MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA PETROL-GAZE DIN PLOIEȘTI
ȘCOALA DOCTORALĂ

REZUMAT - TEZĂ DE DOCTORAT

SISTEM EXPERT SENZITIV LA CONTEXT PENTRU OBIECTE DE PATRIMONIU

Conducător științific
Prof. univ. dr. ing. Nicolae PARASCHIV

Autor
Ing. Mihai Emilian Bărbos

Ploiești
2019

Mulțumiri

La finalul acestei etape din viața mea, doresc să adresez câteva cuvinte de mulțumire celor care m-au îndrumat sau mi-au acordat sprijinul pe durata întocmirii prezentei teze de doctorat.

În primul rând doresc să mulţumesc coordonatorului ştiinţific, domnul profesor Nicolae Paraschiv, pentru îndrumarea competentă, răbdarea, înţelegerea, sprijinul, încurajarea şi susţinerea acordate de-a lungul perioadei de elaborare a tezei de doctorat, precum şi pentru faptul că mi-a oferit libertatea de a aborda aceasta temă într-un mod personal.

Mulțumesc domnului academician Filip Florin Gheorghe și domnului profesor Stefan Trăușan-Matu pentru că au acceptat să analizeze teza de doctorat și să facă parte din comisia de susținere publică a acesteia.

Mulțumesc întregii comisii de îndrumare alcătuită din doamna profesoară Mihaela Oprea, doamna profesoară Otilia Cangea și domnul profesor Gabriel Rădulescu pentru sfaturile şi sugestiile oferite pe tot parcursul stagiului meu doctoral.

Mulțumesc tuturor membrilor departamentului de Automatică, Calculatoare și Electronică din cadrul facultății Inginerie Mecanică și Electrică al Universității de Petrol și Gaze din Ploiești unde am desfășurat activitățile de cercetare din timpul stagiului doctoral.

Nu în ultimul rând, doresc să mulțumesc familiei pentru susținerea morală și înțelegerea de care a dat dovadă de-a lungul acestei etape a vieții mele.

Ploiești, Decembrie 2019

Cuprins

Introducere

Într-un sens mai larg, senzitivitatea la context reprezintă abilitatea unui sistem de a determina și reacționa în timp real la evenimente survenite în mediul extern. Evenimentele monitorizate de către sistem sunt cele relevante pentru contextul sau circumstanțele utilizatorului.

Progresele științifice caracteristice secolului XXI au condus la dezvoltarea de noi tehnologii, protocoale, servicii și aplicații specifice sistemelor senzitive la context. Dintre acestea pot fi menționate:

- dispozitivele de tip *beacon* care oferă o alternativa eficienta în vederea determinării contextului;
- specificațiile protocoalelor *iBeacon* (*Apple*) și *Eddystone* (*Google*) care definesc formatul mesajelor *BLE* folosite de către dispozitivele de tip *beacon*;
- librarii software precum *Google Awareness API* și *Proximity Beacon API* care permit determinarea contextului utilizatorului combinând date provenite din diferite surse și / sau de la diferiți senzori;
- inițiative venite în sprijinul dezvoltării de sisteme și aplicații senzitive la context cum ar fi proiectul *Physical Web* (*Google*);
- servicii și aplicații software care includ module senzitive la context și pe bază de localizare ca de exemplu *Waze*, *Allrecipes* și *Yelp*.

Realizările menționate anterior creează un cadru favorabil dezvoltării de noi sisteme senzitive la context. Drept urmare, prezenta teza de doctorat are obiectivele enumerate mai jos:

O1. Realizarea de cercetări cu privire la posibilitatea dezvoltării unui sistem expert senzitiv la context, destinat furnizării de conținut interactiv turiștilor pe durata vizitelor la siturile culturale.

O2. Transpunerea în practică a rezultatelor aferente obiectivului O1 prin realizarea unui sistem experimental, respectiv a unui demonstrator pentru sistemul expert senzitiv la context propus.

Prin patrimoniu cultural înțelegem reprezentarea modului de viața al unei comunități, transmis din generație în generație, incluzând obiceiuri, practici, locuri și obiecte, expresii artistice și valori [1]. Un obiectiv principal în ceea ce privește managementul patrimoniului cultural este comunicarea semnificației și a necesității de prezervare a acestuia, atât membrilor comunității gazde, cât și vizitatorilor străini [2].

Pornind de la acest deziderat, se poate considera că o vizită la un sit cultural reprezintă un proces de asimilare pentru turist. În timpul vizitei, turistul își însușește informații relevante despre obiceiurile, practicile, locurile, obiectele, expresiile artistice și valorile relaționate cu respectivul sit cultural. Din aceasta perspectivă, au fost identificați mai mulți factori ce pot influenta în mod negativ procesul de asimilare a informațiilor de către turist. Astfel de factori pot fi: personalitatea, experiența, preferințele, interesele și starea de spirit ale turistului [3]. Analiza literaturii de specialitate sugerează faptul că un proces de asimilare / învățare interactiv și adaptiv poate conduce la un nivel crescut de conștientizare și la un interes sporit din partea turistului.

