ktorej sa tieto lokálne transakcie potvrdia.

Aplikácie používajú transakčné systémy na vytváranie a manažovanie transakcií. Tieto systémy zabezpečujú aj zamykanie resp. odomykanie údajov pre určitú transakciu, a tiež zabezpečujú aplikovanie kompenzačných mechanizmov pri zlyhaní transakcie. V aplikáciách vytvorených v jazyku Java sa často používajú transakčné systémy, ktoré sú založené na Java Transaction API (JTA)[12].

Pri vytváraní tejto kapitoly sme vyžili informácie z [27].

---

[12] Java Transaction API (JTA) špecifikuje štandardné rozhrania medzi manažérom transakcií a časťami distribuovaného systému medzi ktoré patria: manažér zdrojov (*resource manager*), aplikačný server a aplikácie podporujúce transakcie. Viac informácií o tejto technológii je možné nájsť v [28].

3.2 Transakcie vo webových službách

V súčasnosti existuje jediný mechanizmus na prepojenie lokálnych transakcií bežiacich v aplikáciách, ktoré vystavujú svoju funkčnosť prostredníctvom webových služieb. Tento mechanizmus predstavuje koordináciu týchto lokálnych transakcií pomocou centrálneho koordinátora. Komunikácia medzi koordinátorom a koordinovanými aplikáciami, v rámci ktorých bežia tieto lokálne transakcie, je realizovaná protokolom SOAP.

Koordinátor vytvára a riadi globálnu transakciu predstavujúcu zoskupenie lokálnych transakcií, ktoré sa vytvárajú v aplikáciách pri prístupe k ich webovým službám. Cieľom tejto globálnej transakcie je vytvorenie spoločného výsledku medzi týmito lokálnymi transakciami. Ak zlyhá jedna lokálna transakcia, globálna transakcia zabezpečí zrušenie ostatných lokálnych transakcií. Globálna transakcia je identifikovaná koordinačným kontextom, ktorý je umiestnený v hlavičke správy SOAP pri komunikácii medzi koordinátorom a koordinovanými aplikáciami, ako aj medzi koordinovanými aplikáciami navzájom. Tento kontext obsahuje identifikátor, na základe ktorého lokálna transakcia zabezpečujúca vykonanie funkčnosti vystavenej pomocou webovej služby zistí, ku ktorej globálnej transakcii patrí. Koordinačný kontext ďalej obsahuje informáciu o type globálnej transakcie (napr. transakcia ACID).

Pri koordinácii sa využívajú rôzne koordinačné protokoly pomocou ktorých koordinátor riadi lokálne transakcie. Príkladom takéhoto protokolu je dvojfázový potvrdzovací protokol. Lokálna transakcia sa stane časťou globálnej transakcie, ak ju aplikácia resp. jej transakčný manažér zaregistruje u koordinátora. Pri tejto registrácii sa špecifikuje aj koordinačný protokol, ktorým má byť táto transakcia koordinovaná.

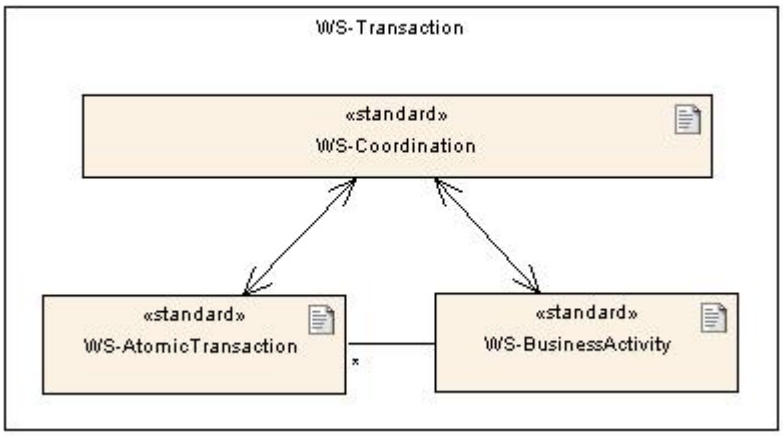
Vďaka dostatočnej abstrakcii procesu koordinácie a samotných koordinovaných aplikácií tento koordinačný mechanizmus umožňuje prepojiť pomocou protokolu SOAP rôzne transakčné systémy (napr. postavené na rôznej technologickej platforme, uložené na rôznych počítačoch v sieti a podobne), ktoré sú schopné sa prispôsobiť a spolupracovať s týmto mechanizmom za účelom dosiahnutia spoločného výsledku.

V dnešnej dobe existujú dva veľmi podobné štandardy, ktoré riešia podporu transakcií vo webových službách založené na tomto „koordinačnom“ mechanizme – WS-Transaction a WS-CAF (Web Services Composite Application Framework). Bližší popis týchto štandardov a ich porovnanie sa nachádza v nasledujúcich kapitolách.

Pri vytváraní tejto kapitoly sme čerpali informácie z nasledovných zdrojov: [29], [30] a [31].

3.2.1 WS-Transaction

WS-Transaction zoskupuje a spája tri štandardy v oblasti transakcií – WS-AtomicTransaction, WS-BusinessActivity a WS-Coordination. Vzťah týchto štandardov je znázornený na obrázku 3.1.



Obrázok 3.1: Vzťah transakčných štandardov.

Pomocou WS-Transaction je možné koordinovať dva typy transakcií – transakcie ACID a dlhotrvajúce transakcie. Transakcie ACID sú popísané štandardom WS-AtomicTransaction a dlhotrvajúce transakcie štandardom WS-BusinessActivity. Tieto štandardy definujú aj koordinačné protokoly, ktoré sa vzťahujú na nimi definované transakcie. Dlhotrvajúca transakcia sa môže skladať z množiny transakcií ACID. WS-Coordination definuje mechanizmus koordinácie transakcií pomocou protokolov, popísaných v štandardoch WS-AtomicTransaction a WS-BusinessActivity.

3.2.1.1 WS-AtomicTransaction

Tento štandard popisuje transakcie ACID a k nim prislúchajúce koordinačné protokoly. Za najdôležitejšie považujeme nasledovné dva koordinačné protokoly pre transakcie ACID:

- **Ukončovací protokol** (*completion protocol*) – týmto protokolom aplikácia začína a iniciuje ukončenie globálnej transakcie. Tiež je informovaná o výsledku tejto transakcie. V rámci tejto transakcie boli vykonané viaceré lokálne transakcie v rôznych aplikáciách, skoordinované napr. pomocou dvojfázového protokolu.

- **Dvojfázový potvrdzovací protokol** (*two-phase commit protocol – 2PC*) – popísaný v kapitole 3.1.

Viac informácií o tomto štandarde je možné nájsť v [29].

3.2.1.2 WS-BusinessActivity

Štandard WS-BusinessActivity definuje dlhotrvajúce transakcie, ktoré slúžia na realizáciu obchodných aktivít resp. procesov. Tieto aktivity sa skladajú z množiny úloh (*task*). Splnenie takejto úlohy môže zabezpečiť skupina webových služieb, ktoré sú realizované pomocou lokálnych transakcií ACID. Môže nastať prípad, kedy je potrebné zrušiť celú obchodnú aktivitu, napr. ak sa nepodarilo splniť

jednu z jej kľúčových úloh. Z tohto dôvodu je potrebné pre každú dlhotrvajúcu transakciu definovať kompenzačný mechanizmus, ktorý sa pokúsi o zrušenie zmien, ktoré boli potvrdené pri úspešnom splnení čiastkových úloh napr. potvrdením lokálnych transakcií ACID. Takéto obchodné aktivity resp. procesy sa často modelujú pomocou jazyka BPEL (viac informácií o tomto jazyku sa nachádza v kapitole 4.5).

WS-BusinessActivity definuje dva typy protokolov, ktoré definujú pravidlá ukončenia participácie aplikácie v obchodnej aktivite. Konkrétne ide o tieto dva protokoly:

- **BusinessAgreementWithParticipantCompletion** - koordinované aplikácie sa samy rozhodnú, že už nechcú participovať v obchodnej aktivite a oznámia koordinátorovi, či ich má oboznámiť o výsledku procesu. Ak chcú byť informované o výsledku procesu a tento proces zlyhal, musia byť schopné svoje potvrdené zmeny kompenzovať.

- **BusinessAgreementWithCoordinatorCompletion** – je podobný predchádzajúcemu protokolu. Rozdielom je skutočnosť, že koordinované aplikácie sa nemôžu samy rozhodnúť o ukončení ich participácie v obchodnej aktivite. Toto ukončenie musí iniciovať koordinátor.

Viac informácií o tomto štandarde je možné nájsť v [29].

3.2.1.3 WS-Coordination

WS-Coordination definuje spôsob vytvorenia globálnej transakcie (reprezentovanej koordinačným kontextom) a registrácie lokálnych transakcií u koordinátora. Definuje tiež štruktúru správ SOAP, ktoré slúžia na koordináciu transakcií pomocou protokolov, ktoré sú definované vo WS-AtomicTransaction a WS-BusinessActivity. Pre tieto účely koordinátor obsahuje tri typy webových služieb:

1. **aktivačná služba** – slúži na vytvorenie koordinačného kontextu,

2. **registračná služba** – umožňuje zaregistrovať lokálnu transakciu u koordinátora,

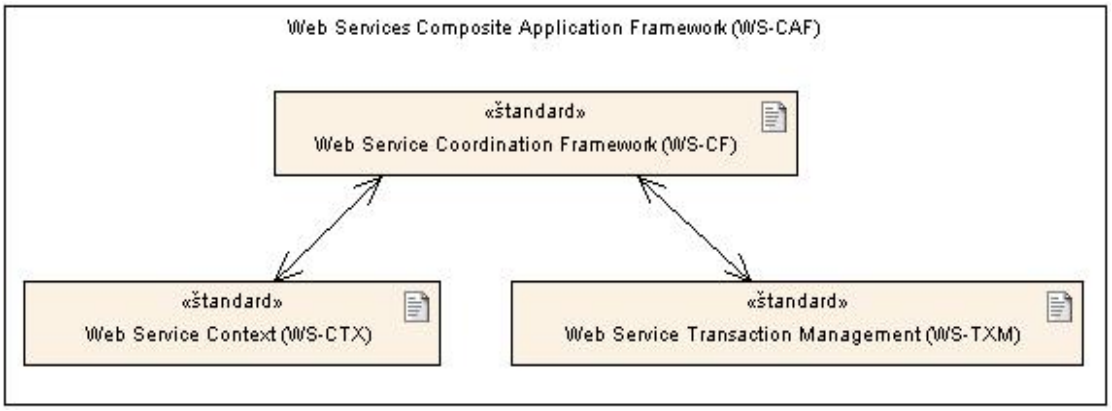
3. **koordinačná služba** – slúži na komunikáciu aplikácie resp. jej transakčného manažéra s koordinátorom.

Viac informácií o tomto štandarde je možné nájsť v [30].

3.2.2 Web Services Composite Application Framework (WS-CAF)

Štandard WS-CAF v oblasti podpory transakcií vo webových službách predstavuje alternatívu ku štandardu WS-Transaction. Jeho cieľom je vytvoriť prostredie, v ktorom by viacero kompozitných aplikácií pomocou protokolu SOAP mohlo spolupracovať na dosiahnutí spoločného výsledku. Skladá sa z troch častí (štandardov) – WS-CTX (Web Service Context), WS-CF (Web Service Coordination

Framework) a WS-TXM (Web Service Transaction Management). Ich vzájomný vzťah je znázornený na obrázku 3.2.



Obrázok 3.2: Štruktúra štandardu WS-CAF.

Štandard WS-CTX definuje koordinačný kontext. WS-CF definuje pravidlá koordinácie lokálnych transakcií participujúcich aplikácií prostredníctvom koordinátora. Štandard WS-TXM definuje množinu protokolov, ktorými koordinátor riadi tieto transakcie. WS-TXM definuje tri typy transakcií:

1. **transakcie ACID,**

2. **dlhotrvajúce transakcie,**

3. **transakcie pre obchodné procesy** – obchodný proces môže pozostávať z viacerých ACID alebo dlhotrvajúcich operácií. Tento typ globálnej transakcie predstavuje zoskupenie ACID a dlhotrvajúcich transakcií.

Viac informácií o tomto štandarde je možné nájsť v [31].

3.2.3 WS-Transaction vs. WS-CAF

Tieto dva štandardy si v úlohe podpory transakcií webových služieb navzájom konkurujú. Sú postavené na rovnakých princípoch a zabezpečujú podobnú podporu pre transakcie vo webových službách. Líšia sa pri rozdelení prvkov „koordinačného“ mechanizmu do štandardov, z ktorých sa skladajú. Rozdiely WS-Transaction a WS-CAF sú znázornené v tabuľke 3.1.

	WS-Transaction	WS-CAF	
Autor špecifikácie	Microsoft, IBM, BEA, ...	Sun Microsystems, Oracle, Arjuna Technologies, …	
Koordinačný kontext definovaný v štandarde	WS-Coordination	WS-CTX	
Koordinátor definovaný v štandarde	WS-Coordination	WS-CF	
Transakcie ACID	WS-AtomicTransaction	WS-TXM	Transakcie obchodných procesov (WS-TXM)
Dlhotrvajúce transakcie	WS-BusinessActivity	WS-TXM	

Tabuľka 3.1: Porovnanie štandardov WS-Transaction a WS-CAF.

Koordinačný kontext definovaný vo WS-CTX sa veľmi podobá koordinačnému kontextu vo WS-Transaction. Koordinačný kontext vo WS-Transaction slúži iba na identifikáciu globálnej transakcie. Vo WS-CAF slúži aj ako miesto pre zdieľanú informáciu medzi participujúcimi aplikáciami. Tieto aplikácie tu ukladajú informáciu o stave ich transakcie, ktorá je určená pre koordinátora.

Koordinátor WS-CF sa v princípe fungovania veľmi podobá na koordinátora definovaného vo WS-Transaction. Rozdiel badať v štruktúre a pomenovaní koordinačných webových služieb na strane koordinátora ako aj na strane participujúcich aplikácií.

WS-TXM sa dá prirovnať k štandardom WS-AtomicTransaction a WS-BusinessActivity. WS-TXM oproti štandardom WS-AtomicTransaction a WS-BusinessActivity definuje jemnejšie delenie typov transakcií. Koordinačné protokoly pre transakcie ACID sú veľmi podobné, ale koordinačné protokoly pre dlhotrvajúce transakcie sú odlišné.

3.3 Implementácie štandardov zaoberajúcich sa podporou transakcií vo webových službách

Podarilo sa nám nájsť a preskúmať dve implementácie štandardu WS-Transaction. Pre štandard WS-CAF sa nám nepodarilo nájsť žiadnu implementáciu.

3.3.1 Apache Kandula 1.0

Apache Kandula predstavuje *open source* implementáciu transakcií vo webových službách. Tento produkt je implementovaný v jazyku Java a jeho hlavným cieľom je koordinácia webových služieb. Tieto služby sprístupňujú funkčnosť a údaje aplikácií, ktorých transakčný systém je založený na technológii Java Transaction API (JTA). Kandula dnes implementuje štandardy WS-Coordination a WS-AtomicTransaction, a teda umožňuje koordináciu iba transakcií ACID. V súčasnosti je produkt Kandula možné aplikovať len na dva transakčné systémy, ktoré sú založené na JTA: Java Open Transaction Manager (JOTM) a transakčný systém od spoločnosti JBoss. Oba tieto transakčné systémy predstavujú *open source* implementáciu JTA (viac informácií o týchto dvoch produktoch je možné nájsť v [32] a [33]).

Ďalším cieľom Apache Kandula je jeho rozšírenie o implementáciu dlhotrvajúcich transakcií definovaných štandardom WS-BusinessActivity. Je snaha o vytvorenie podpory v tomto produkte aj pre ďalšie implementácie transakčných systémov založených na JTA.

3.3.1.1 Architektúra Apache Kandula

Apache Kandula je postavený na produkte Apache Axis ako implementácii webových služieb. Apache Kandula obsahuje obslužné programy JAX-RPC (*handlers*), ktoré sa definujú v konfiguračnom súbore produktu Axis (súbor WSDD). Tieto programy zabezpečujú manipuláciu s transakčnými artefaktmi (napr. koordinačný kontext) v správach SOAP.

Apache Kandula sa skladá z dvoch častí. Jednou je koordinátor transakcií a druhou je programové rozhranie určené pre koordinované lokálne transakčné systémy.

Koordinátora produktu Apache Kandula si môžeme predstaviť ako aplikáciu napísanú v jazyku Java, nasadenú vo webovom kontajneri J2EE. Navonok vystavuje webové služby, ktoré slúžia na vytvorenie a koordináciu globálnej transakcie. Najdôležitejšie z nich sú: aktivačná, registračná a koordinačná služba (bližší popis týchto služieb sa nachádza v kapitole 3.2.1.3).

Lokálne transakcie koordinovaných aplikácií sú manažované lokálnymi transakčnými systémami JTA. Tieto systémy komunikujú pomocou programového rozhrania (určeného pre klientskú časť) produktov Apache Axis a Apache Kandula protokolom SOAP s koordinátorom globálnej transakcie. Cieľom tejto komunikácie je iniciovanie globálnej transakcie, pripojenie sa (zaregistrovanie sa) a participovanie v tejto transakcii a aj jej ukončenie. Tieto koordinované aplikácie vystavujú dva typy služieb:

1. **participujúcu (***Participant***)** – je volaná koordinátorom a umožňuje volanie lokálneho transakčného systému (napr. koordinátor sa prostredníctvom tejto služby informuje u lokálneho transakčného systému, či je pripravený potvrdiť svoje zmeny),

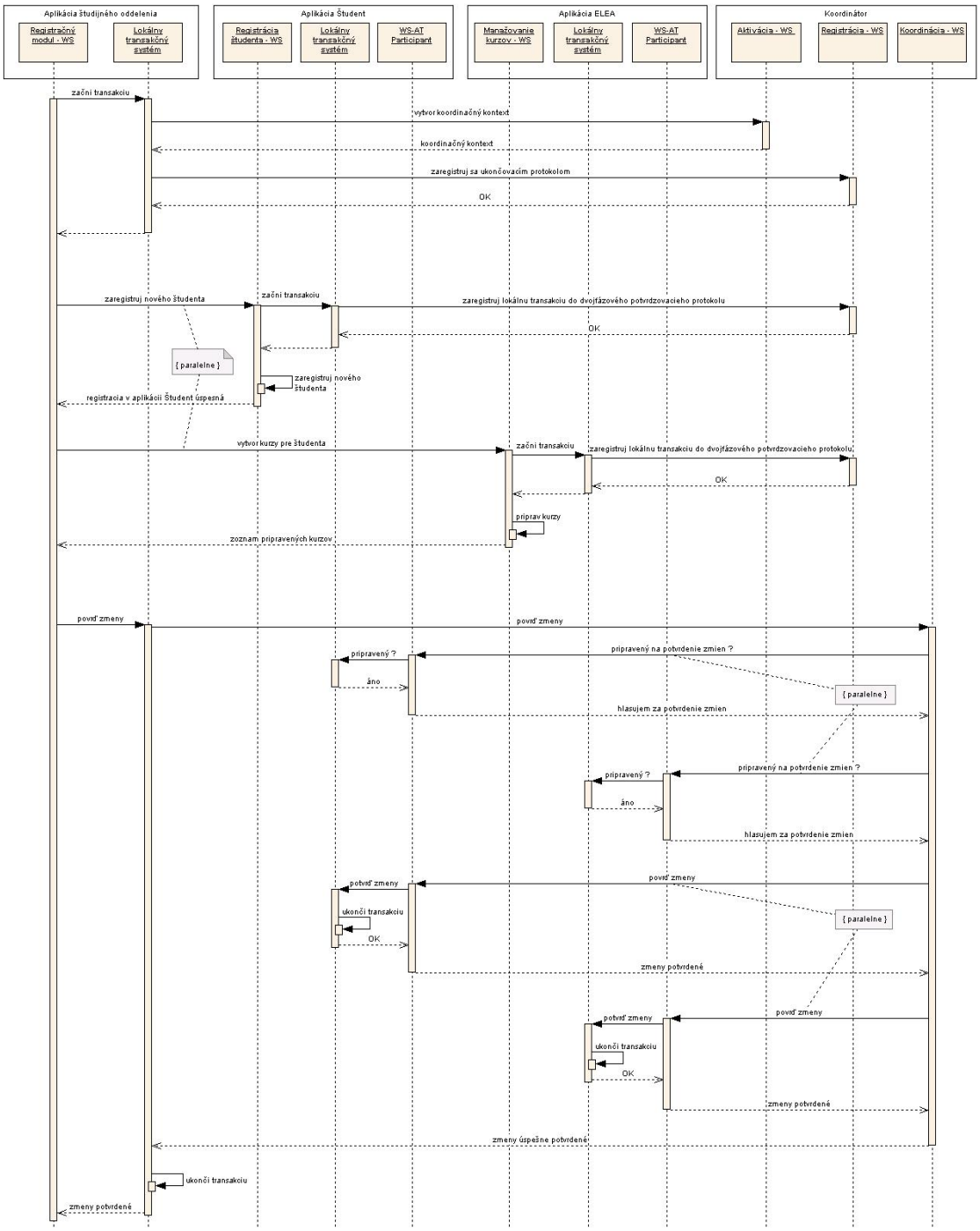
2. **aplikačné** – vystavujú funkčnosť samotnej aplikácie.

Koordinácia transakcií prostredníctvom správ SOAP si vyžaduje identifikáciu komunikujúcich strán. To znamená, že v hlavičke správy musí byť určené, kto posiela a komu je správa určená. Touto problematikou sa zaoberá štandard WS-Addressing. Produkt Kandula vyžíva *open source* implementáciu tohto štandardu v podobe produktu Apache Addressing[13].

3.3.1.2 Príklad využitia produktu Kandula v informačnom systéme Univerzity Komenského (UK)

Predpokladajme registráciu nového študenta v informačnom systéme popísanú v kapitole 3, ktorá sa skladá z úspešného zaregistrovania študenta v aplikácii Študent a z úspešného vytvorenia kurzov pre neho v aplikácii ELEA. Manažment týchto transakcií ACID je riešený pomocou produktu Apache Kandula. Tento príklad je znázornený na obrázku 3.3. Tento obrázok predstavuje sekvenčný diagram UML, ktorý popisuje komunikáciu medzi aplikáciou študijného oddelenia, aplikáciami Študent, ELEA a koordinátorom transakcií.

[13] Štandard WS-Addressing sa zaoberá adresovaním na úrovni komunikácie SOAP, ktorá sa využíva pre lepšie smerovanie (*routovanie*) správ SOAP. Viac informácií o tomto štandarde a produkte Apache Addressing je možné nájsť v [34].



Obrázok 3.3: Registrácia nového študenta v informačnom systéme UK s využitím produktu Apache Kandula.

Lokálne transakcie aplikácií Študent a ELEA sú riadené ich lokálnymi transakčnými systémami kompatibilnými s JTA. Koordinátor je samostatná aplikácia nasadená v rámci informačného systému UK. Aplikácia študijného oddelenia využíva klientskú časť programového rozhrania Apache Kandula a Apache Axis na komunikáciu s koordinátorom. Každá z aplikácií Študent a ELEA vystavuje participujúcu službu (WS-AT Participant) za účelom možnosti prístupu koordinátora k jej transakčnému systému.

Aplikácia študijného oddelenia najskôr vytvorí koordinačný kontext, ktorý neskôr vkladá do hlavičky všetkých správ SOAP pri komunikácii s aplikáciami Študent a ELEA. Všetky tri tieto aplikácie využívajú programové rozhranie produktu Apache Kandula na manipuláciu s týmto kontextom. Aplikácia študijného oddelenia iniciuje a ukončuje globálnu transakciu, a preto sa zaregistruje u koordinátora ukončovacím protokolom. Transakčné systémy aplikácií Študent a ELEA informujú koordinátor o tom, že chcú, aby ich lokálne transakcie boli koordinované dvojfázovým potvrdzovacím protokolom. Znamená to, že ich registrujú u koordinátora a súčasne špecifikujú protokol, ktorým majú byť koordinované.

Pri bližšom skúmaní a testovaní produktu Kandula sme zaznamenal niektoré jeho funkčné chyby a nedostatky. Verzia 1.0 má problémy pri rušení transakcie (t.j. pri volaní operácie *rollback*) a funguje iba s produktom Apache Axis do verzie 1.2-RC2. Napriek tomu predpokladáme, že tento produkt má perspektívu reálneho využitia pre potreby informačného systému. Samozrejme bude potrebné použitie novšej („bezchybnej“) verzie alebo zásah vývojára do kódu tohto produktu. Viac informácií o tomto produkte je možné nájsť v [35].

3.3.2 ArjunaTS 4.0

Arjuna Transaction Service (ArjunaTS) je transakčný systém implementujúci podporu distribuovaných transakcií v technológiách: CORBA[14], J2EE a webové služby. Tento komerčný produkt od spoločnosti ArjunaTS je implementovaný v jazyku Java. Transakcie pre webové služby sú v ňom implementované podľa štandardu WS-Transaction.

Tento produkt je možné nasadiť len na jednu z nasledovných platforiem pre webové služby:

- WebMethods Glue 5,
- WebLogic,
- Apache Axis 1.1 ako súčasť aplikačného servera JBoss.

Koordinácia transakcií je zabezpečená prostredníctvom webových služieb (aktivačná, registračná a koordinačná) koordinátora. Koordinátor komunikuje s koordinovanými aplikáciami (resp. s ich lokálnymi transakčnými systémami) protokolom SOAP prostredníctvom „participujúcej“ webovej služby týchto aplikácií. Všetky tieto služby sú implementované ako triedy technológie Java Servlet, ktoré vedia pracovať s protokolom SOAP. Nie sú postavené v súlade s technológiou JAX-RPC ako je to

[14] Common Object Request Broker Architecture (CORBA) je technológia od skupiny OMG, ktorá umožňuje realizovať distribuovaný systém. Zabezpečuje volanie operácií vzdialených objektov v distribuovanom prostredí.

v prípade produktov Sun JWSDP a Apache Axis. Ak sa produkt ArjunaTS nasadí na Apache Axis v rámci servera JBoss, ArjunaTS bude využíva programové rozhrania produktu Axis iba na prácu so správami SOAP. ArjunaTS nedefinuje svoje vlastné obslužné programy JAX-RPC v konfiguračnom súbore Axis (súbor WSDD).

Z pohľadu koordinovaných aplikácií sa celá komunikácia s koordinátorom javí ako volanie príslušného programového rozhrania (API) produktu ArjunaTS. Podobne to bolo riešené pri produkte Apache Kandula. Toto rozhranie transformuje danú komunikáciu na volanie príslušnej webovej služby koordinátora. Táto implementácia tiež zabezpečuje automatické vkladanie a vyberanie koordinačného kontextu do a zo správ protokolu SOAP. Na samotné využitie tohto produktu je potrebné nasadiť transakčný koordinátor na jednu z podporovaných vyššie spomenutých platforiem a vystaviť funkčnosť koordinovaných aplikácií pomocou webových služieb s využitím programového rozhrania produktu ArjunaTS. Webové služby koordinovaných aplikácií sa môžu nasadiť tiež iba na jednu z týchto podporovaných platforiem.

Súčasťou tohto produktu je ukážková aplikácia, pomocou ktorej sme otestovali transakcie ACID ako aj obchodnú aktivitu realizovanú prostredníctvom dlhotrvajúcej transakcie. Táto aplikácia predstavuje rezervačný systém, ktorý sa skladá z troch samostatných modulov. Tieto moduly slúžia na rezerváciu taxíku, lístkov do divadla a miesta v reštaurácii. Túto aplikáciu sme nasadili a otestovali na aplikačnom serveri JBoss.

Tento produkt predstavuje stabilnú a spoľahlivú implementáciu transakcií vo webových službách, a preto ho považujeme za ďalšieho vhodného kandidáta na využitie v prostredí informačného systému. Za jeho veľké negatívum považujeme obmedzenie iba na tri platformy webových služieb. V dnešnej dobe sa tento produkt transformuje na *open source* projekt spoločnosti JBoss (viac informácií o tomto procese je možné nájsť v kapitole 3.3.3). Od tejto transformácie očakávame rozšírenie produktu ArjunaTS o ďalšie podporované platformy webových služieb. Ďalší význam tejto transformácie vidíme v možnosti prepojenia resp. rozšírenia tohto produktu o implementáciu technológie, ktorá sa venuje odlišnej problematike webových služieb (napr. bezpečnosti). Viac informácií o tomto produkte je možné nájsť v [36].

3.3.3 JBoss Transaction Service (JBossTS)

V súčasnosti pracuje spoločnosť JBoss na dokončení *open source* transakčného systému, ktorý bude predstavovať implementáciu distribuovaných transakcií pre aplikácie vyvinuté na platforme J2EE a implementáciu transakcií pre webové služby. Základom tohto produktu bude produkt ArjunaTS. Ide o transformáciu ArjunaTS na *open source* projekt spoločnosti JBoss.

Viac informácií o tomto produkte je možné nájsť v [37].

3.3.4 Prepojenie WS-Transaction a WS-Security

Dnes podpora bezpečnosti a transakcií vo webových službách hrá významnú úlohu pri integrácii aplikácií informačného systému. Vzniká otázka možnosti prepojenia technológií pre tieto oblasti. Jadrom bezpečnosti vo webových službách je technológia WS-Security. WS-Transaction je jediná technológia pre podporu transakcií vo webových službách ku ktorej dnes existujú implementácie. WS-Security a WS-Ttransaction definujú artefakty (bezpečnostné resp. transakčné), ktorými sa správy SOAP majú rozšíriť. Definujú mechanizmy, ktorými sa má komunikácia SOAP zabezpečiť a vytvoriť podpora pre transakcie v tejto komunikácii.

Podľa nášho názoru, jednou z možností prepojenia týchto dvoch technológií je komunikácia SOAP, ktorá bude obsahovať súčasne bezpečnostné (napr. XML podpis) a transakčné (napr. transakčný kontext) artefakty. Klient webovej služby vloží oba typy týchto artefaktov do správ SOAP. Implementácia webových služieb vyextrahuje tieto artefakty, spracuje ich (napr. overí XML podpis, vytvorí lokálnu transakciu, ...) a zavolá príslušnú funkčnosť aplikácie. Implementácie WS-Security a WS-Transaction predstavujú iba rozšírenie protokolu SOAP, a preto sú postavené na určitej implementácii webových služieb. Z tohto dôvodu predpokladáme, že najjednoduchšie bude prepojiť produkty implementujúce WS-Security a WS-Transaction, ktoré sú postavené na rovnakej implementácii webových služieb. Samotná implementácia webových služieb v týchto produktoch zostane rovnaká a prepoja sa iba časti produktov týkajúce sa bezpečnosti a transakcií webových služieb.

Ak je implementácia webových služieb založená na JAX-RPC (bližší popis JAX-RPC je možné nájsť v kapitole 2.3.3.2.1), je možné využiť obslužné programy JAX-RPC (*handlers*) na prácu s bezpečnostnými a transakčnými artefaktmi v správach SOAP. Jedna skupina týchto programov JAX-RPC bude mať na starosti vkladanie a overovanie bezpečnostných artefaktov ako napr. XML podpis, XML šifra, autentifikačné tokeny a podobne. Druhá skupina týchto programov bude vkladať a pracovať s transakčnými artefaktmi ako napr. koordinačný kontext. Bude iniciovať operácie súvisiace s transakciami. Príkladom môže byť vytvorenie lokálnej transakcie, ktorú zaregistruje u koordinátora.

Medzi kandidátov na zabezpečenie webových služieb patria XWS-Security (Sun JWSDP), Apache WSS4J a VeriSign TSIK. Kandidáti v oblasti transakcií sú Apache Kandula a ArjunaTS. WSS4J a TSIK sú založené na rovnakej implementácii webových služieb. Z tohto dôvodu sa v ďalšej časti kapitoly obmedzíme len na možnosť prepojenia produktov XWS-Security a WSS4J s produktmi Kandula a ArjunaTS.

3.3.4.1 Apache Kandula a WS-Security

Produkt Apache Kandula je možné najlepšie prepojiť s produktom Apache WSS4J, lebo sú postavené na rovnakej implementácii webových služieb (Apache Axis). Do spoločného konfiguračného súboru produktu Axis (súbor WSDD) sa zadefinujú obslužné programy JAX-RPC oboch produktov. Problémom môže byť zvolenie verzie produktu Axis s ktorou bude vedieť spolupracovať Kandula ako aj WSS4J.

Prepojenie produktu Apache Kandula a frameworku XWS-Security môže byť veľmi zložité, lebo každý používa inú implementáciu webových služieb. Riešením môže byť modifikácia produktu Kandula tak, aby sa dal nasadiť v rámci platformy JWSDP (JWSDP je implementácia webových služieb, na ktorej je postavený framework XWS-Security). Náročnosť tohto riešenia spočíva v zmene implementácie produktu Kandula.

3.3.4.2 ArjunaTS a WS-Security

Predpokladáme, že produkt ArjunaTS je možné čiastočne prepojiť s produktom Apache WSS4J, vďaka rovnakej implementácie webových služieb. Je však možné zabezpečiť iba samotnú komunikáciu medzi koordinovanými aplikáciami (resp. ich klientmi), ale nie komunikáciu medzi koordinátorom a koordinovanými aplikáciami (resp. ich „participujúcimi“ webovými službami). Dôvodom je uzavretá resp. skrytá implementácia tejto komunikácie v produkte ArjunaTS. ArjunaTS nie je postavená na obslužných programoch JAX-RPC, ale na vlastnej implementácii pomocou tried technológie Java Servlet.