Sistemul expert propus, implementat prin intermediul demonstratorului realizat de către autor, permite crearea de asocieri virtuale între personaje digitale (*avatar*) și obiecte de patrimoniu. Fiecare obiect de patrimoniu din lumea reală poate avea un corespondent în cadrul sistemului. Totodată, pentru fiecare obiect de patrimoniu definit în sistem vor exista și unul sau mai multe personaje digitale asociate cu acesta.

Sistemul experimental, se bazează pe conceptul interacțiunilor virtuale între turiști și personajele digitale. Vizitatorii unui sit cultural nu sunt simpli spectatori. Aceștia vor putea sa *comunice* cu personajele digitale prin intermediul sistemului. În cadrul demonstratorului, un personaj digital are rolul de a comunica turiștilor informații despre obiectul de patrimoniu căruia îi este asociat.

Pentru a-și îndeplini rolul, personajele digitale prind *viață* pe terminalele de tip *smartphone* ale vizitatorilor și poartă conversații cu aceștia. Conversațiile sunt adaptive în sensul că pot evolua diferit în funcție de interacțiunea dintre *avatar* (respectiv personajul digital) și turist. Stimulul care declanșează interacțiunea depinde de contextul utilizatorului, sau altfel spus, de proximitatea vizitatorului față de un anumit obiect de patrimoniu la un moment dat.

Demonstratorul este un sistem automat, tranzacțional, distribuit și are o arhitectură de tip client-server. Acesta monitorizează contextul utilizatorului pe durata unei vizite la un sit cultural, iar în baza modificărilor contextului, personaje digitale **inițializează automat conversații** cu utilizatorul pentru a-i transmite informații despre obiectele de patrimoniu.

Teza de doctorat intitulată *Sistem Expert Senzitiv la Context pentru Obiecte de Patrimoniu* este structurată în cinci capitole prezentate succint în cele ce urmează.

Capitolul 1 reprezintă o analiză detaliată a stadiului cunoașterii în domeniile conexe tezei de doctorat și a sistemului experimental propus spre realizare. În cadrul acestui capitol sunt prezentate noțiuni și concepte teoretice, tehnologii, domenii de aplicare și tipuri de probleme cu privire la:

- sisteme senzitive la context;
- servicii *WEB*;
- dispozitive *beacon BLE*;
- motoare de inferență bazate pe reguli;
- limbajul *UML* utilizat pentru proiectarea și modelarea sistemului experimental.
- principiile și metodele de proiectare software care au fost studiate și folosite ulterior la realizarea demonstratorului.

Capitolul 2 este dedicat metodelor și algoritmilor pentru determinarea contextului.

Prima parte a capitolului include o analiză asupra anumitor arhitecturi și protocoale de comunicație ce pot fi folosite în cadrul sistemelor senzitive la context. Analiza este axată pe următoarele 3 variante de implementare ale serviciilor *WEB* identificate în capitolul 1:

- servicii *WEB SOAP*;
- servicii *WEB RESTfull* / *XML*;
- servicii *WEB RESTfull* / *JSON*.

Pornind de la un scenariu generic, frecvent întâlnit în cadrul sistemelor senzitive la context, este propus un experiment în scopul determinării variantei optime (dintre cele 3 menționate anterior) din

punct de vedere al tipului de răspuns. Iar în baza rezultatelor obținute, este selectată varianta *RESTfull* / *JSON* pentru implementarea comunicației în cadrul demonstratorului.

A doua parte a capitolului include prezentarea a 3 metode pentru determinarea contextului propuse de către autor, și anume: MIBL (metoda indirectă bazată pe localizare), MDFC (metoda directă fără *cache*) și MDCC (metoda directă cu *cache*). Totodată, sunt analizate și posibilitățile de integrare ale metodelor propuse cu bazele de date relaționale și serviciile *WEB* de tip *RESTful*. În acest sens, este realizat un nou experiment în vederea determinării metodei optime din punct de vedere al numărului de cereri și al cantității de date transferate între client și server. Pe baza rezultatelor experimentale obținute, este aleasă metoda MDCC (metoda directa cu *cache*) pentru a fi utilizată în cadrul demonstratorului.

În **capitolul 3** sunt prezentate contribuțiile autorului cu privire la proiectarea și realizarea componentelor server ale sistemului experimental realizat.

Capitolul 3 începe prin definirea termenului demonstrator în raport cu obiectivele tezei de doctorat. Totodată este caracterizată corelația demonstrator – sistem expert. În acest sens, sunt evidențiate atât funcțiile generale ale sistemelor expert cât și modul de implementarea a acestora în cadrul demonstratorului realizat.

Capitolul 3 continuă prin enunțarea conceptului care stă la baza demonstratorului.