ArjunaTS a XWS-Security sa líšia v implementácii webových služieb. Možnosťou je opäť iba nasadenie koordinovaných aplikácií v rámci platformy JWSDP. Pre tieto aplikácie sa v JWSDP definujú obslužné programy, ktoré zabezpečia prácu s bezpečnostnými a transakčnými artefaktmi. Samotná komunikácia s koordinátorom sa nedá zabezpečiť prostredníctvom XWS-Security opätovne z dôvodu uzavretej resp. skrytej implementácie produktu ArjunaTS.

Tabuľka 3.2 znázorňuje zhrnutie mojich vyššie popísaných záverov.

	Apache Kandula	ArjunaTS
Apache WSS4J	- zoskupenie definícií obslužných programov v spoločnom konfiguračnom súbore produktu Axis	- zoskupenie definícií obslužných programov JAX-RPC v spoločnom konfiguračnom súbore produktu Axis - je možné zabezpečiť iba komunikáciu medzi koordinovanými aplikáciami
XWS-Security	- transformácia implementácie webových služieb produktu Kandula a koordinovaných aplikácií na platformu JWSDP	- transformácia implementácie webových služieb koordinovaných aplikácií na platformu JWSDP - je možné zabezpečiť iba komunikáciu medzi koordinovanými aplikáciami

Tabuľka 3.2: Možnosti prepojenia implementácií WS-Transaction a WS-Security.

4 Web Services Invocation Framework (WSIF)

V v procese integrácie existujúcich resp. novo vznikajúcich aplikácií do spoločného informačného systému sa dnes často používajú rôzne integračné technológie ako J2EE Connector Architecture, Java Message Service, webové služby, CORBA, Microsoft COM/DCOM a ďalšie.

Zvyčajne sa tieto technológie navzájom kombinujú, lebo každá z nich má svoju silu, ale aj slabinu v určitej aplikačnej resp. technologickej doméne. K silným stránkam webových služieb patrí napr. možnosť prepojenia systémov napísaných v rôznych programovacích jazykoch a nasadených na rôznych miestach. Príkladom ich slabej stránky je spomalenie aplikácie pri preklade SOAP správ na príslušné volania operácií aplikácie a formát XML, v ktorom sa tieto správy prenášajú. Tento formát predstavuje nadbytočnú informáciu z pohľadu obsahu prenášaných dát. Na druhej strane je dôležitý z pohľadu SOAP ako komunikačného protokolu. Príkladom spomalenia aplikácie je komunikácia SOAP, v ktorej sa vykomunikuje veľké množstvo malých správ. Ďalším negatívom webových služieb je nemožnosť štandardne prenášať transakcie medzi komunikujúcimi aplikáciami prostredníctvom protokolu SOAP (jedno z riešení je aplikovanie technológie WS-Transaction).

Technológiu WSIF môžeme vnímať ako prostriedok na prekonanie niektorých z obmedzení resp. nedostatkov webových služieb.

4.1 Popis WSIF

WSIF umožňuje pristupovať k službám bez ohľadu na to ako alebo kde sú tieto služby vystavené. Funkčnosť aplikácie je možné vyjadriť formou množiny procedúr a funkcií, ktoré táto aplikácia poskytuje okolitému svetu. Túto množinu procedúr a funkcií si môžeme predstaviť ako službu, ktorú táto aplikácia poskytuje externým aplikáciám. Hlavným cieľom WSIF je umožniť prístup k týmto službám rôznymi protokolmi (napr. cez SOAP, RMI, ...) alebo rôznymi spôsobmi (napr. priame volanie metód aplikácie napísanej v jazyku Java, prostredníctvom systému JMS, ...).

WSIF odstraňuje obmedzenie vyvíjať služby iba pre konkrétny protokol alebo prostredie. Umožňuje dynamicky zvoliť spôsob prístupu ku službám. Príkladom je aplikácia A1 vystavujúca svoju funkčnosť prostredníctvom služby, ku ktorej je možné pristupovať protokolom SOAP a RMI. RMI sa vo všeobecnosti preferuje z dôvodu vyššej rýchlosti. Aplikácia A2, ktorá chce využívať službu aplikácie A1, sa za behu rozhodne, či použije protokol SOAP alebo RMI (napr. medzi aplikáciami A1 a A2 môže byť umiestený firewall, ktorý zabraňuje A2 pristúpiť k službe A1 pomocou spojenia RMI).

WSIF sa dá využiť v procese integrácie samostatných aplikácií do spoločného informačného systému ako mechanizmus na univerzálnu komunikáciu medzi týmito aplikáciami.

Prvá implementácia tejto technológie bola vytvorená spoločnosťou IBM. Neskôr spoločnosť IBM túto implementáciu uvoľnila skupine Apache v rámci ktorej vznikol

open source projekt Apache WSIF. Apache WSIF je implementácia WSIF v jazyku Java.

4.2 Architektúra WSIF

Služby vystavené prostredníctvom WSIF sú popísané v rozšírenom dokumente WSDL. Tento dokument obsahuje popis operácií služieb a informácie potrebné na prístup k týmto službám. WSDL je rozdelený na abstraktnú a konkrétnu časť. Abstraktná časť popisuje štruktúru operácií poskytovanej služby (vstupné a výstupné parametre). Konkrétna časť obsahuje popis poskytovateľa tejto služby a informáciu o spôsobe komunikácie s týmto poskytovateľom.

WSIF rozširuje „konkrétnu" časť dokumentu WSDL o nové typy poskytovateľov služieb a o dodatočné informácie potrebné pre nadviazanie komunikácie s týmito poskytovateľmi. Príkladom takéhoto poskytovateľa je komponent EJB. Na nadviazanie spojenia s komponentom EJB je potrebné špecifikovať kontext pomocou ktorého je možné nájsť tento komponent (*JNDI naming context*), názov rozhrania, ktoré spravuje tento komponent (*home interface*), pomenovanie tohto komponentu v cieľovom kontajneri EJB (*JNDI name*), adresu a port, na ktorom počúva kontajner EJB.

Apache WSIF obsahuje priamu podporu pre nasledovné typy protokolov resp. spôsoby, pomocou ktorých je možné pristupovať k službám:

- služby sprístupnené cez protokol SOAP (webové služby),
- služby implementované ako aplikácie napísané v jazyku Java (t.j. vytvorené v rámci platfrormy Java 2 Standard Edition) – t.j. prístupné cez priame volanie aplikácie Java,
- služby implementované ako komponenty technológie Enterprise Java Beans, ktoré sú volané pomocou protokolu RMI cez IIOP,
- služby sprístupnené pomocou protokolu (resp. technológie) Java Message Service[15],
- služby sprístupnené pomocou technológie J2EE Connector Architecture (JCA)[16].

---

[15] Java Message Service (JMS) je systém na zabezpečenie spoľahlivej a asynchrónnej komunikácie medzi aplikáciami. Táto komunikácia je realizovaná výmenou správ. Viac informácií možno nájsť v [38].

[16] J2EE Connector Architecture (JCA) definuje architektúru na prepojenie platformy J2EE (aplikačný server a aplikácie postavené na platforme J2EE) s heterogénnymi podnikovými systémami (napr. aplikácia vyvinutá pre sálový počítač - *mainframe*, databázové systémy, staršie a iné aplikácie napísané

Apache WSIF je možné rozšíriť aj o nové spôsoby prístupu k službám. Toto rozšírenie v sebe zahŕňa implementáciu komunikácie s poskytovateľom tejto služby, rozšírenie dokumentu WSDL o definíciu nového poskytovateľa služby a rozšírenie implementácie produktu Apache WSIF tak, aby tento produkt vedel pracovať s takto rozšíreným dokumentom WSDL.

WSIF pri práci s dokumentmi WSDL používa produkt WSDL4J, ktorý predstavuje referenčnú implementáciu špecifikácie JSR-110. JSR-110 špecifikuje programové rozhranie určené na vytváranie a manipuláciu s dokumentmi WSDL. Autorom produktu WSDL4J je spoločnosť IBM.

Ak sa na komunikáciu použije jeden zo spôsobov založených na technológii Java, je možné využiť automatickú distribúcii transakcií pomocou Java Transaction API (JTA).

4.3 Prístup k službe pomocou WSIF

Prístup k službe pomocou WSIF je možné realizovať dvomi spôsobmi. Prvým spôsobom je vygenerovanie klientských tried (*stubs*) pre službu definovanú v dokumente WSDL. Pomocou triedy `WSIFServiceFactory` (vytvorenej podľa návrhového vzoru *factory method*) sa vytvorí abstraktná služba, predstavujúca inštanciu triedy `WSIFService`. Táto trieda nadviaže spojenie s cieľovou službou využitím časti implementácie WSIF, ktorá zabezpečuje komunikáciu s touto službou. Pomocou tejto triedy sa vytvorí inštancia vygenerovanej triedy (*stubs*), pomocou ktorej je možné pristupovať k cieľovej službe. Druhý spôsob predstavuje metóda Dynamic Invocation Interface (DII), ktorá umožňuje dynamicky volať operácie služby bez nutnosti vygenerovania klientských tried. Za behu sa vytvorí definícia volania operácie cieľovej služby pomocou programového rozhrania WSIF.

Viac o metóde DII a o klientských objektoch (*stubs*) je možné nájsť v [1].

4.4 Príklad využitia WSIF v informačnom systéme Univerzity Komenského (UK)

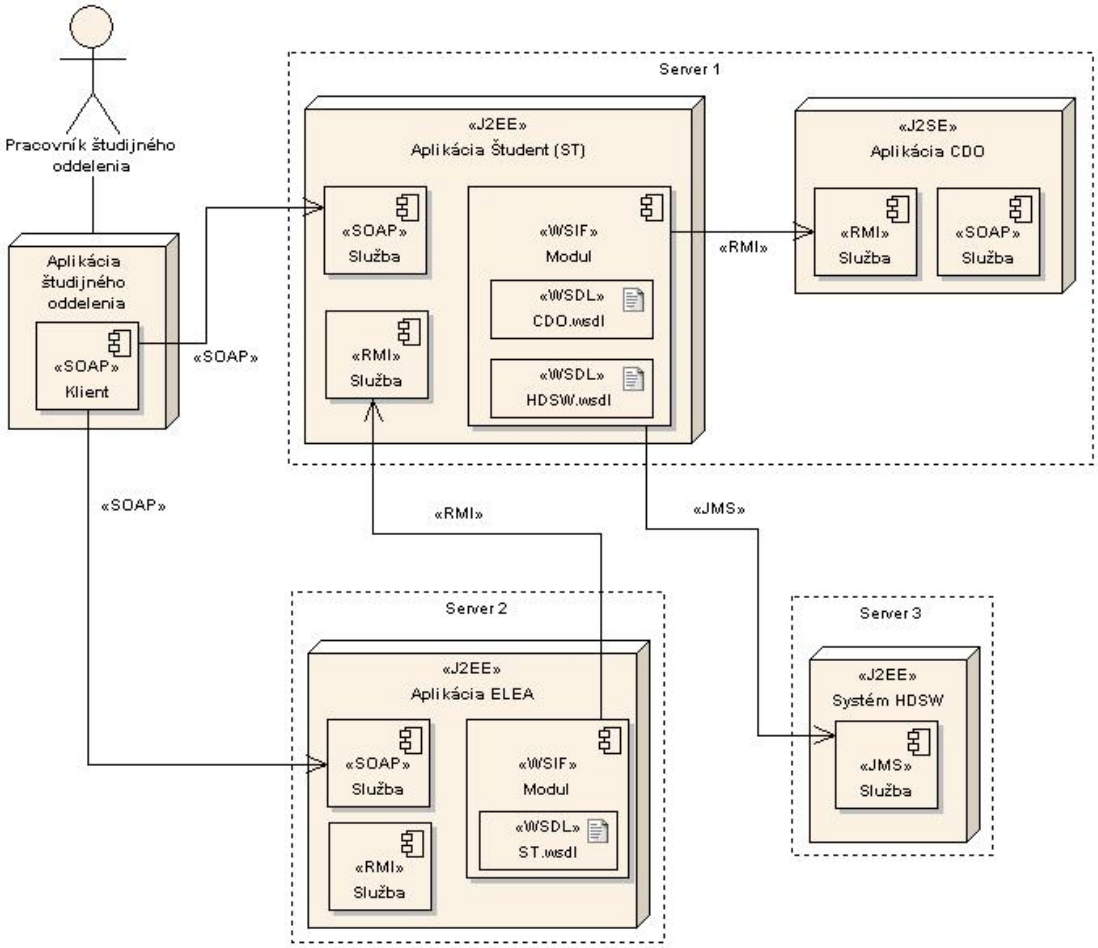
Využitie WSIF si môžeme ukázať na príklade registrácie nového študenta v informačnom systéme UK. Predpokladajme nasledovný scenár. Pracovník študijného oddelenia zadá informácie (osobné údaje, zapísané predmety, ...) o novom študentovi prostredníctvom aplikácie študijného oddelenia do aplikácie Študent (ST). Aplikácia ST najskôr požiada o vytvorenie univerzitného osobného čísla (UOČ) aplikáciu CDO, ktorá spravuje centrálnu databázu osôb UK. Na základe tohto čísla sa

---

v jazyku odlišnom od jazyka Java). Prepojenie je realizované pomocou adaptéru (*resource adapter*), ktorý sa nainštaluje na aplikačný server. Tento adaptér zabezpečuje komunikáciu medzi platformou J2EE a podnikovým (*enterprise*) systémom. Viac informácii o JCA možno nájsť v [39].

informácie o novom študentovi uložia v aplikácii ST. ST pošle tieto informácie aplikácii HDSW (HelpDesk Software), ktorá pripraví konto pre študenta na prístup do počítačových učební UK. Po úspešnom uložení informácií o študentovi v ST, pracovník študijného oddelenia pomocou aplikácie študijného oddelenia vytvorí kurzy pre nového študenta v aplikácii ELEA (systém pre elektronické vzdelávanie - *e-learning*). ELEA si najskôr zistí informácie o študentovi v aplikácii ST (napr. zapísané predmety) a na základe toho ponúkne zoznam možných kurzov pre tohto študenta.

Pri registrácii je potrebné, aby navzájom komunikovalo 5 aplikácií informačného systému UK. Predpokladajme, že aplikácie ST, HDSW, ELEA sú vyvinuté na platforme J2EE. Aplikácia CDO je vyvinutá na platforme J2SE. Aplikácie ST, ELEA a CDO vystavujú svoju funkčnosť prostredníctvom služieb ku ktorým je možné pristupovať prostredníctvom protokolu SOAP a RMI. Pripravenie konta v aplikácii HDSW nie je úplne automatický proces (je potrebné aby študent dodatočne poskytol údaje ako napr. heslo), a preto je jeho funkčnosť vystavená prostredníctvom asynchrónnej služby, s ktorou je možné komunikovať prostredníctvom JMS. Aplikácia študijného oddelenia obsahuje klienta protokolu SOAP, pomocou ktorého je možné pristupovať k službám aplikácie ST a ELEA. Popis prístupov ku jednotlivým aplikáciám je znázornený na obrázku 4.1.



Obrázok 4.1: Popis aplikácií informačného systému UK a ich vzájomnej komunikácie.

Aplikácie ST a ELEA pomocou WSIF pristupujú k službám ostatných aplikácií bez toho, aby obsahovali detaily týkajúce sa spôsobu komunikácie. Pomocou WSIF pristupujú rovnakým spôsobom k službám bez rozdielu či sa s týmito službami komunikuje protokolom SOAP, RMI alebo systémom JMS. Spôsob komunikácie s týmito službami je popísaný v príslušných dokumentoch WSDL, s ktorými operuje WSIF. Ako je vidno na obrázku 4.1, aplikácie ST a CDO sa nachádzajú na rovnakom serveri, a preto sa preferuje možnosť prístupu k CDO cez RMI. Môže sa stať, že CDO sa presunie na server umiestnený za firewallom a ST už nebude môcť pristupovať k službám CDO pomocou protokolu RMI. Bez nutnosti veľkého zásahu do aplikácie ST sa nastaví prístup k CDO pomocou protokolu SOAP. Aplikácia ST s využitím WSIF sa dá navrhnúť tak, aby sa tejto zmene prispôsobila dynamicky za behu.

4.5 WSIF v spojení s BPEL

BPEL (Business Process Execution Language) je jazyk na modelovanie obchodných procesov v rámci ktorých sa využíva funkčnosť z rôznych zdrojov (napr. rôzne existujúce podnikové aplikácie). Jazyk BPEL má formát XML. Proces namodelovaný v jazyku BPEL je vykonávaný a manažovaný špeciálnym programom (*BPEL engine*),

ktorý číta a vykonáva jednotlivé akcie toho procesu. Akcia procesu môže predstavovať zavolanie webovej služby určitej aplikácie.

BPEL4WS 1.1 (Business Process Execution Language for Web Services) predstavuje aktuálnu špecifikáciu jazyka BPEL. V dnešnej dobe sa pripravuje nová verzia s označením WS-BPEL 2.0. Tieto špecifikácie jazyka BPEL sa obmedzujú iba na funkčnosť, ktorá je vystavená prostredníctvom webových služieb. Viac informácií o WS-BPEL je možné nájsť v [40].

Obchodné procesy väčšinou využívajú funkčnosť aplikácií, ktorá je implementovaná a sprístupnená rôznymi spôsobmi. Napr. ako komponenty EJB, triedy jazyka Java, programy napísané v transakčných databázových jazykoch[17] ako Oracle PL/SQL alebo Microsoft T-SQL, prostredníctvom protokolu SOAP, technológie J2EE Connector Architecture a podobne. Nevýhodou BPEL4WS resp. WS-BPEL je, že sa obmedzujú iba na webové služby. Jedno z riešení tohto problému je použitie WSIF ako podporného nástroja jazyka BPEL.

WSIF zabezpečí prístup k funkčnosti aplikácie využívanej v obchodnom procese, ktorá je sprístupnená rôznymi spôsobmi. Prepojenie BPEL s WSIF bude realizované dokumentom WSDL, ktorý bude rozšírený o nové spôsoby prístupu k tejto funkčnosti pomocou WSIF. Ďalšou výhodou prepojenia WSIF s BPEL je automatický prenos transakcií, zvýšenie výkonu a podobne.

WSIF je v súčasnosti podporovaný viacerými produktmi BPEL postavenými na platforme J2EE. Medzi autorov týchto produktov patria spoločnosti ako IBM a Oracle. Príkladom produktu BPEL s podporou WSIF je Oracle BPEL Process Manager, ktorý spoločnosť Oracle získala akvizíciou spoločnosti Collaxa. Viac informácií o tomto produkte v spojení s WSIF je možné nájsť v [41].

4.6 Test WSIF

Funkčnosť Apache WSIF sme otestovali na jednoduchej aplikácii vytvorenej v jazyku Java (J2SE), ktorá pristupuje k štyrom testovacím službám. Túto aplikáciu sme vytvorili pomocou príkladov, ktoré sa nachádzajú v tomto produkte. Prvá služba je implementovaná v jazyku Java (J2SE), druhá ako komponent EJB, tretia je prístupná protokolom SOAP a štvrtá je asynchrónna služba, ku ktorej je prístup pomocou JMS. Všetky služby okrem prvej sme nasadili ako aplikácie J2EE na aplikačný server JBoss.

Viac informácií o WSIF je možné nájsť v [42].

---

[17] Transakčný databázový jazyk rozširuje jazyk SQL o nové prvky ako procedúry, funkcie, generátory postupnosti čísiel (*sequence*), spúšťače (*trigger*), nové údajové typy a podobne. Programy napísané v týchto jazykoch sú uložené a vykonávané priamo v prostredí databázy.

5 Klienti webových služieb

Webové služby predstavujú jednu z možností prístupu k funkčnosti a dátam informačného systému. Táto funkčnosť a dáta môžu byť určené pre ďalšie aplikácie alebo pre koncových používateľov. Z tohto dôvodu vzniká potreba existencie programov, ktoré umožnia pristupovať k týmto webovým službám. Ich úlohou je vytvoriť určitú „medzi-vrstvu“ medzi aplikáciou resp. používateľom a webovou službou. Táto „medzi-vrstva“ zabezpečí potrebnú komunikáciu SOAP. Programy tejto „medzi-vrstvy“ nazveme klientmi webových služieb.

5.1 Typy klientov webových služieb

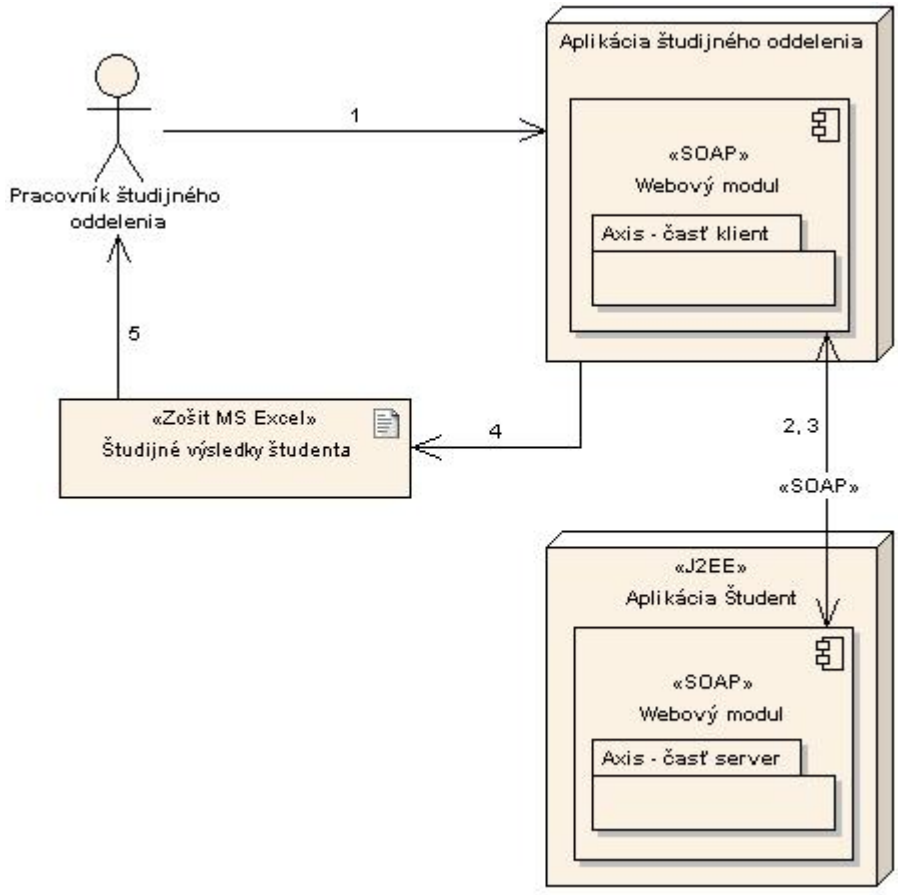
Vo všeobecnosti existujú dva typy klientov webových služieb resp. dva spôsoby prístupu k webovým službám:

1. Pomocou knižníc implementujúcich protokol SOAP. Tieto knižnice sú dodávané s produktmi implementujúcimi webové služby. Príkladom takýchto produktov je Sun JWSDP, Apache Axis, MS SOAP Toolkit a podobne. Tento spôsob je určený pre externé aplikácie.

2. Prostredníctvom klientských aplikácií určených pre koncových používateľov.

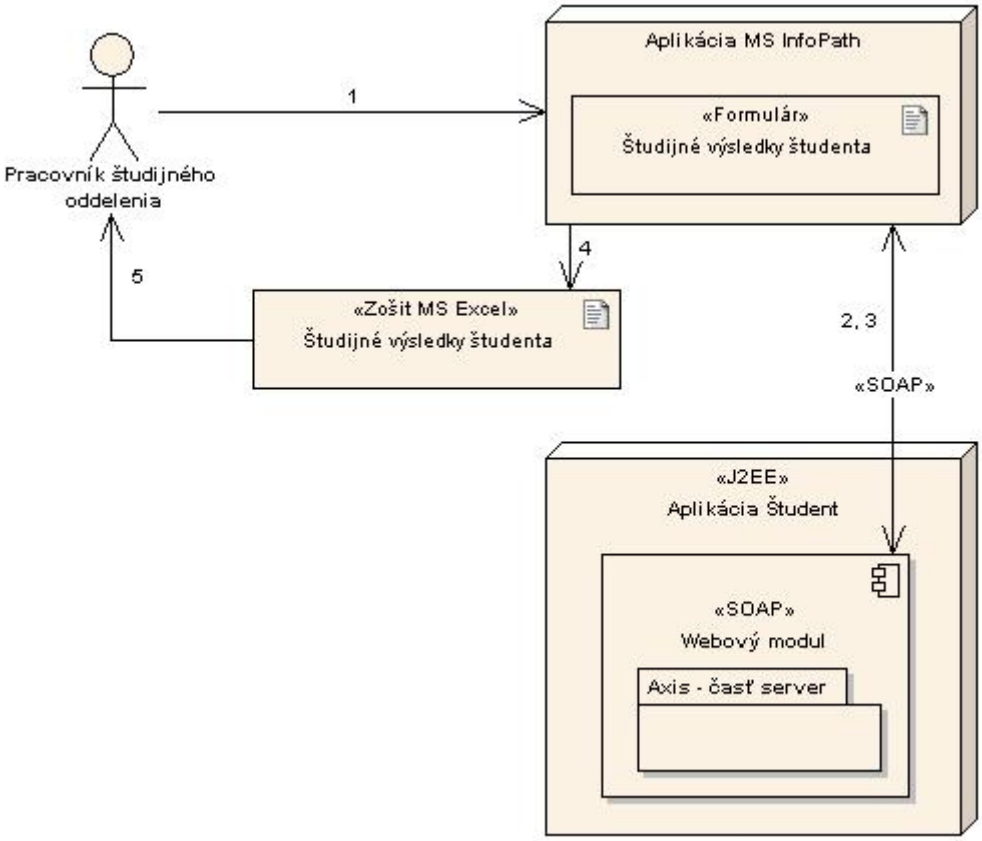
Ukážme si tieto dva spôsoby prístupu ku webovej služby na nasledovnom príklade. Predstavme si tlačovú zostavu o študijných výsledkoch študenta UK, ktorú chce vytvoriť pracovník študijného oddelenia. Údaje o študentovi potrebné na vytvorenie tejto tlačovej zostavy sa nachádzajú v aplikácii Študent. K týmto údajom sa bude dať dostať pomocou webovej služby aplikácie Študent, podobne ako v príklade uvedenom v kapitole 4.4. Tlačová zostava sa vytvorí ako zošit aplikácie MS Excel.

Jeden spôsob vytvorenia takejto zostavy je pomocou aplikácie študijného oddelenia. Táto aplikácia bude obsahovať webový modul na prístup k webovej službe aplikácie Študent. Tento spôsob je znázornený na obrázku 5.1. Toto riešenie demonštruje použitie knižníc produktu Apache Axis na vytvorenie klienta webovej služby. Ide o prvý typ klienta webovej služby.



Obrázok 5.1: Prístup k webovej službe pomocou knižníc produktu Apache Axis. (Prvý typ klienta webovej služby.)

Druhý spôsob je vytvorenie tlačovej zostavy pomocou formulára aplikácie MS InfoPath. Táto aplikácia umožňuje získať údaje pomocou webových služieb. Tento spôsob je znázornený na obrázku 5.2. Toto riešenie demonštruje prístup k webovej službe pomocou klientskej aplikácie. Ide o druhý typ klienta webovej služby.



Obrázok 5.2: Prístup k webovej službe pomocou klientskej aplikácie MS InfoPath. (Druhý typ klienta webovej služby.)

Podarilo sa nám nájsť a preskúmať viacero klientov webových služieb, ale iba jednu klientskú aplikáciu (aplikáciu druhého typu). Ide o aplikáciu MS InfoPath. Našli sme spôsob ako je možné pomocou nástroja MS SOAP Toolkit pristupovať k webovým službám z prostredia MS Office.

Aplikácia MS InfoPath a nástroj MS SOAP Toolkit sú bližšie popísané v nasledovných kapitolách.

5.2 Microsoft Office InfoPath 2003

Aplikácia MS InfoPath slúži na vytváranie rôznych formulárov (dochádzka, výkazy resp. tlačové zostavy o pracovnej činnosti a podobne) v prostredí MS Windows. Údaje pre tieto formuláre môžu byť získané z rôznych zdrojov. Príkladom je databáza MS SQL, MS Access a webové služby. Aplikácia InfoPath je súčasťou balíku MS Office 2003.

Pri návrhu formulára, ktorého dáta sa majú získavať pomocou webovej služby, sa určí umiestenie dokumentu WSDL (môže byť uložený lokálne alebo niekde na webe identifikovaný internetovou adresou - URL). Ďalej si InfoPath z tohto dokumentu automaticky zistí definíciu webovej služby a ponúkne množinu jej operácií, pomocou ktorých sa získajú dáta pre tento formulár alebo pomocou ktorých sa vypočítajú určité

položky tohto formulára. Aplikácia InfoPath vie pracovať iba s webovými službami, ktoré sú typu *document/literal*. Popis tohto typu webovej služby je možné nájsť v [1].

Ak si používateľ otvorí takto vytvorený formulár, aplikácia InfoPath sa najskôr spojí s webovou službou a získa potrebné informácie na vytvorenie obsahu formulára. Formulár môže byť navrhnutý tak, aby si používateľ mohol dynamicky počas vypĺňania tohto formulára vypočítať určité položky pomocou operácií webovej služby. V tomto prípade aplikácia InfoPath automaticky zabezpečí komunikáciu s webovou službou, tak ako pri otvorení a inicializácii formulára.

Táto aplikácia by sa dala využiť v prostredí informačného systému UK na prístup k webovým službám, ktoré by zabezpečovali vytváranie rôznych druhov tlačových zostáv o študentoch UK. Informácie o týchto študentoch by sa získavali prostredníctvom webových služieb aplikácií ako Študent a CDO.

5.3 Microsoft SOAP Toolkit

MS SOAP Toolkit je nástroj implementujúci webové služby postavené na technologickej platforme spoločnosti Microsoft. Pomocou tohto nástroja je možné vytvoriť webovú službu a klienta webovej služby v programovacích jazykoch Visual Basic, C, C++ a podobne. Tento nástroj sa dá tiež použiť na vystavenie rozhrania objektu COM prostredníctvom webovej služby, vygenerovať k nej príslušný dokument WSDL a vytvoriť klientov pre túto webovú službu.

MS SOAP Toolkit obsahuje potrebné programové rozhranie na vytvorenie klienta webovej služby a rozhranie na prácu s protokolom SOAP. Toto programové rozhranie je dodávané ako knižnica DLL v rámci tohto nástroja. Vytvoriť klienta webovej služby je možné dvoma spôsobmi. Jedným spôsobom je využiť dynamického klienta (objekt typu `SoapClient30`), ktorý sa najskôr nainicializuje pomocou dokumentu WSDL pre cieľovú webovú službu a následne je možné volať túto službu prostredníctvom tohto klienta. Druhým spôsobom je využitie programového rozhrania pre vytvorenie samotnej komunikácie SOAP. To znamená, pomocou tohto rozhrania sa vytvorí správa SOAP, odošle sa cieľovej službe a následne sa spracuje prijatá správa SOAP. Tento druhý spôsob predstavuje riešenie nižšej úrovne, ktoré umožňuje väčšiu kontrolu nad komunikáciou SOAP.

Tento nástroj je vhodné použiť na vytvorenie klientských aplikácií v prostredí Microsoft Windows, taktiež je vhodné ho využiť v spojení s ostatnými produktmi spoločnosti Microsoft, ktoré chcú pristupovať k dátam a funkčnosti iných aplikácií vystavených cez webové služby. Príkladom môže byť vytvorenie makra (napísanom v jazyku Visual Basic for Applications) v programe Microsoft Excel, ktoré bude predstavovať klienta webovej služby. Takéto makro na prístup k webovej službe z prostredia MS Excel sa použilo v aplikácii VIKUK-WS. Viac informácií o tejto aplikácii a jej ukážkovom klientovi je možné nájsť v kapitole 6.

Dnes spoločnosť Microsoft odporúča tento nástroj nahradiť nástrojmi a knižnicami dodávanými v rámci platformy .NET (t.j. prejsť aj pri vývoji webových služieb na platformu .NET).

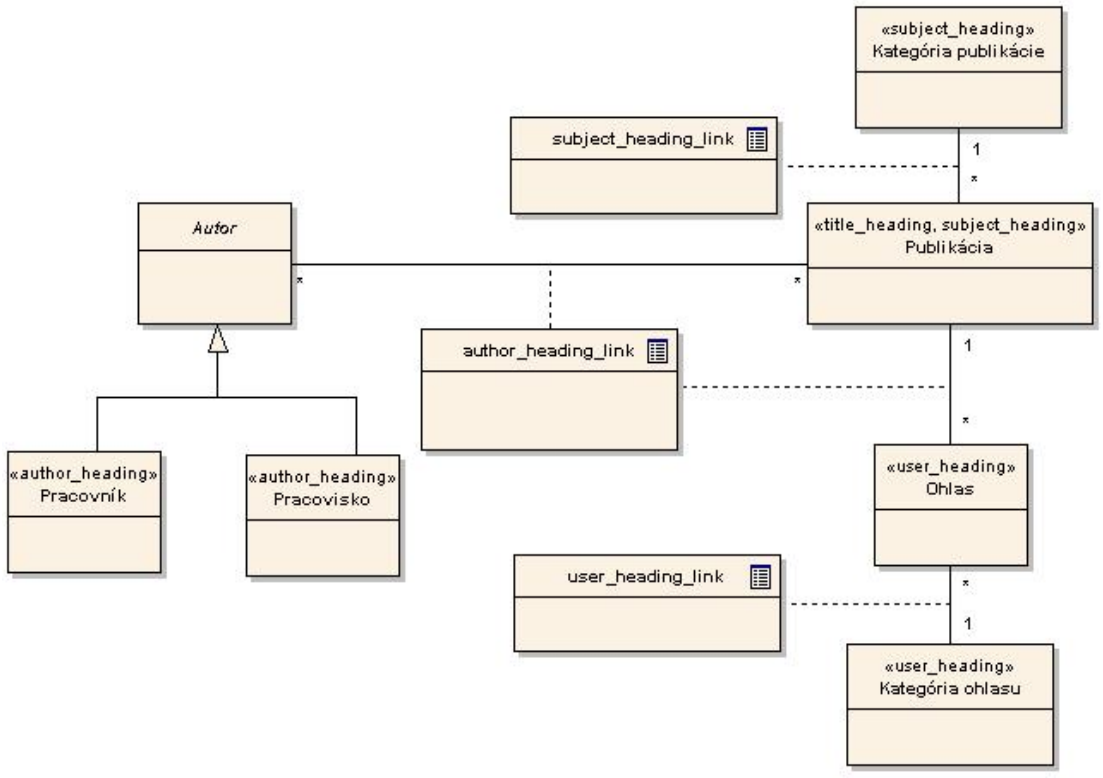
Viac informácií o tomto nástroji je možné nájsť v [43].