Ulterior, este propusă arhitectura software a sistemului experimental. Câteva dintre elementele server principale ale demonstratorului, dezvoltate de către autor și evidențiate în capitolul 3 sunt:

- baza de date;
- domeniului demonstratorului;
- componentele de acces la date;
- componenta de acces la cunoaștere;
- componenta de autentificare a utilizatorilor;

Modelul componentelor software server a fost realizat intermediul diagramelor UML (de clase și de secvențe). Aceste diagrame se regăsesc tot în capitolul 3 și în anexa A.2.**Error! Reference source not found.**

Capitolul 4 evidențiază contribuțiile autorului cu privire la proiectarea și realizarea aplicațiilor client ale demonstratorului.

Astfel, din punct de vedere al proiectării, sunt definite cazurile de utilizare aferente celor doua aplicații ale sistemului experimental, respectiv:

- 7 cazuri de utilizare identificate pentru aplicația *Android* destinată turiștilor;
- 7 cazuri de utilizare ale aplicației *WEB* pentru administratorii obiectelor de patrimoniu.

Tot în capitolul 4 sunt incluse și diagramele cazurilor de utilizare pentru cele două aplicații client.

În ceea ce privește realizarea aplicațiilor, capitolul 4 tratează probleme ce țin de arhitectura acestora. Mai mult decât atât, sunt evidențiate platformele, librăriile software și mediile de dezvoltare utilizate pentru realizarea celor două aplicații.

De asemenea, este propus și un set de 4 noi experimente în vederea determinării răspunsului demonstratorului la două tipuri de intrări (respectiv treaptă și rampă) în cazul utilizării metodelor MDFC și MDCC. În timpul experimentelor sunt monitorizate traficul generat în rețea, timpul de răspuns și resursele de memorie alocate de către serverul demonstratorului. Capitolul 4 conține și rezultatele obținute în urma acestor experimente.

Este de menționat faptul că primele două capitole corespund unor activități de studiu și analiză fiind subordonate obiectivului O1 al tezei de doctorat. În ceea ce privește capitolele 3 și 4, acestea tratează probleme legate de proiectarea și realizarea sistemului experimental, fiind relevante pentru îndeplinirea obiectivului O2.

În final, **capitolul 5** este rezervat concluziilor. Tot în capitolul 5 sunt prezentate atât contribuțiile originale ale tezei de doctorat, cât și modul în care a fost realizată diseminarea rezultatelor. În această ultimă parte, autorul tezei de doctorat propune și câteva posibile direcții de continuare ale cercetărilor.

Pe lângă cele 5 capitole principale, teza de doctorat mai include:

- Lista figurilor (118 elemente);
- Lista tabelelor (75 elemente);
- Un glosar (57 elemente) în care sunt explicați termenii și abrevierile folosite în teză;
- 3 anexe organizate după cum urmează:
    - Anexa A.1 cu date experimentale aferente capitolului 2;
    - Anexa A.2 cu diagrame de secvențe *UML* specifice componentelor server prezentate în capitolul 3;
    - Anexa A.3 cu rezultate obținute ca urmare a experimentelor prezentate în capitolul 4.
- bibliografia care conține 102 de referințe (22 dintre acestea fiind din surse accesibile pe Internet) prezentate în ordinea citarii în teză.

De asemenea, codului sursă aferent demonstratorului realizat de către autor este disponibil în arhiva (fișierul *arhiva_cod_sursa.rar*) de pe CD-ul ce însoțește teza de doctorat. Alături de arhiva menționată, respectivul CD conține și un fișier cu instrucțiuni (*readme.docx*).

Precizări referitoare la prezentul rezumat:

1. Rezumatul conține formele integrale pentru introducere, concluzii parțiale la fiecare capitol, concluziile generale, diseminarea rezultatelor, contribuții și direcții de cercetare.
2. Cuprinsul din teza de doctorat a fost adaptat pentru rezumat.
3. Au fost păstrate numerele asociate referințelor bibliografice, figurilor, tabelelor și relațiilor din teza de doctorat.
4. Din motive de spațiu, nu au fost incluse listele figurilor și tabelelor.
5. Din aceleași motive nu au fost incluse anexele.

Capitolul 1. Stadiul actual al cunoaşterii în domeniile conexe tezei de doctorat

Acest prim capitol al tezei de doctorat se constituie într-un studiu multidisciplinar axat pe mai multe domenii relevante pentru tematica acesteia. Teza este însoțita și de un demonstrator. În consecință, conceptele, noțiunile teoretice, tehnologiile și exemplele practice studiate și prezentate în cadrul acestui capitol sunt elemente suport pentru realizarea demonstratorului.

Demonstratorul este un sistem senzitiv la context. Așadar, capitolul 1 tratează în primul rând problematica sistemelor senzitive la context.

Unul dintre elementele principale ce stă la baza sistemului experimental dezvoltat este motorul de inferență, după cum reiese și din arhitectura software propusă în capitolul 3. Drept urmare, capitolul 1 include noțiuni și exemple caracteristice motoarelor de inferență bazate pe reguli.