6 Ukážková aplikácia VIKUK-WS

V rámci diplomovej práce sme vytvorili ukážkovú aplikáciu VIKUK-WS, ktorá demonštruje klientský prístup k dátam aplikácie Virtuálna knižnica Univerzity Komenského (VIKUK) prostredníctvom webovej služby. VIKUK poskytuje služby v oblasti automatizácie knižničných činností vrátane uchovávania informácií o publikačnej činnosti pracovníkov UK. Aplikácia VIKUK je realizovaná na základe softvérového produktu VIRTUA, ktorý vytvorila americká spoločnosť VTLS.

6.1 Súčasný stav aplikácie VIKUK

Aplikácia VIKUK v rámci publikačnej činnosti eviduje informácie o publikáciách a ohlasoch pracovníkov a pracovísk UK (väčšinu týchto pracovísk tvoria katedry jednotlivých fakúlt UK). Na obrázku 6.1 je znázornený doménový model aplikácie VIKUK vzťahujúci sa na publikačnú činnosť v rámci UK, ktorý sa nám podarilo identifikovať z jej dátového modelu. Tento model je vyjadrený pomocou diagramu tried UML.



Obrázok 6.1: Doménový model aplikácie VIRTUA reprezentujúci publikačnú činnosť UK. Abstraktné entity sú znázornené kurzívou (*italic*).

Každá entita predstavuje skupinu informácií, ktorá je uložená v určitých databázových tabuľkách. Tieto tabuľky sú vyjadrené v diagrame ako stereotypy jednotlivých entít. Relácie medzi entitami sú vyjadrené pomocou asociačných tried, ktoré predstavujú databázové tabuľky.

Entita `Autor` reprezentuje informácie o autoroch publikácií. Ak pracovník vydá publikáciu na pracovisku UK, tak aj toto pracovisko je v aplikácii VIKUK evidované ako jeden z autorov tejto publikácie. Entita `Publikácia` zoskupuje informácie týkajúce sa konkrétnej publikácie ako jednoznačný identifikátor publikácie v aplikácii VIKUK, názov publikácie, informácie o autorovi resp. skupine autorov, rok vydania a podobne. Každá publikácia patrí do určitej kategórie. Informácie o kategóriách publikácií reprezentuje entita `Kategória publikácie`. Entita `Ohlas` predstavuje informácie o ohlasoch publikácií (t.j. citáciách). Príkladom takejto informácie je jednoznačný identifikátor ohlasu v aplikácii VIKUK, identifikácia publikácie, na ktorú sa ohlas vzťahuje, rok ohlasu a podobne. Rovnako ako pri kategóriách, každý ohlas patrí do určitej kategórie. Informácie o kategóriách ohlasov sú zoskupené v entite `Kategória ohlasu`.

V súčasnosti aplikácia VIKUK obsahuje webové rozhranie na prístup k informáciám o publikačnej činnosti UK. Neexistuje však spôsob, ako by sa dali tieto informácie poskytnúť externým aplikáciám na ďalšie použitie. Príkladom takéhoto ďalšieho použitia je automatické generovanie tlačových zostáv o publikačnej činnosti zamestnancov a pracovísk UK. V dnešnej dobe tieto zostavy robia ručne (resp. poloautomaticky) správcovia aplikácie VIKUK.

Jedným z riešení tohto problému je vystavenie údajov aplikácie VIKUK prostredníctvom webovej služby. Pomocou tejto služby by sa umožnilo externým aplikáciám získať tieto údaje. Z týchto údajov by mohli automaticky generovať tlačové zostavy o publikačnej činnosti na UK. Presne takéto riešenie predstavuje aplikácia VIKUK-WS.

Aplikácia VIKUK-WS poskytuje informácie o publikačnej činnosti jednotlivých pracovníkoch a pracovísk UK prostredníctvom webovej služby. Hlavným cieľom aplikácie VIKUK-WS je ukázať využitie webových služieb pri riešení reálnych potrieb informačného systému. Na tejto aplikácii je možné vidieť prepojenie dát staršej (*legacy*) aplikácie s ľubovoľnou aplikáciou pomocou webových služieb bez obmedzenia na technologickú platformu, ktorá sa použila na ich implementáciu.

6.2 Popis aplikácie VIKUK-WS

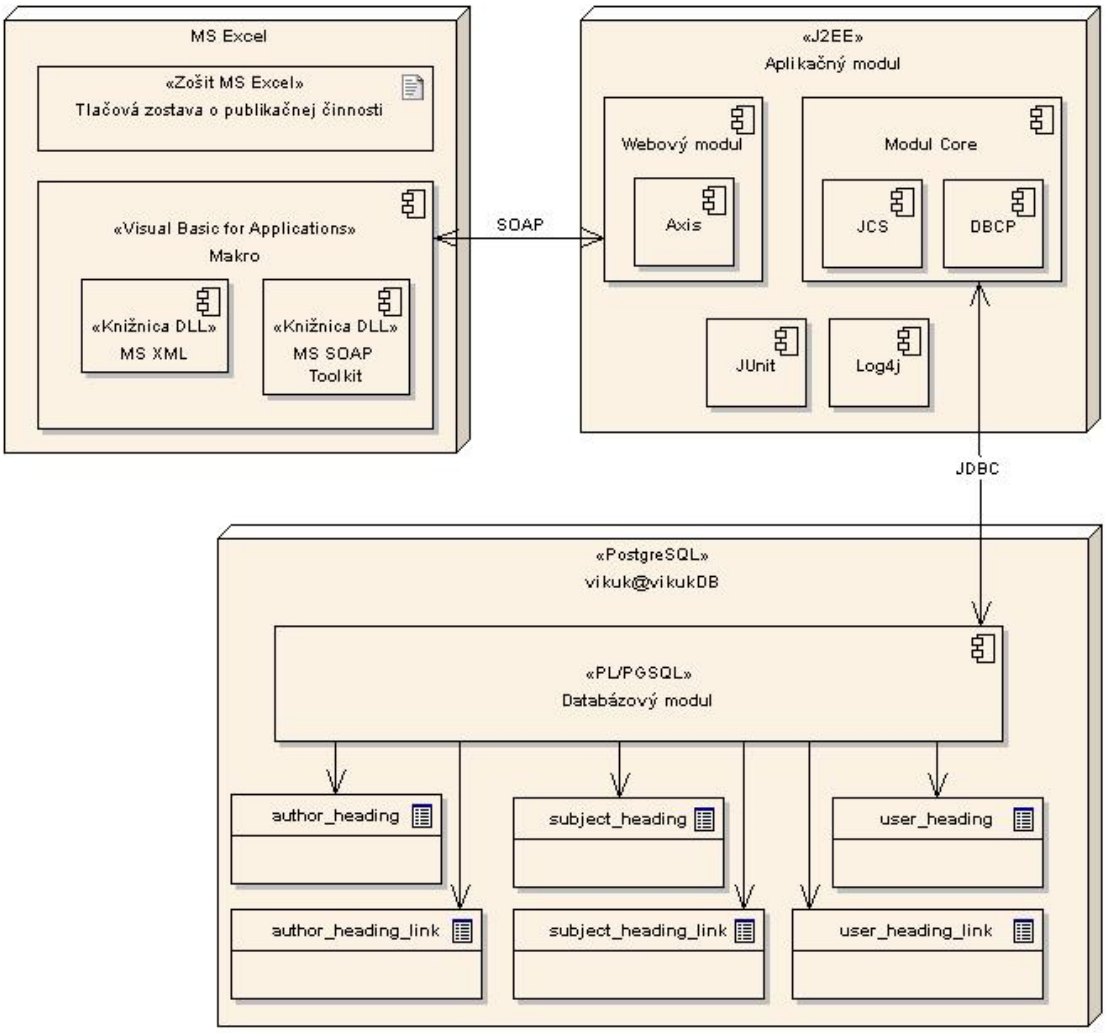
Dáta aplikácie VIKUK sú uložené v databáze Oracle 8i. Z dôvodov, že databáza Oracle 8i je komerčný produkt a zároveň jej hardvérové požiadavky presahujú naše možnosti, a taktiež z dôvodu reálnosti prezentácie tejto ukážkovej aplikácie, rozhodli sme sa pre migráciu časti dát do databázy PostgreSQL 8.1. Premigrovali sme len tie dáta, ktoré sú potrebné na získanie počtu publikácií a ohlasov pracovníkov a pracovísk UK. Pre databázu PostgreSQL sme sa rozhodli preto, že ide o stabilný *open source* produkt, a že sa podobá databáze Oracle. Obsahuje podobnú štruktúru uloženia dát, podobnú funkčnosť v podobe generátorov postupnosti čísiel (*sequences*), spúšťačov (*triggers*), vlastných typov a podobne. Jeden z podporovaných transakčných jazykov PostgreSQL je PL/PGSQL, ktorý sa veľmi podobá transakčnému jazyku PL/SQL od spoločnosti Oracle.

V súčasnosti nemáme k dispozícii štandardné programové rozhranie pre prístup k dátam aplikácie VIKUK, a preto bolo potrebné implementovať takéto rozhranie v aplikácii VIKUK-WS. Ak by implementácia takéhoto rozhrania existovala, stačilo by túto implementáciu vystaviť prostredníctvom webovej služby.

Samotné získavanie informácií o pracovníkoch a katedrách, počte ich publikácií a ohlasov, sme sa rozhodli implementovať prostredníctvom funkcií PL/PGSQL. Pre túto možnosť sme sa rozhodli kvôli tomu, že získavanie týchto informácií je dosť časovo náročná operácia zapríčinená zložitým dátovým modelom aplikácie VIKUK. Ďalším dôvodom je možnosť priamočiarej transformácie týchto funkcií na funkcie jazyka PL/SQL, bez nutnosti veľkého zásahu do ostatných vrstiev aplikácie VIKUK-WS.

6.2.1 Architektúra aplikácie

Obrázok 6.2 znázorňuje architektúru aplikácie VIKUK-WS.



Obrázok 6.2: Architektúra aplikácie VIKUK-WS.

Aplikácia VIKUK-WS sa skladá z troch základných častí:

1. **Databázový modul** – množina funkcií vytvorených v jazyku PL/PGSQL (ďalej iba funkcie PL/PGSQL) operujúcich nad dátami aplikácie VIKUK. Obsahuje funkcie na získanie informácií o autorovi publikácie, pracovisku, kde daná publikácia vznikla a funkcie na získanie počtu publikácií a ohlasov pre príslušného autora alebo pracovisko. Prvé dve funkcie sú potrebné na získanie jednoznačného identifikátora autora alebo pracoviska, ktorý sa využíva v ďalších dvoch funkciách. Podarilo sa nám zrýchliť výpočet počtu publikácií a ohlasov z cca 52 sekúnd na cca 1 sekundu pomocou optimalizácie príkazov jazyka SQL.

2. **Aplikačný modul** – aplikácia postavená na platforme J2EE, ktorá sprístupňuje funkčnosť vytvorenú v jazyku PL/PGSQL prostredníctvom webovej služby. Realizuje tiež konverziu medzi objektovým a relačným svetom. Tento modul sa skladá sa z dvoch samostatných častí:

    a. **Modul Core** – aplikácia napísaná v jazyku Java, ktorá zabezpečuje volanie funkcií PL/PGSQL prostredníctvom rozhrania JDBC[18]. Tento modul je navrhnutý tak, aby ho bolo možné jednoducho rozšíriť o ďalší typ dátového zdroja. Pri transformácii aplikácie do ostrej prevádzky sa tento modul jednoducho rozšíri o platformu Oracle, t.j. o volanie funkcií napísaných v jazyku PL/SQL. Na volanie funkcií PL/PGSQL sme použili spoločnú množinu databázových spojení (*connection pooling*), ktoré sa vytvoria až pri prvom prístupe do databázy. Ďalej sa udržiavajú otvorené a zdieľajú sa medzi rôznymi požiadavkami na prístup do databázy. Pri implementácii tohto mechanizmu sme využili *open source* produkt Apache Jakarta Commons DBCP[19], ktorý rieši problematiku zdieľania databázových spojení. Tento mechanizmus zvýšil rýchlosť volania funkcií PL/PGSQL.

    b. **Webový modul** – sprístupňuje funkčnosť modulu Core prostredníctvom webovej služby. Samotná webová služba bola postavená na projekte Apache Axis verzie 1.3. Túto implementáciu webových služieb sme zvolili z dôvodu, že Axis je stabilný a spoľahlivý produkt pomocou ktorého sa jednoducho vyvíjajú a manažujú webové služby. Je možné ho nasadiť na ľubovoľný webový kontajner J2EE bez potreby dodatočnej konfigurácie.

    Aplikačný modul je nasadený na *open source* webovom kontajneri J2EE Apache Tomcat. Apache Tomcat je veľmi rozšírený, jednoduchý a stabilný produkt.

---

[18] JDBC (Java Database Connectivity) je programové rozhranie pre prístup k relačným databázam v jazyku Java.

[19] Apache Jakarta Commons DBCP (Database Connection Pooling) – viac informácií o tomto projekte je možné nájsť v [44].

Zrýchlenie aplikácie na úrovni tohto modulu sa nám podarilo realizovať kešovaním (*caching*) požiadaviek na vykonanie funkcií PL/PGSQL. Popis takejto požiadavky sme reprezentovali triedou jazyka Java, ktorej inštancia sa spolu s výsledkom vykonanej funkcie PL/PGSQL ukladá do lokálnej keši. Táto keš je súčasťou modulu Core. Pri každom vybavovaní požiadavky na vykonanie funkcie PL/PGSQL, modul Core najskôr vyhľadá túto požiadavku v keši. Ak sa nenájde takáto požiadavka v keši, tak sa vykoná funkcia PL/PGSQL a výsledok spolu s popisom požiadavky sa uloží do tejto keši. Ak bola takáto požiadavka už niekedy obslúžená, nachádza sa v keši a výsledok volania funkcie PL/PGSQL reprezentujúci túto požiadavku sa získa veľmi rýchlo, bez nutnosti znovu vykonania tejto funkcie v databáze. Tento mechanizmus kešovania je znázornený na obrázku 6.5. Ide o veľmi rýchle a efektívne vybavenie opakujúcich sa volaní funkcií PL/PGSQL, ktorých vykonanie trvá dlhší čas z dôvodu zložitého dátového modelu a veľkého množstva dát v aplikácii VIKUK.

Pri implementácii kešovania sme využili *open source* produkt Apacha Jakarta Java Caching System[20]. Tento produkt je v aplikáciách napísaných v jazyku Java veľmi uznávaný. Obsahuje širokú množinu podporovaných vlastností. Príkladom týchto vlastností sú rôzne spôsoby ukladania kešovaných dát, vzdialená synchronizácia, možnosť nastavenia expirácie (vypršanie platnosti) dát, podpora prostredia klaster[21] (*cluster*) a podobne.

Významné činnosti aplikačného modulu sú zapísané (*logged*) pomocou produktu Apache Log4j do špeciálneho súboru vytvorenom vo webovom kontajneri Apache Tomcat. Príkladom takejto významnej činnosti je zlyhanie spojenia s databázou. V tomto prípade je vytvorený záznam v tomto súbore, ktorý obsahuje dátum a čas zlyhania spojenia, a aj chybovú správu ovládača JDBC. Obsahuje tiež informáciu o funkcii PL/PGSQL, ktorá sa mala vykonať.

Na testovanie funkčnosti aplikačného modulu sme vytvorili a implementovali testovacie scenáre (*test cases*) pomocou produktu JUnit. Časťou týchto testovacích scenárov je webový klient, ktorý umožňuje otestovať webovú službu aplikácie VIKUK-WS.

Apache Log4j a JUnit sú *open source* produkty implementované v jazyku Java. Pre tieto produkty sme sa rozhodli preto, že sa v súčasnosti najčastejšie používajú v aplikáciách vytvorených v jazyku Java pre potreby vytvorenia záznamov o behu týchto aplikácií, a pre potreby vytvorenia testovacích scenárov pre tieto aplikácie.