Aspecte legate de proiectarea demonstratorului sunt prezente în două capitole ale tezei de doctorat (capitolul 3 și capitolul 4) și într-una dintre anexe (A.2). Acestea capitole includ diagrame *UML*. Ca atare, în capitolul curent este abordată și tematica limbajului *UML*.

În final, capitolul 1 introduce conceptul de *design-pattern*, termen consacrat în domeniul proiectării și dezvoltării software. Astfel, sunt prezentate câteva dintre metodele de proiectare și dezvoltare software relevante pentru realizarea demonstratorului.

Capitolului se încheie cu prezentarea concluziilor parțiale.

1.1. Concepte, tehnologii, realizări şi tipuri de aplicaţii caracteristice sistemelor senzitive la context

Prezentul subcapitol reprezintă un rezumat cu privire la stadiul actual al cunoașterii în domeniul sistemelor senzitive la context.

1.1.1. Sistemele senzitive la context

În subcapitolul 1.1.1 au fost definite noțiuni precum context, sistem senzitiv la context și serviciu pe bază de localizare. Totodată, au fost identificate tipurile de context (direct și indirect) și categoriile de servicii pe bază de localizare (*push* și *pull*). Nu în ultimul rând, au fost furnizate exemple de aplicații, servicii și librării software (*API*) caracteristice sistemelor senzitive la context ca de exemplu: *Google Awareness API*, *Google Beacon Platform*, *Allrecipes Dinner Spinner*, *Waze* etc.

1.1.2. Servicii *WEB*

În general, serviciile *WEB* au un domeniu foarte larg de aplicare acestea stând la baza comunicației între componentele sistemelor orientate pe arhitectura de tip client / server [9] [10]. Arhitectura client / server este una frecvent întâlnita și în rândul sistemelor senzitive la context [3] [5]. Drept urmare, serviciile *WEB* reprezintă o varianta viabilă pentru realizarea modulelor de comunicație în cadrul sistemelor senzitive la context.

În subcapitolul 1.1.2 este definit conceptul de serviciu *WEB*. De asemenea sunt prezentate tipurile de servicii *WEB* (*SOAP* și *RESTful*) împreună cu standardele și protocoalele relaționate acestora (respectiv *WSDL*, *XML*, *XSD*, *JSON* etc.).

În ceea ce privește stadiul actual, sunt furnizate câteva exemple concrete atât din categoria serviciilor *SOAP* cât și *RESTful*, respectiv: *AWS Product Advertising API*, *Twitter REST API*, *Google Maps Directions API*, *Bing Trafic API* etc.

1.1.3. Dispozitive de tip *beacon BLE*

Un *beacon* este un dispozitiv de dimensiuni mici (aproximativ 3cmX5cmX2cm) care emite periodic un identificator unic prin intermediul semnalelor radio [14]. Dispozitive compatibile, cum ar fi terminale de tip *smartphone* și tabletă, au abilitatea de a detecta semnalele radio emise și de a identifica în mod unic orice *beacon* aflat în apropiere [14]. Puterea semnalului și frecvența de emisie aferente unui dispozitiv *beacon* sunt configurabile [14].

Dispozitivele de tip *beacon* folosesc tehnologia BLE (*Bluetooth Low Energy*). Spre deosebire de tehnologia clasică *Bluetooth*, *BLE* oferă costuri și consum de energie reduse menținând o rază de acțiune similara [15].

Din punct de vedere al standardizării, există două protocoale disponibile pentru dispozitivele *beacon BLE*, și anume:

- *iBeacon* – un protocol dezvoltat de compania *Apple* care este disponibil pe sistemul de operare *iOS* 7 și pe versiunile ulterioare [14];
- *Eddystone* – un protocol dezvoltat de compania *Google* și care este de asemenea folosit în cadrul proiectului *Physical Web* [16].

Referitor la implementările hardware disponibile pe piață, există mai multe modele de *beacon* compatibile cu ambele protocoale [14].

Prin intermediul dispozitivelor de tip *beacon*, dezvoltatorii de aplicații au acum posibilitatea de a crea legături între evenimente din lumea digitală și mișcările utilizatorilor din lumea reală [17]. Astfel proximitatea unui utilizator fată de un obiect din lumea reală poate declanșa o acțiune digitală [17].

Dispozitivele de tip *beacon BLE* sunt folosite intr-o multitudine de aplicații din domeniul serviciilor pe bază de localizare și *IoT* (*Internet of Things*) cum ar fi [18]:

- furnizarea de informații relevante în funcție de localizare (*Point-of-Interest Information*);
- navigație în interiorul clădirilor (*Indoor Navigation*);
- localizarea și urmărirea bunurilor (*Asset Tracking*);
- utilizare eficientă a spațiului în interiorul clădirilor (*Space Utilization*).