---

[20] Apacha Jakarta Java Caching System (JCS) – viac informácií o tomto projekte je možné nájsť v [45].

[21] Klaster (*cluster*) – množina počítačov v sieti, na ktorých je nasadená rovnaká aplikácia. Pri požiadavke na vykonanie funkčnosti tejto aplikácie je táto požiadavka vybavená na najmenej zaťaženom počítači.

3. **Ukážkový klient** – k funkčnosti aplikácie VIKUK-WS je možné pristupovať z akejkoľvek externej aplikácie, ktorá vie pracovať s protokolom SOAP. Tieto aplikácie je potrebné upraviť podľa súboru WSDL aplikácie VIKUK-WS (zväčša sa jedná o automatický proces, napr. o automatické generovanie klientských programov pomocou nástrojov ako axis-wsdl2java, jwsdp-wscompile, MS SOAP Toolkit, ...).

   V rámci aplikácie VIKUK-WS sme vytvorili ukážkového klienta pre prístup k webovej službe aplikácie VIKUK-WS. Ukážkový klient sa skladá z dvoch častí:

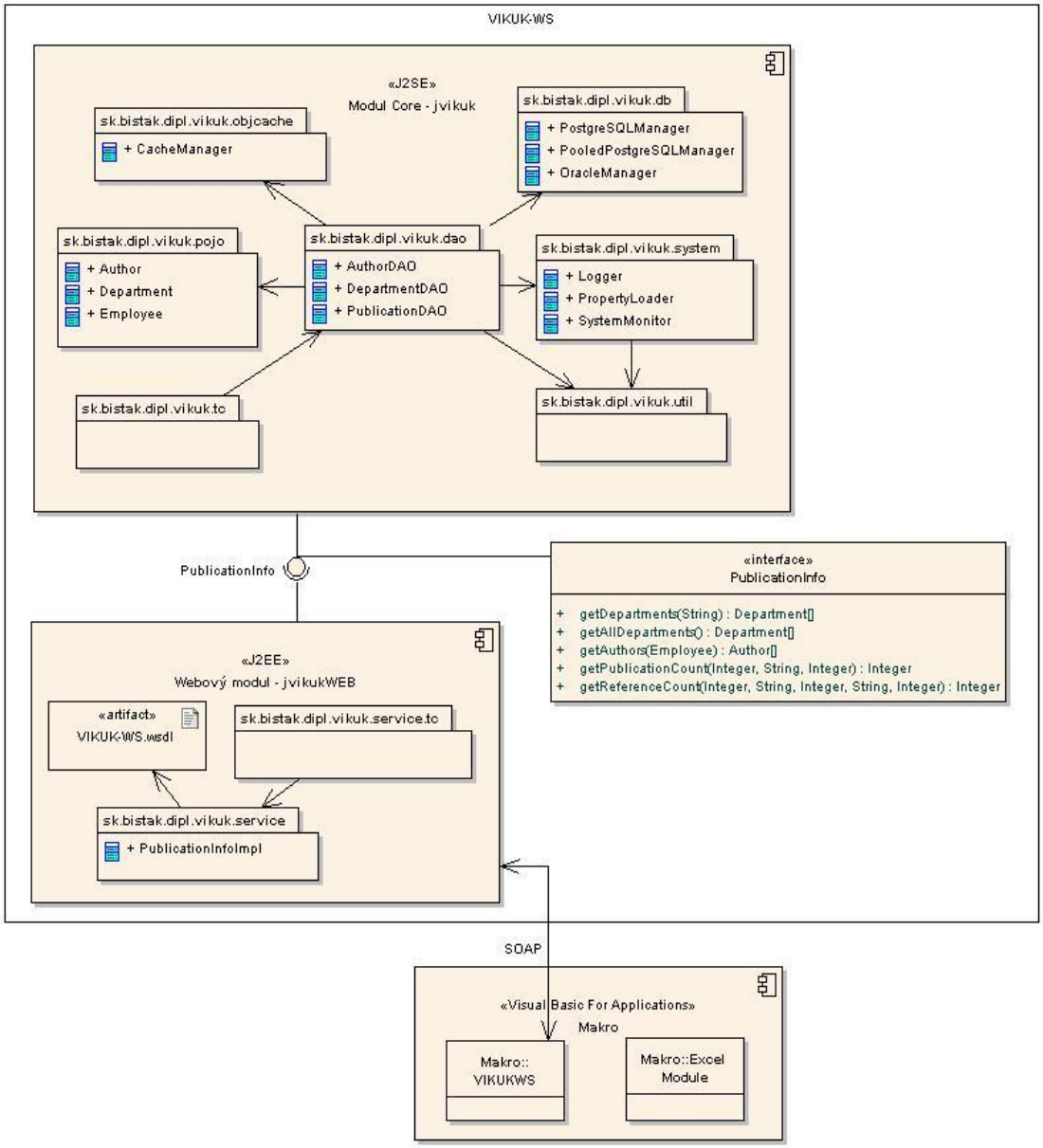
   a. **Makro** – predstavuje základný prvok ukážkového klienta. Je vytvorené v jazyku Visual Basic for Applications a realizuje komunikáciu s webovou službou aplikačného modulu pomocou knižnice MS SOAP Toolkit verzie 3.0. Knižnica MS SOAP Toolkit predstavuje najjednoduchší prístup k webovým službám z prostredia MS Excel. Toto makro predstavuje samostatný komponent, ktorý je možné znovu použiť v ďalších aplikáciách pre prístup k informáciám o publikačnej činnosti UK.

   b. **Zošit MS Excel** - predstavuje výkaz o publikačnej činnosti niektorých pracovísk UK a o pracovníkoch jednej katedry UK. Tieto informácie sa získavajú volaním webovej služby aplikačného modulu pomocou makra popísaného v bode a. Týmto zošitom je znázornené reálne využitie aplikácie VIKUK-WS za účelom vytvorenia automatických tlačových zostáv o publikačnej činnosti pracovníkov a pracovísk UK. Demonštruje tiež spôsob prístupu k webovej službe aplikačného modulu s využitím makra.

   Viac informácií o klientoch webových služieb je možné nájsť v kapitole 5.

6.2.2 Detailný popis štruktúry modulov

Obrázok 6.3 znázorňuje umiestnenie balíkov, rozhraní a tried vytvorených v jazyku Java v jednotlivých moduloch aplikácie VIKUK-WS. Znázorňuje jednotlivé časti makra predstavujúceho klienta webovej služby. Na tomto obrázku sa nachádzajú iba najvýznamnejšie balíky, triedy a rozhrania.



Obrázok 6.3: Detailný popis štruktúry modulov aplikácie VIKUK-WS.

Technický názov modulu Core je jvikuk a obsahuje nasledovné balíky:

- `sk.bistak.dipl.vikuk.pojo` – obsahuje triedy `Author`, `Department` a `Employee`, ktoré reprezentujú autora publikácie, pracovisko a pracovníka UK. (POJO = Plain Old Java Object)

- `sk.bistak.dipl.vikuk.db` – obsahuje triedy na manažment spojení JDBC s rôznymi typmi databáz. Príkladom takýchto tried sú triedy `PostgreSQLManager`, `PooledPostgreSQLManager` a `OracleManeger`. Trieda `PooledPostgreSQLManager` manažuje zdieľanú množinu spojení JDBC (*connection pooling*) s databázou PostgreSQL.

- `sk.bistak.dipl.vikuk.objcache` - obsahuje triedy reprezentujúce požiadavky na vykonanie funkcií PL/PGSQL a triedu `CacheManager`, ktorá manažuje lokálnu keš.

- `sk.bistak.dipl.vikuk.system` - obsahuje triedy súvisiace s behom, správou a konfiguráciou modulu Core. Trieda `Logger` zabezpečuje zapisovanie významných udalostí o behu aplikácie do špeciálneho súboru (*logging*). Trieda `PropertyLoader` slúži na manipuláciu s konfiguračným súborom jvikuk.properties. Trieda `SystemMonitor` monitoruje stav systému, na ktorom beží aplikácia (napr. zisťuje informácie o veľkosti voľnej pamäti).

- `sk.bistak.dipl.vikuk.dao` - základný balík modulu, ktorého triedy sú postavené na návrhovom vzore Data Access Object (DAO). Tieto triedy implementujú metódy na získavanie informácií o publikačnej činnosti UK. Môžu byť rozšírené o nové typy dátových zdrojov v ktorých sú uložené informácie o publikačnej činnosti UK (napr. o databázu Oracle). Viac informácií o vzore DAO je možné nájsť v [46].

- `sk.bistak.dipl.vikuk.util` - obsahuje podporné nástroje pre tento modul.

- `sk.bistak.dipl.vikuk.tc` - obsahuje implementáciu testovacích scenárov pre tento modul.

Modul Core navonok poskytuje tri rozhrania DAO ktoré sú zoskupené v rozhraní `PublicationInfo`. Toto rozhranie je vystavené webovým modulom prostredníctvom webovej služby pomocou produktu Apache Axis. Technický názov webového modulu je jvikukWEB. Základ webového modulu tvoria nasledovné balíky:

- `sk.bistak.dipl.vikuk.service` – najvýznamnejšia trieda v tomto balíku je `PublicationInfoImpl`, ktorá implementuje rozhranie `PublicationInfo` pomocou modulu Core. Metódy tejto triedy sú implementované ako delegácia metód tried DAO modulu Core.

- `sk.bistak.dipl.vikuk.service.tc` - obsahuje implementáciu testovacích scenárov pre webovú službu.

Makro sa skladá z dvoch základných častí:

- modul `VIKUKWS` – implementuje klienta webovej služby pomocou knižnice MS SOAP Toolkit,

- modul `Excel Module` – obsahuje podporné funkcie na manipuláciu so zošitom MS Excel.

Viac informácií o týchto balíkoch, triedach a rozhraniach je možné nájsť v dokumentácii aplikácie VIKUK-WS dodanej na CD, priloženej k tejto práci.

6.2.3 Popis webovej služby

Webová služba aplikácie VIKUK-WS sprístupňuje údaje o autoroch (pracovníkoch resp. pracoviskách UK) publikáciách a informácie o počte publikácií, ohlasov pre týchto autorov. Tabuľka 6.1 popisuje operácie webovej služby.

Poznámka: Každý pracovník a pracovisko UK má pridelené unikátne číslo v systéme VIKUK, aby ho bolo možné jednoznačne identifikovať. Pracovisko má navyše pridelený kód, ktorý predstavuje reťazec znakov. Pomocou tohto kódu je tiež možné jednoznačne identifikovať pracovisko v aplikácii VIKUK. Napr. UKOMFKI je kód Katedry informatiky Fakulty matematiky, fyziky a informatiky UK.

Operácia WS	Popis	Parametre
getDepartments	Vráti zoznam pracovísk, ktoré vyhovujú vstupnému parametru. Pracovisko je reprezentované komplexným objektom, ktorý obsahuje jednoznačný identifikátor (id), popis pracoviska (description) a kód pracoviska (code).	Reťazec (text) reprezentujúci názov alebo kód pracoviska resp. časť názvu alebo časť kódu pracoviska. Reťazec môže obsahovať špeciálne znaky ? a *. Pričom ? predstavuje jeden znak a * predstavuje nula alebo viac znakov.
getAllDepartments	Vráti zoznam všetkých pracovísk.	Žiadne.
getAuthors	Vráti zoznam pracovníkov (autorov publikácií), ktorí vyhovujú vstupným parametrom. Pracovník je reprezentovaný komplexným objektom, ktorý obsahuje jednoznačný identifikátor (id) a kód pracoviska (code), kde pracuje.	Reťazec (lastName) reprezentujúci priezvisko resp. časť priezviska, reťazec (firstName) reprezentujúci krstné meno resp. jeho časť a reťazec (code) reprezentujúci kód pracoviska, kde pracuje resp. jeho časť. Tento reťazec môže obsahovať špeciálne znaky ? a * (majú rovnaký význam ako pri operácii getDepartments).
getPublicationCount	Vráti počet publikácií pre zadaného autora (pracovník resp. pracovisko UK) podľa zadaných kritérií.	Číslo (authId) reprezentujúce jednoznačný identifikátor autora publikácie, reťazec (catgCode) reprezentujúci kód kategórie publikácie a číslo (year) reprezentujúci rok vydania publikácie.

getReferenceCount	Vráti počet ohlasov pre zadaného autora (pracovník resp. pracovisko UK) publikácie podľa zadaných kritérií.	Číslo (authId) reprezentujúce jednoznačný identifikátor autora publikácie, reťazec (pubCatgCode) reprezentujúci kód kategórie publikácie, číslo (pubYear) reprezentujúce rok vydania publikácie, reťazec (refCatgCode) reprezentujúci kód kategórie ohlasu a číslo (refPubYear) reprezentujúce rok ohlasu.

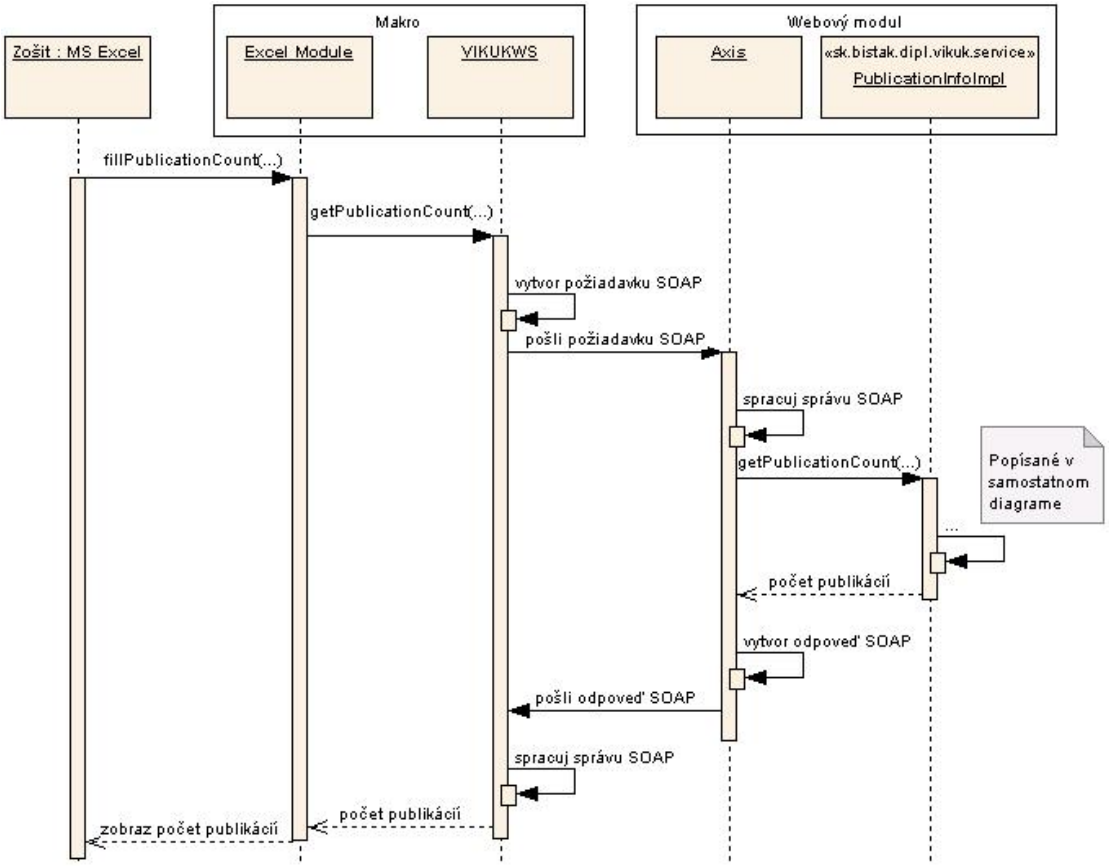
Tabuľka 6.1: Popis webovej služby.

Viac informácii o tejto webovej službe je možné nájsť v dokumente WSDL, ktorý obsahuje komentáre a je súčasťou aplikácie VIKUK-WS.