Câteva exemple de dispozitive *beacon* produse de compania *Accent Systems* sunt: *IBKS 105*, *IBKS USB* și *IBKS PLUS*. Acestea se diferențiază prin tipul de alimentare, dimensiuni, autonomie, distanța de operare etc. În figura 1.2 sunt prezentate cele trei modele de dispozitive *beacon* menționate anterior (*IBKS 105*, *IBKS USB* și respectiv *IBKS PLUS*). Este de menționat faptul că, modelul *IBKS 105* a fost cel folosit în cadrul demonstratorului.

a b c

Fig. 1.2 Exemplu dispozitive *beacon IBKS 105* (a), *IBKS USB* (b) și *IBKS PLUS* (c)

Tabelul 1.11 include caracteristicile extrase din fișele tehnice ale celor 3 modele de *beacon* reprezentate în figura **Error! Reference source not found.**.

Tabel 1.11 Caracteristici tehnice pentru dispozitivele *beacon IBKS 105*, *IBKS USB* și *IBKS PLUS* [19]

#	*IBKS 105*	*IBKS USB*	*IBKS PLUS*
Dimensiuni	Ø52.6 x 11.3mm	38x 18.9 x 5.1mm	84 x 84 x 24mm
Material carcasă	*ABS*	*ABS*	*ABS*
Greutate	24g	3g	153g (cu baterie litiu), 182g (cu baterie alcalină)
Culoare carcasă	alb mat	alb mat	alb mat
Procesor	*Nordic nRF51822*	*Nordic nRF51822*	*Nordic nRF51822*
Metoda de fixare	autocolant	conector *USB*	autocolant,şuruburi,flanşe
Protocolul radio	*BLE*	*BLE*	*BLE*
Temperatura de operare	de la -25ºC până +60ºC	de la -25ºC până la +75ºC	de la -40ºC până la +85ºC (baterie litiu) / de la -20ºC până la +54ºC (baterie alcalină)
Distanță de operare	până la 50m	până la 100m	până la 100m
Temperatura de depozitare	de la 0ºC până la +35ºC	de la -40ºC până +125ºC	de la 5ºC până la +30ºC
Alimentare	baterie CR24773V – 1000mAh	USB 5V	4 baterii de tip AA
Protocoale / profile	*iBeacon* și *Eddystone* (*UID*, *URL*, *TLM* și *EID*)	*iBeacon* și *Eddystone* (*UID*, *URL*, *TLM* și *EID*)	*iBeacon* și *Eddystone* (*UID*, *URL*, *TLM* și *EID*)
Senzori opționali	*Hall*, Accelerometru	Temperatură	*Hall*, Accelerometru, Temperatură
Firmware	*OTA* (*Over The Air*)	*OTA*	*OTA*
Certificări	*CE*, *FCC*, *IC*, *Anatel*	*FCC* și *CE*	FCC, CE și IP67
Autonomie	30-40 luni în funcție de puterea de emisie (TX) pentru o frecvență de lucru de 1s / 3-4 luni în funcție de puterea de emisie (TX) pentru o frecvență de lucru de 100ms	-	aprox. 120 luni în funcție de puterea de emisie (TX) pentru o frecvență de lucru de 1s / 15-20 luni în funcție de puterea de emisie (TX) pentru o frecvență de lucru de 100ms

Următoarea caracteristică (ilustrată în figura 1.5) a fost preluată de la adresa https://accent-systems.com. Acestea exprimă dependența între raportul de putere la recepție (*RX*) și distanța de propagare a semnalului pentru modelul de *beacon IBKS 105* (folosit în cadrul demonstratorului) din tabelul 1.11. Curbele sunt trasate pentru toată mulțimea de valori posibile ale raportului de putere la emisie (*TX*). Raportul *TX* este un parametru care poate fi modificat pentru a permite configurarea dispozitivului *beacon* în funcție cerințele aplicației.

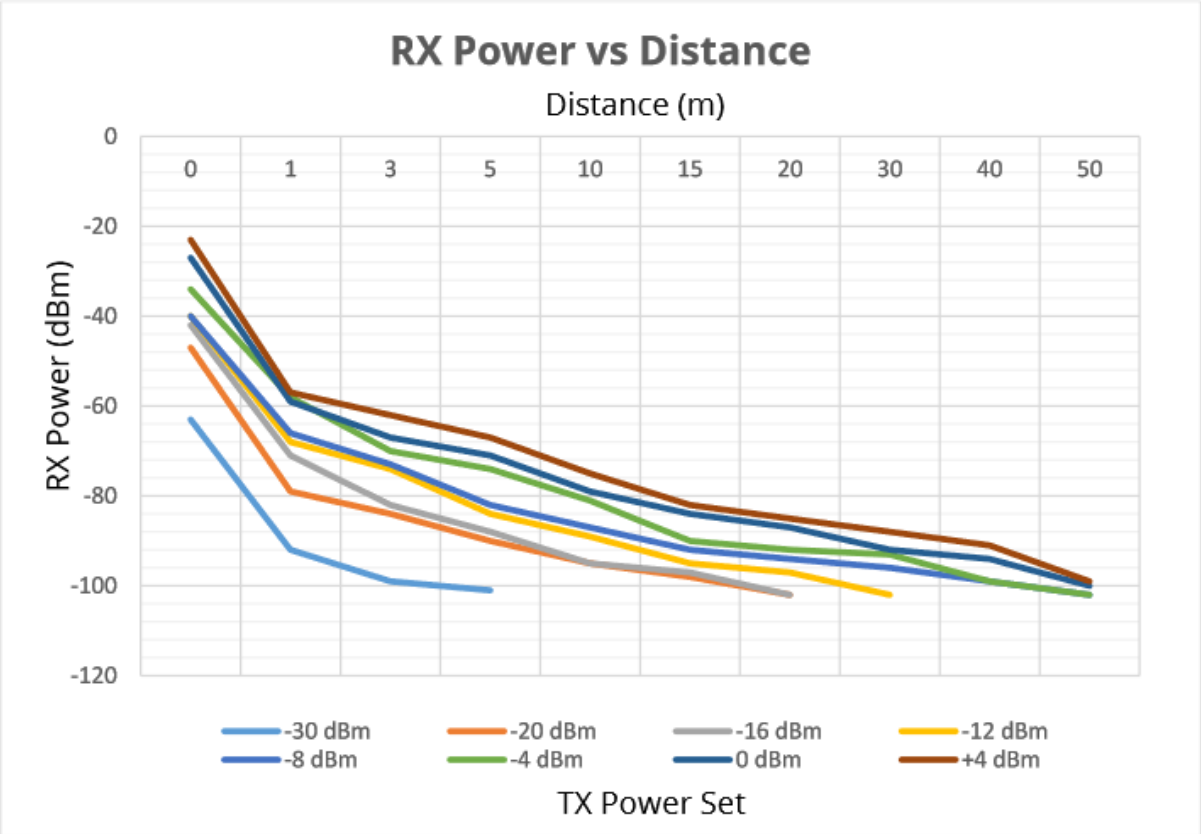

Fig. 1.5 Dependența între raportul de putere la recepție și distantă pentru modelul de *beacon IBKS 105* [19]

1.2. Motoare de inferență bazate pe reguli

În cadrul acestui capitol sunt definite noțiuni teoretice precum: sistem expert, regulă, faptă, bază de cunoștințe, model de reprezentare a cunoașterii, motor de inferență și raționament. Sunt prezentate și două tipuri de raționament caracteristice motoarelor de inferență (*forward chaining* și *backward chaining*).

În ceea ce privește posibilitățile de aplicare, motoarele de inferență sunt folosite în mai multe domenii dintre care și la conducerea proceselor în timp real [20] [22].

Totodată, în ultima vreme, motoarele de inferență au devenit parte integrantă a unor produse software din categoria sistemelor de management a proceselor de *business* (*BPMS – Business Process management System*) [24] [25] [26] [27] [28]. Motoarele de inferență stau la baza componentelor *BRE* (*Business Rules Engine*) din cadrul soluțiilor *BPMS* [24] [25] [26] [27] [28]. În același timp, soluțiile *BPMS* sunt folosite pentru definirea, analiza și automatizarea proceselor de *business* ale organizațiilor într-o multitudine de domenii și industrii precum telecom, finanțe, logistica etc. [26]. Cu toate ca fac parte din pachete de produse *BPMS*, în anumite situații, motoarele

de inferență pot fi folosite și independent de acestea. Tabelul 1.13 include câteva exemple de componente *BRE* din cadrul unor sisteme *BPMS* consacrate.

Tabel 1.13 Exemple componente *BRE* și *BPMS*

BRE / motor inferență	Model	Raționament	Platforma	Suport JSR-94	*BPMS*	*Open Source*	Vânzător / producător
Drools Expert [25]	reguli producție	raționament înainte (*Rete-OO*), înapoi și hibrid	*JAVA SE / JAVA EE*	DA	*Drools BPMS*	DA	*Red Hat*
Oracle Business Rules Engine [24]	reguli producție	raționament înainte (*Rete*)	*JAVA EE*	DA	*Oracle Fusion Middleware*	NU	*ORACLE*
OpenRules Rule Solver [26]	reguli producție	raționament înainte (*Rete*), secvențial	*JAVA EE*	DA	*OpenRules BPMS*	NU	*OpenRules*
BizTalk Business Rules Engine [27]	reguli producție	raționament înainte	*.NET*	NU	*BizTalk BPMS*	NU	*MICROSOFT*
IBM ODM Decision Server Rules (fostul *ILOG JRules*) [28]	reguli producție	raționament înainte (*RetePlus*)	*JAVA EE*	DA	*IBM ODM*	NU	*IBM*

Specificația *JSR-94* evidențiată în tabelul 1.13 definește un *API* care permite interacțiunea cu motoare de inferență din cadrul platformelor *Java SE* și *EE*.

Dintre exemplele prezentate în tabelul 1.13, doar motorul de inferență *Drools Expert* poate fi folosit independent (*stand-alone*). Celelalte variante presupun instalarea și utilizarea întregii suite *BPMS* împreună cu un server de aplicație aferent. Totodată, utilizarea motorului *Drools* nu implică costuri suplimentare deoarece este *open-source*. Drept urmare, acesta reprezintă soluția adoptată în cadrul demonstratorului.