6.2.4 Príklad fungovania aplikácie

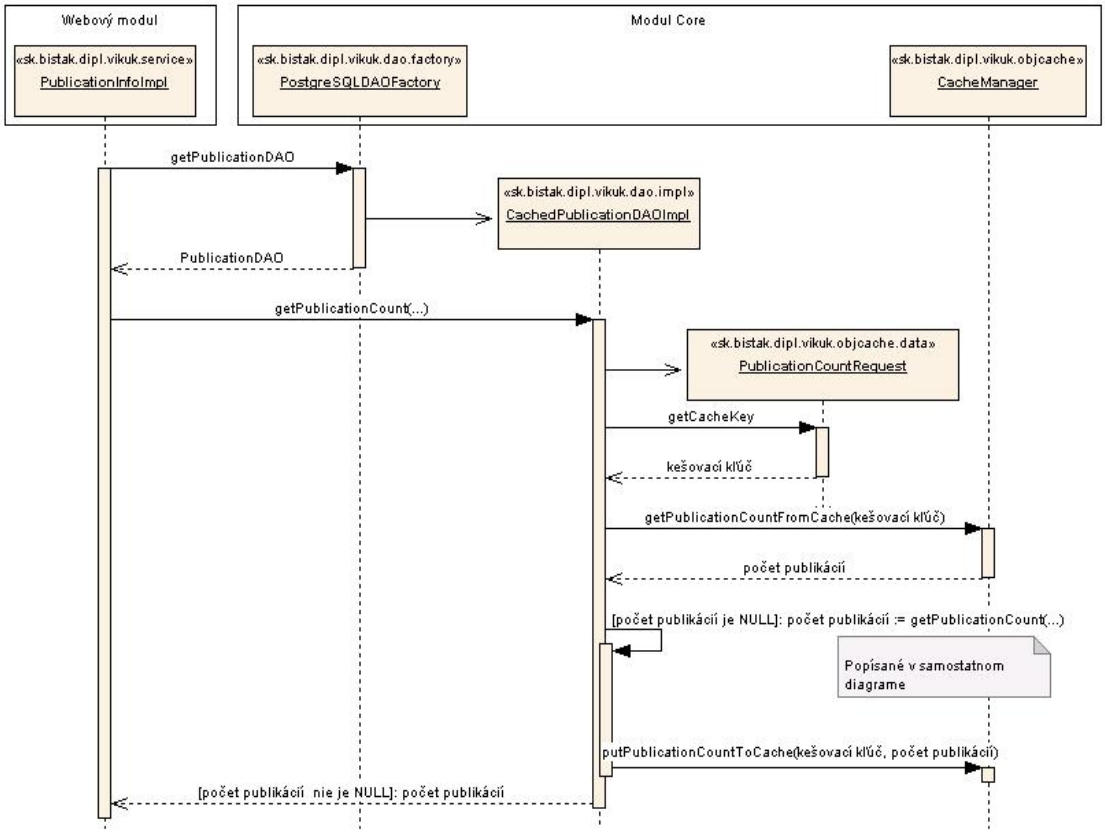
Nasledovné tri obrázky 6.4, 6.5 a 6.6 predstavujú sekvenčné diagramy UML, ktoré demonštrujú vyplnenie počtu publikácií v zošite MS Excel pomocou aplikácie VIKUK-WS.

Poznámka: Názvy tried a ich metód aplikácie VIKUK-WS sú napísané v anglickom jazyku, a preto sa aj v týchto diagramoch vyskytuje anglické pomenovanie tried a k nim prislúchajúcich metód.



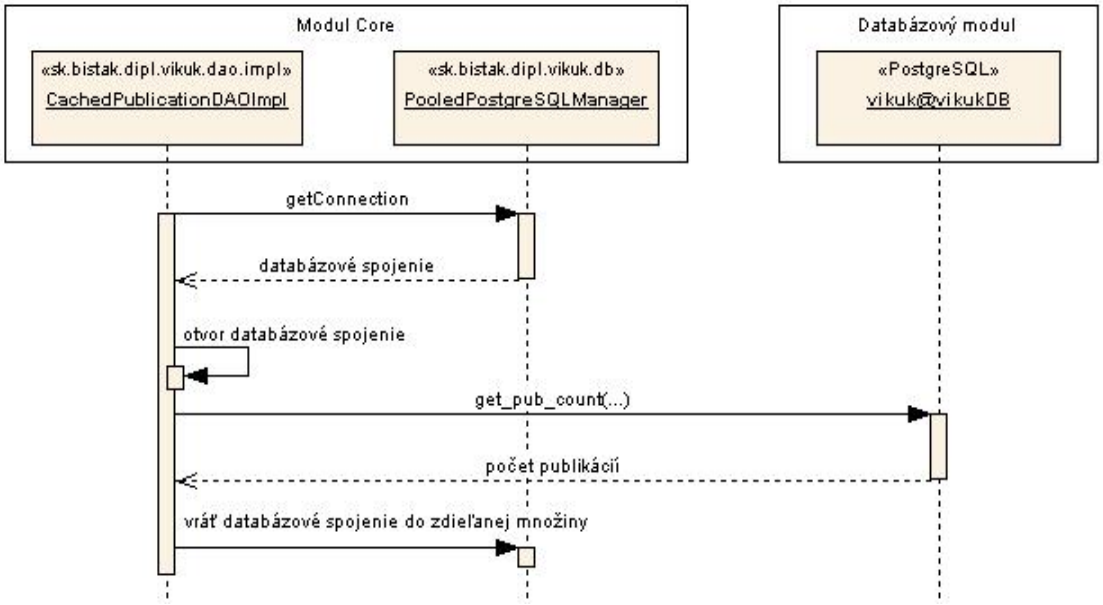
Obrázok 6.4: Sekvenčný diagram demonštrujúci komunikáciu medzi zošitom MS Excel a webovým modulom aplikácie VIKUK-WS.

Obrázok 6.4 znázorňuje ako zošit MS Excel s využitím makra zavolá webovú službu webového modulu, ktorá vystavuje metódu (`getPublicationCount`) na zistenie počtu publikácií pracovníkov a pracovísk UK. Samotné získavanie tohto počtu sa vykoná pomocou modulu Core. Získavanie počtu publikácií je znázornené na obrázku 6.5.



Obrázok 6.5: Sekvenčný diagram znázorňujúci detailnú komunikáciu medzi webovým modulom a modulom Core aplikácie VIKUK-WS.

Modul Core sa najskôr pokúsi nájsť počet publikácií v lokálnej keši (`getPublicationCountFromCache`). Ak ho tam nájde, tak ho priamo vráti webovému modulu. Ak tento počet nenájde v keši, tak ho získa pomocou funkcie PL/PGSQL databázového modulu (`get_pub_count`). Toto volanie funkcie PL/PGSQL je znázornené na obrázku 6.6. Takto získaný počet publikácií uloží pre ďalšie použitie do keši (`putPublicationCountToCache`).



Obrázok 6.6: Sekvenčný diagram zobrazujúci komunikáciu medzi modulom Core a databázovým modulom aplikácie VIKUK-WS.

6.3 Zabezpečenie

Aplikácia VIKUK-WS v súčasnosti neposkytuje citlivé dáta ku ktorým by mala mať prístup iba určitá skupina osôb alebo externých aplikácií. Aj z tohto dôvodu aplikácia VIKUK-WS dnes nie je špeciálne zabezpečená.

Aplikáciu VIKUK-WS by bolo možné zabezpečiť dvoma spôsobmi:

1. **Proti neoprávnenému prístupu** – je možné riešiť autentifikáciou a šifrovaním správ na úrovni protokolu SOAP pomocou technológie WS-Security. Keďže aplikácia je postavená na produkte Apache Axis, najjednoduchšie riešenie je použitie frameworku Apache WSS4J. Ďalšou možnosťou je migrácia webového modulu aplikácie na platformu Sun JWSDP a využitie zabezpečovacieho frameworku XWS-Security. Klienti takto zabezpečenej webovej služby by museli byť schopní pracovať s technológiou WS-Security. Napr. v prípade ukážkovej aplikácie by bolo potrebné doimplementovať túto funkčnosť do jej makra, nakoľko knižnica MS SOAP Toolkit nepodporuje technológiu WS-Security.

2. **Proti zahlteniu** – aplikácia VIKUK-WS zabezpečuje vykonanie časovo náročných funkcií PL/PGSQL, a preto existuje nebezpečenstvo zahltenia tejto aplikácie z dôvodu úmyselných, neopodstatnených a veľmi intenzívnych volaní webovej služby tejto aplikácie. Riziko takého zahltenia sa znižuje kešovaním požiadaviek na vykonanie funkcií PL/PGSQL v tejto aplikácii.

   Problém zahltenia aplikácie VIKUK-WS by bolo možné riešiť na úrovni protokolu HTTP. Každé HTTP spojenie s webovou službou by obsahovalo užívateľské meno a heslo. Webový server by tieto dva údaje overil

v komunikácii SOAP na úrovni bezpečnostného mechanizmu volanej triedy Java Servlet[22] alebo by sa overili s využitím dodatočného obslužného programu JAX-RPC (*handler*). Pri druhom spôsobe riešenia je možný audit počtu prístupov k webovej službe. Napr. pri prekročení denného limitu by sa prístup pre tohto používateľa k tejto službe automaticky zamietol. Ide o istý spôsob využitia autentifikácie pre potreby riešenia zahltenia aplikácie VIKUK-WS. Samotná autentifikácia používateľa by sa mohla realizovať v obslužných programoch JAX-RPC pomocou systému JAAS.

6.4 Nasadenie do ostrej prevádzky informačného systému UK

Ukážková aplikácia VIKUK-WS bola navrhnutá tak, aby bolo pri jej nasadení do ostrej prevádzky informačného systému UK nutné vykonať čo najmenej zmien. Je potrebné vykonať nasledovné kroky:

1. Nainštalovať J2SE (verziu 1.5.0_05) na niektorý server v rámci informačného systému UK.

2. Na tento server nainštalovať webový server Apache Tomcat (verzie 5.5.15).

3. Transformovať funkcie PL/PGSQL na funkcie jazyka PL/SQL. Predpokladáme, že k zvýšeniu doby odozvy aplikácie prispeje dodatočná optimalizácia vzniknutého kódu napísaného v jazyku PL/SQL, a tiež bude potrebná verifikácia príkazov jazyka SQL správcami aplikácie VIKUK.

4. Rozšíriť implementáciu aplikačného modulu o dátový zdroj Oracle a nastaviť správne pripojenia na databázu.

5. Nastaviť parametre kešovania aplikácie VIKUK-WS podľa možností cieľového servera (napr. podľa predpokladaného zaťaženia, veľkosti pamäti a podobne) a vytvoriť adresár na webovom serveri Tomcat, kde sa majú ukladať kešované dáta.

6. Nasadiť aplikačný modul na nainštalovaný webový server Apache Tomcat.

7. V makre zošitu MS Excel nastaviť správne umiestenie webovej služby (t.j. IP adresu a port nainštalovaného webového servera).

Celá aplikácia vrátane zdrojových kódov, konfiguračných súborov, vytváracieho a nasadzovacieho skriptu, licencie, technickej dokumentácie sa nachádza na CD priloženom k tejto práci. Aplikácia VIKUK-WS sa môže distribuovať a používať

[22] Ako bolo spomínané v skorších kapitolách, webové služby v produkte Apache Axis sú implementovaná pomocou technológie Java Servlet.

v rámci softvérovej licencie GPL[23]. Z dôvodu, že táto aplikácia je otvorená pre ďalšie použitie a rozšírenie pre kohokoľvek (t.j. aj pre ľudí v zahraničí), celá aplikácia vrátane dokumentácie je písaná v anglickom jazyku.

6.5 Problémy pri vytváraní aplikácie

Pri návrhu a vytváraní aplikácie VIKUK-WS sme zaznamenali niekoľko problémov. Najväčšie problémy sa týkali dát a dátového modelu aplikácie VIKUK.

VIKUK obsahuje dátový model, ktorý je podľa nášho názoru veľmi zložitý a jeho databázové tabuľky obsahujú heterogénne dáta. Príkladom je tabuľka `author_heading`, ktorá obsahuje popis pracovníkov, pracovísk, konferencií a seminárov, ktoré súvisia s publikačnou činnosťou UK. Ak sme chceli zistiť počet publikácií pracovísk UK, museli sme pracoviská v tejto tabuľke najskôr odlíšiť od ostatných typov entít (napr. od pracovníkov). Ďalším problémom tabuliek dátového modelu VIKUK je, že niektoré stĺpce týchto tabuliek obsahujú viacero rôznych údajov. Príkladom je stĺpec `heading_display` tabuľky `author_heading`, ktorý obsahuje názov pracoviska spolu s jeho kódom. Tento kód predstavuje jeden z jednoznačných identifikátorov pracoviska v aplikácii VIKUK.

Problémom je tiež štruktúra a obsah dát. Existujú minimálne štyri spôsoby, ako sú pracovníci zapísaní v tabuľke `heading_display`. Príkladom je formát `Priezvisko, Meno Kód_oddelenia` a formát `Priezvisko, Meno Rok-Kód_oddelenia`. `Rok-` reprezentuje rok nástupu pracovníka na pracovisko, ktoré je identifikované kódom (`Kód_pracoviska`).

Tieto problémy spôsobujú spomalenie vyhľadávania údajov o publikačnej činnosti a v niektorých prípadoch zabraňujú možnosti automatického spracovania týchto údajov.

[23] General Public License (GPL) – licencia garantujúca slobodu zdieľania a zmeny slobodného softvéru. Viac informácií možno nájsť v licencii aplikácie VIKUK-WS a v [47].

Záver

Cieľom tejto práce je prieskum, analýza a popísanie súčasného stavu v oblasti webových služieb so zameraním sa na ich použitie v podnikových informačných systémoch. Ide konkrétne o oblasť bezpečnosti, podpory transakcií a čiastočne aj o technológiu Web Services Invocation Framework a prostriedky na sprístupnenie webových služieb klientom. Úlohou je analýza relevantných štandardov (napr. WS-Security, WS-Transaction a ďalšie) a prieskum súčasného stavu ich implementácie. Ďalším cieľom je vytvorenie ukážkovej aplikácie demonštrujúcej využitie niektorých preskúmaných technológií v kontexte podnikového informačného systému.

Podľa nášho názoru sa nám tento cieľ podarilo splniť. Preskúmali sme jednotlivé štandardy, popísali ich vzájomný vzťah, možnosť ich prepojenia a načrtli spôsob ich využitia v prostredí podnikových informačných systémov. V práci sme sa zamerali hlavne na štandardy WS-Security a WS-Transaction, ktoré v súčasnosti predstavujú základ bezpečnosti a podpory transakcií vo webových službách.

Preskúmali a otestovali sme najvýznamnejšie implementácie bezpečnostných štandardov, medzi ktoré patria produkty Apache XML Security (implementácia XML podpisu a XML šifry), Sun XWS-Security a Apache WSS4J (implementácie WS-Security), VeriSign TSIK (implementácia WS-Security a XKMS) a implementácia jazyka XACML od spoločnosti Sun. V oblasti transakcií existujú iba dva štandardy - WS-Transaction a WS-CAF. V tejto oblasti sme preskúmali a otestovali dve implementácie štandardu WS-Transaction – Apache Kandula a ArjunaTS. Nepodarilo sa nám však nájsť žiadnu implementáciu štandardu WS-CAF. Väčšina týchto implementácií je zameraná na aplikácie, ktoré sú vyvinuté v jazyku Java. Pri práci s týmito implementáciami sme získali skúsenosti s dvoma implementáciami webových služieb Sun JWSDP a Apache Axis, ktorých popis a porovnanie sme zahrnuli tiež do obsahu tejto práce. Zanalyzovali sme možnosti prepojenia niektorých bezpečnostných a transakčných implementácií.

V ďalšej časti tejto práce sme preskúmali, popísali a otestovali technológiu Web Services Invocation Framework spolu s jej implementáciou. Vytvorili sme príklad, v ktorom sme načrtli využitie tejto technológie v prostredí informačného systému UK.

Na konci práce sme zanalyzovali možnosti a prostriedky na prístup k webovým službám. V rámci tejto analýzy sme našli spôsob prístupu k webovým službám z balíku MS Office. Nepodarilo sa nám však nájsť žiadnu voľne dostupnú klientskú aplikáciu. Vytvorili sme a popísali ukážkovú aplikáciu VIKUK-WS, ktorá demonštruje využitie webových služieb v informačnom systéme UK a klientsky prístup k týmto službám. Pomocou aplikácie VIKUK-WS je možné pristupovať k údajom o publikačnej činnosti UK z ľubovoľnej aplikácie a dá sa využiť na automatické generovanie tlačových zostáv o publikačnej činnosti UK z prostredia MS Excel.

Pôvodne sme plánovali vytvoriť viacero aplikácií demonštrujúcich využitie preskúmaných technológií, ale kvôli nedostatku času a rozsahu tejto práce sme od týchto plánov upustili. Podľa nášho názoru je aplikácia VIKUK-WS postačujúca na znázornenie využitia webových služieb v podnikovom informačnom systéme a je ju

tiež možné bez väčších problémov rozšíriť o funkčnosť demonštrujúcu použitie technológií ako WS-Security a WS-Transaction.

Pri návrhu aplikácie VIKUK-WS a pri analýze jednotlivých technológií webových služieb sme využili a aplikovali poznatky z rôznych oblastí ako bezpečnosť, transakcie, webové služby, softvérové inžinierstvo (návrhové vzory, modelovanie pomocou jazyka UML, testovanie), databázy (optimalizácia príkazov SQL, zdieľanie databázových spojení, kešovanie databázových údajov) a poznatky z rôznych programovacích jazykov (Java, PostgreSQL, Visual Basic for Applications) a z technologickej platformy J2EE (Java Servlet, EJB, JTA, JMS).

Pri vytváraní tejto práce sme vychádzali zo štúdia odbornej literatúry, špecifikácií jednotlivých štandardov, informácií nájdených na internete a z dokumentácie jednotlivých produktov. Ďalej sme preskúmali a otestovali niektoré príklady dodávané s týmito produktmi. U niektorých *open source* produktov sme zanalyzovali aj ich zdrojové kódy, buď z dôvodu lepšieho pochopenia fungovania a základných princípov týchto produktov, alebo z dôvodu odstránenia chýb, ktoré vznikli pri inštalácii alebo pri behu týchto produktov.