În subcapitolul 1.2 sunt prezentate și câteva probleme de planificare și alocare a resurselor critice propuse la competiții internaționale. Câteva exemple de asemenea probleme care pot fi rezolvate prin utilizarea motoarelor de inferență sunt prezentate în continuare:

- generarea orarului la nivel de universitate (problema a fost propusă în cadrul celei de a treia sesiuni a competiției internaționale *ITC 2007 - Track 3 - Curriculum Course Scheduling*);
- planificarea automată a examenelor pentru o universitate (detalii despre modelul problemei pot fi găsite pe site-ul universității *Queen's University of Belfast* la pagina menționată în referința [1]);
- alocarea proceselor pe servere sau *load balancing* (problemă propusă de *Google* în cadrul competiției internaționale *ROADEF/EURO Challenge 2012: Machine Reassignment*).

Nu în ultimul rând, în subcapitolul 1.2 este exemplificat modul de utilizare al tabelelor de decizie într-o problemă care presupune determinarea ratei dobânzii unui depozit bancar la termen.

1.3. Limbajul *UML*

Conform figurii 1.11, preluată din referința [33], ingineria sistemelor se află la convergența mai multor domenii inginerești precum:

- inginerie software;
- inginerie electrică;
- inginerie mecanică;
- ingineria materialelor.

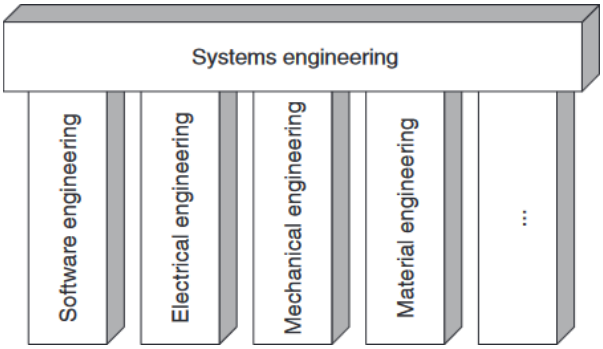

Fig. 1.11 Disciplinele ingineriei sistemelor [33]

Termenul *UML* reprezintă abrevierea pentru *Unified Modeling Language* și se referă la un limbaj vizual folosit în cadrul ingineriei software pentru proiectarea și modelarea sistemelor de programe [34].

Literatura de specialitate identifică mai multe domenii de aplicare ale limbajului *UML* cum ar fi:

- ingineria *hardware* (*hardware engineering*) și a proceselor economice (*business process engineering*) [35];
- modelarea aplicațiilor și sistemelor de timp real (*real-time systems*) [36] [37];
- modelarea sistemelor încorporate (*embedded systems*) [36] [37];
- modelarea sistemelor distribuite (*distributed computing systems*) [38].

Având în vedere atât cele menționate anterior, cât și faptul că tehnologiile suport ale demonstratorului realizat intră în sfera ingineriei software și a sistemelor de programe, limbajul *UML* a fost selectat pentru modelarea și proiectarea sistemului experimental.

Limbajul *UML* se bazează pe reprezentări grafice denumite diagrame. Diagramele *UML* reprezintă elementele fundamentale ale limbajului. Conform clasificărilor din referințele [38], [39] și [40], în funcție de scop, diagramele *UML* se împart în 2 categorii, și anume:

- diagrame structurale (*structural diagrams*);
- diagrame comportamentale (*behavioral diagrams*).

- Diagramele structurale sunt utile pentru modelarea unui sistem din punct de vedere static. Acestea pun în evidență elementele care alcătuiesc sistemul și relațiile dintre ele. Prin intermediul diagramelor structurale este descris sistemul (sau părți ale acestuia) din punct de vedere al funcțiilor îndeplinite. Diagramele structurale nu includ aspecte comportamentale ale sistemului (respectiv modul în care sistemul își îndeplinește funcțiile).
- Diagramele comportamentale sunt potrivite pentru modelarea unui sistem din punct de vedere dinamic. Acestea definesc modul în care sistemul se comportă în timp, evidențiind ordinea și condițiile necesare pentru declanșarea anumitor operații. Totodată, diagramele comportamentale pot să descrie interacțiunile dintre:

    - elementele interne ale sistemului;
    - sistem și mediul extern.

Figura 1.12, preluată și adaptată din [37], evidențiază tipurile de diagrame *UML* pentru fiecare dintre cele 2 categorii menționate anterior.