Výsledky tejto práce sa dajú využiť v podnikovom informačnom systéme, ktorý využíva alebo plánuje využívať webové služby. Aplikácia VIKUK-WS pravdepodobne nájde svoje uplatnenie v informačnom systéme UK ako prostriedok na automatické spracovanie údajov o publikačnej činnosti UK.

V tejto práci by bolo možné pokračovať analýzou ďalších technológií súvisiacich s webovými službami alebo skúmaním a rozpracovaním iných oblastí a smerov využitia webových služieb. Ďalšou možnosťou je využitie výsledkov tejto práce pri širšej implementácii a nasadení zabezpečených webových služieb podporujúcich transakcie v rámci informačného systému UK.

Slovník pojmov

Business Process Execution Language (BPEL) Jazyk na modelovanie obchodných procesov v rámci ktorých sa využíva funkčnosť z rôznych existujúcich podnikových aplikácií. Jazyk BPEL má formát XML. Proces namodelovaný v jazyku BPEL je vykonávaný a manažovaný špeciálnym programom - *BPEL engine*.

Component Object Model (COM) Platforma na vytváranie a spravovanie komponentov v prostredí Microsoft Windows.

Common Object Request Broker Architecture (CORBA) Technológia vytvorená skupinou OMG, ktorá umožňuje realizovať distribuovaný systém. Zabezpečuje volanie operácií vzdialených objektov v distribuovanom prostredí.

Distributed Component Object Model (DCOM) Technológia spoločnosti Microsoft definujúca komponenty v distribuovanom prostredí a spôsob komunikácie medzi nimi. Vychádza z technológie COM.

Enterprise Java Beans (EJB) Technológia komponentov vytvorených v jazyku Java a uložených v špeciálnych kontajneroch EJB. Je súčasťou platformy J2EE.

Extensible Stylesheet Language Transformations (XSLT) Technológia na transformáciu dokumentov XML do inej štruktúry (napr. na iný dokument XML, dokument HTML, ...). Transformácia je popísaná špeciálnym dokumentom XML (*stylesheet*), v ktorom je presne definované, na akú štruktúru sa jednotlivé časti zdrojového dokumentu XML majú transformovať.

Heš (hash) Reťazec znakov a čísiel stanovenej dĺžky, ktorý sa vypočíta pomocou špeciálnej matematickej (hešovacej) funkcie zo zdrojovej správy. Ak sa zmení obsah správy, zmení sa aj hodnota vypočítaného hešu.

Internet Inter-Orb Protocol (IIOP) Protokol zabezpečujúci komunikáciu medzi objektmi ORB (Object Request Broker) technológie CORBA. Tento protokol beží nad protokolom TCP/IP.

Java APIs for XML based RPC (JAX-RPC) Špecifikácia programového rozhrania pre volanie vzdialených procedúr (*remote procedure call - RPC*) vytvorených v jazyku Java pomocou dokumentov XML.

Java Authentication and Authorization Service (JAAS) Špecifikácia mechanizmu a programových rozhraní určených na autentifikáciu a autorizáciu v programoch vytvorených v jazyku Java.

Java Cryptography Architecture (JCA) Špecifikuje mechanizmus a programové rozhrania pre šifrovanie a manipulovanie so šifrovacími kľúčmi v prostredí Java.

Java Database Connectivity (JDBC) Špecifikácia a implementácia programového rozhrania pre prístup k relačným databázam v jazyku Java.

Java Message Service (JMS) Systém na zabezpečenie spoľahlivej a asynchrónnej komunikácie medzi aplikáciami. Táto komunikácia je realizovaná výmenou správ. Tento systém je súčasťou platformy J2EE.

Java Servlet Technológia definujúca triedy jazyka Java, určené na prácu s protokolom HTTP. Tieto triedy sú umiestnené na webovom kontajneri J2EE. Táto technológia je súčasťou platformy J2EE.

Java Specification Request (JSR) Návrh špecifikácie programového rozhrania určitej technológie založenej na platforme Java. Tento návrh sa vytvára v rámci otvoreného procesu *Java Community Process* (JCP).

Java Transaction API (JTA) Špecifikácia programového rozhrania medzi manažérom transakcií a časťami distribuovaného systému medzi ktoré patria: manažér zdrojov (*resource manager*), aplikačný server a aplikácie podporujúce transakcie.

Java 2 Platform, Enterprise Edition (J2EE) Platforma určená na vývoj a beh distribuovaných, serverových, viacvrstvových aplikácií napísaných v jazyku Java. Tieto aplikácie (aplikácie J2EE) sú uložené na aplikačnom serveri (server J2EE).

Java 2 Platform, Standard Edition (J2SE) Prostredie určená na vývoj samostatne stojacich aplikácií v jazyku Java. Tieto aplikácie sú uložené a vykonávané vo virtuálnom stroji (Java Virtual Machine - JVM) na lokálnom počítači.

J2EE Connector Architecture (JCA) Architektúru na prepojenie platformy J2EE (aplikačný server a aplikácie postavené na platforme J2EE) s heterogénnymi podnikovými systémami (napr. aplikácia vyvinutá pre sálový počítač - *mainframe*, databázové systémy, staršie a iné aplikácie napísané v jazyku odlišnom od jazyka Java). Prepojenie je realizované pomocou adaptéru (*resource adapter*), ktorý zabezpečuje komunikáciu medzi platformou J2EE a podnikovým (*enterprise*) systémom.

Kanonikalizácia (canonicalization) Proces normalizácie dokumentu XML. Dva dokumenty XML môž byť sémanticky rovnaké, ale môžu sa líšiť napr. znakmi konca riadkov (Windows vs. Unix), počtom a umiestnením prázdnych znakov, formátom a poradím atribútov, rôznym umiestnením špecifikátorov menného priestoru elementov XML (*namespace*) a podobne

Kódovanie Base-64 (Base-64 encoding) Algoritmus na transformáciu binárnych dát do textovej podoby (reťazec ASCII znakov).

Lightweight Directory Access Protocol (LDAP) Protokol na prístup k adresárom prostredníctvom protokolu TCP/IP. V týchto adresároch sú údaje uložené v stromovej štruktúre. Každý záznam takéhoto adresára obsahuje údaje vo forme množiny pomenovaných atribútov.

Lístok (ticket) Kerberos Lístok s obmedzenou časovou platnosťou určený na autentifikáciu. Je vytvorený pomocou protokolu Kerberos s využitím tajného kľúča, ktorý je uložený v centre na distribúciu kľúčov (*Key Distribution Center*).

Open Source Filozofia spôsobu distribúcie a používania zdrojového kódu programu. Kód vytvorený podľa tejto filozofie je voľne prístupný a použiteľný pre ostatných vývojárov.

Procedural Language/PostgreSQL Structured Query Language (PL/PGSQL) Procedurálny jazyk rozširujúci jazyk SQL. Programy vytvorené v tomto jazyku sú uložené a vykonávané v prostredí databázy PostgreSQL. Je veľmi podobný jazyku PL/SQL.

Procedural Language/Structured Query Language (PL/SQL) Procedurálny jazyk spoločnosti Oracle rozširujúci jazyk SQL. Rozširuje jazyk SQL o nové štruktúry v podobe procedúr, funkcií, generátorov postupnosti čísiel (*sequences*), spúšťačov (*triggers*), vlastných typov a podobne. Programy napísané v tomto jazyku sú uložené a vykonávané v prostredí databázy Oracle.

Remote Method Invocation (RMI) Technológia na realizáciu prístupu ku vzdialeným objektom vytvoreným v jazyku Java.

Remote Procedure Call (RPC) Protokol na volanie procedúr a funkcií umiestených a vykonávaných na vzdialenom počítači.

RMI-IIOP (RMI cez IIOP) Prenos volaní RMI cez protokol IIOP.

Secure Socket Layer (SSL) Protokol na zabezpečenie spojovacej (*session*) vrstvy pomocou kľúča symetrickej šifry, ktorý je vytvorený pomocou asymetrickej šifry.

Service Oriented Architecture (SOA) Architektonický vzor založený na službách (nielen webových), ktoré predstavujú základný kameň architektúry aplikácie, ktorá je vytvorená podľa tohto vzoru.

Simple Object Access Protokol (SOAP) Protokol založený na výmene správ vo formáte XML, ktorý slúži na prístup k funkčnosti a dátam aplikácií vytvorených v rôznych programovacích jazykoch a na rôznych platformách. Spolu s dokumentom WSDL predstavuje základ webových služieb.

Web Service Definition Language (WSDL) Jazyk vo formáte XML na vytvorenie presného popisu webovej služby. Popisuje štruktúru operácií služby, štruktúru správ SOAP (vstupné a výstupné parametre), akým spôsobom majú byť správy SOAP prenášané (napr. cez HTTP, SMTP, JMS), miesta uloženia webovej služby a podobne.

XPath Jazyk vo formáte XML na zápis výrazov pomocou ktorých je možné presne identifikovať a vyhľadávať určitú časť dokumentu XML.

Zoznam bibliografických odkazov

[1] MONSON-HAEFEL, R. *J2EE™ Web Services*. Addison-Wesley, 2003. ISBN 0-321-14618-2.

[2] ROSENBERG, Jothy, REMY, David, L. Securing *Web Services with WS-Security*. Indianapolis: Sams Publishing, 2004. ISBN 0-672-32651-5.

[3] BARTEL, M., BOYER, J., FOX, B., LAMACCHIA, B., SIMON, E. *XML-Signature Syntax and Processing* [online]. W3C, Feb. 2002. Špecifikácia štandardu. Formát HTML. Dostupné na internete: <http://www.w3.org/TR/xmldsig-core>

[4] IMAMURA, T., DILLAWAY, B., SIMON, E. *XML Encryption Syntax and Processing* [online]. W3C, Dec. 2002. Špecifikácia štandardu. Formát HTML. Dostupné na internete: <http://www.w3.org/TR/xmlenc-core>

[5] NADALIN, A., KALER, C., HALLAM-BAKER, P., MONZILLO, R. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1* [online]. OASIS, Sep. 2003. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>

[6] *OASIS Web Services Security (WSS) Technical Commitee* [online]. Formát HTML. Dostupné na internete: <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss>

[7] NADALIN, A., KALER, C., HALLAM-BAKER, P., MONZILLO, R. *Web Services Security SOAP Message Security 1.0* : WS-Security 2004 [online]. OASIS, Mar. 2004. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

[8] NADALIN, A., GRIFFIN, P., KALER, C., HALLAM-BAKER, P., MONZILLO, R. *Web Services Security Username Token Profile 1.0* [online]. OASIS, Mar. 2004. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

[9] BAJAJ, S., BOX, D., CHAPPELL, D. *Web Services Policy Framework (WSPolicy)* [online]. Verzia 1.2. Mar. 2006. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <http://xml.coverpages.org/ws-policy200603.pdf>

[10] HALLAM-BAKER, P., MYSORE, S. *XML Key Management Specification (XKMS 2.0)* [online]. Verzia 2. W3C, Jun. 2005. Formát HTML. Dostupné na internete: <http://www.w3.org/TR/xkms2>

[11] *Liberty Alliance Project* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.projectliberty.org>

[12] *Open Source Initiative* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.opensource.org>

[13] *The Legion of the Bouncy Castle* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.bouncycastle.org>

[14] *Apache XML Security* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://xml.apache.org/security>

[15] *OpenSAML* [online]. Internet2. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.opensaml.org>

[16] *Internet2* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.internet2.edu>

[17] *Java Community Process* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://jcp.org>

[18] *The Java™ Web Services Tutorial* [online]. Sun Microsystems, Jun. 2005. Príručka. Formát HTML. Dostupné na internete: <http://java.sun.com/webservices/docs/1.6/tutorial/doc/index.html>

[19] *GlassFish project* [online]. Domovská stránka projektu. Formát HTML. Dostupné na internete: <https://glassfish.dev.java.net>

[20] *Java Web Services Developer Pack* [online]. Sun Microsystems. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://java.sun.com/webservices/jwsdp/index.jsp>

[21] *Apache Axis* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://ws.apache.org/axis>

[22] KALER, C., HALLAM-BAKER, P., MONZILLO, R., NADALIN, A. *Web Services Security Security Assertion Markup Language (SAML) Token Profile 1.0* [online]. OASIS, Dec. 2004. Špecifikácia štandardu. Formát PDF. Dostupné na internete: <http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf>

[23] *Apache WSS4J (Web Service Security For Java)* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://ws.apache.org/wss4j>

[24] *Sun's XACML Implementation* [online]. Sun Microsystems. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://sunxacml.sourceforge.net>

[25] *Apache TSIK (Trust Service Integration Kit)* [online]. Apache Software Foundation. Domovská stránka inkubačného projektu. Formát HTML. Dostupné na internete: <http://incubator.apache.org/tsik/index.html>

[26] *Trust Service Integration Kit (TSIK)* : XML Trust Services [online]. VeriSign. Domovská stránka produktu. Formát HTML. Dostupné na internete:

<http://www.verisign.com/products-services/security-services/pki/xml-trust-services/index.html>

[27] TANENBAUM, Andrew, S., STEEN, Maarten van. *Distributed systems. Principles and Paradigms*. International Edition. New Jersey : Prentice Hall, 2002. ISBN 0-13-121786-0.

[28] *Java™ Transaction API (JTA) Specification* [online]. Verzia 1.0.1B. Sun Microsystems, Nov. 2002. Špecifikácia technológie. Formát PDF. Dostupné na internete: <http://java.sun.com/products/jta>

[29] CABRERA, L.F., COPELAND, G., JOHNSON, J. In *Coordinating Web Services Activities with WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity* [online]. Microsoft Corporation, Jan. 2004. Formát HTML. Dostupné na internete: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsacoord.asp>

[30] CABRERA, L.F., COPELAND, G., COX, W. *Web Services Coordination (WS-Coordination)* [online]. MSDN Library, Sep. 2003. Špecifikácia štandardu. Formát HTML. Dostupné na internete: <http://msdn.microsoft.com/webservices/webservices/understanding/advancedwebservices/default.aspx?pull=/library/en-us/dnglobspec/html/ws-coord0903.asp>

[31] *OASIS Web Services Composite Application Framework (WS-CAF) Technical Committee* [online]. Formát HTML. Dostupné na internete: <http://www.oasis-open.org/committees/ws-caf/charter.php>

[32] *JOTM (Java Open Transaction Manager)* [online]. ObjectWeb.Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://jotm.objectweb.org>

[33] *JBoss* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.jboss.com>

[34] *Apache Addressing* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://ws.apache.org/addressing>

[35] *Apache Kandula* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://ws.apache.org/kandula>

[36] *ArjunaTS (Arjuna Transaction Service)* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.arjuna.com/products/arjunats/index.htm>

[37] *JBoss Transactions* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://www.jboss.com/products/transactions>

[38] *Java™ Message Service Specification* [online]. Verzia 1.1. Sun Microsystems, Apr. 2002. Špecifikácia technológie. Formát PDF. Dostupné na internete: <http://java.sun.com/products/jms/docs.html>

[39] *J2EE™ Connector Architecture Specification* [online]. Verzia 1.5. Sun Microsystems, Nov. 2002. Špecifikácia technológie. Formát PDF. Dostupné na internete: <http://java.sun.com/j2ee/connector/download.html>

[40] *OASIS Web Services Business Process Execution Language (WSS) Technical Commitee* [online]. Špecifikácia štandardu. Špecifikácia štandardu. Formát HTML. Dostupné na internete: <http://www.oasis-open.org/committees/wsbpel/charter.php>

[41] BRIGHT, Marta. Web Services – Anyhow, Anywhere. In *Oracle Magazine* [online]. September-October 2005, p. 63-64. Dostupné na internete: <http://www.oracle.com/technology/oramag/oracle/05-sep/o55standard.html>

[42] *Apache WSIF (Web Services Invocation Framework)* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://ws.apache.org/wsif>

[43] *Microsoft SOAP Toolkit* [online]. Domovská stránka produktu. Formát HTML. Dostupné na internete: <http://msdn.microsoft.com/webservices/webservices/building/soaptk/default.aspx>

[44] *Apache Jakarta Commons DBCP* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://jakarta.apache.org/commons/dbcp>

[45] *Apache Jakarta Java Caching System (JCS)* [online]. Apache Software Foundation. Domovská stránka projektu. Formát HTML. Dostupné na internete: <http://jakarta.apache.org/jcs>

[46] DEEPAK, Alur, CRUPI, John, MALKS, Dan. *Core J2EE Patterns: Best Practicies and Design Strategies*. Second Edition. Palo Alto : Sun Microsystems Press – A Prentice Hall Title, 2003. ISBN 0-13-142246-4.

[47] *GNU General Public License* [online]. Domovská stránka. Formát HTML. Dostupné na internete: <http://www.gnu.org/copyleft/gpl.html>