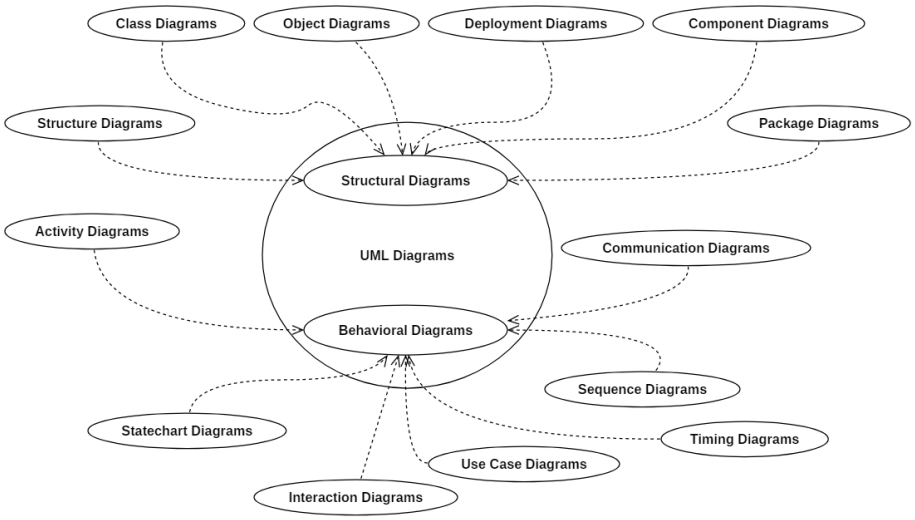

Fig. 1.12 Tipuri de diagrame *UML* [37]

În funcție de relevanța pentru proiectarea demonstratorului, în subcapitolul 1.3 sunt tratate următoarele aspecte legate de limbajul *UML*:

- cazuri de utilizare (*use cases*);
- diagrame de cazuri de utilizare (*use case diagrams*);
- diagrame de clase (*class diagrams*);
- diagrame de secvențe (*sequence diagrams*).

O importanță aparte pentru proiectarea aplicațiilor din cadrul demonstratorului o au cazurile de utilizare.

Conform definiției date în referința [41], un caz de utilizare reprezintă descrierea textuală a succesiunii operațiilor ce caracterizează interacțiunea dintre sistem și actorii externi în raport cu un anumit scop care trebuie îndeplinit.

Cu titlu de exemplu, în tabelul 1.18 este prezentat șablonul selectat pentru definirea cazurilor de utilizare ale demonstratorului, șablon preluat și adaptat din referința [41].

Tabel 1.18 Șablon folosit pentru definirea cazurilor de utilizare [41]

USE CASE <allocated code> CAZ UTILIZARE <cod alocat>	<expresie scurtă începând cu un verb>	
Scope Scop	<descrierea scopului cazului de utilizare>	
Primary actors Actori Principal	<denumirile rolurilor principalilor participanților ai cazului de utilizare>	
Secondary actors Actori Secundari	<denumirile rolurilor participanților secundari ai cazului de utilizare>	
Stakeholder & Interests Părți Interesate și Interese	*Stakeholder* Parte Interesată	*Interest* Interes
	<denumirea parții interesate>	<se descrie în ce fel este interesata partea>
	<denumirea parții interesate>	<se descrie în ce fel este interesata partea>
	…	…
Preconditions Precondiții	<se descrie starea în care trebuie să se afle sistemul înainte de execuția pașilor cazului de utilizare>	
Success End Condition Condiție Finală de Succes	<se descrie starea în care trebuie să se afle sistemul după execuția cu succes a pașilor cazului de utilizare>	
Failed End Protection Condiție Finală de Eșec	<se descrie starea în care trebuie să se afle sistemul ca urmare a întreruperii execuției pașilor cazului de utilizare datorita unei erori>	
Trigger Declanșator	<se descrie acțiunea asupra sistemului care declanșează executarea pașilor cazului de utilizare>	
DESCRIPTION DESCRIERE	*Step* Pas	*Action* Acțiune
	1	<se enumeră pașii cazului de utilizare în ordine începând cu primul de după declanșator până la îndeplinirea condiției finale de succes>
	2	…
	3	…
	4	…
	…	…
EXTENSIONS EXTENSII	*Step* Pas	*Branching Action* Acțiune Ramificație
	1a	<condiția care determina ramificația> <acțiunea sau numele cazului de utilizare referit>
	1a1	…
	1a1a	…
	1a1b	…
	1a2	…
	…	…

1.4. Principii și metode de proiectare software

În subcapitolul 1.4 sunt prezentate câteva metode de proiectare software relevante pentru realizarea demonstratorului (respectiv *Singleton*, *Builder*, *Factory*, *DAO* etc.). Totodată, sunt furnizate exemple de aplicare ale metodelor în limbajul de programare *Java* și sunt evidențiate

avantajele utilizării acestora. Exemplele din subcapitolul 1.4 au fost propuse și realizate de către autor.

1.5. Concluzii parţiale

Capitolul 1 reprezintă un studiu multidisciplinar axat pe mai multe domenii relevante pentru tematica tezei de doctorat. În același timp, conceptele, noțiunile teoretice, tehnologiile și exemplele practice studiate și prezentate în cadrul acestui capitol sunt elemente suport pentru realizarea demonstratorului.

Prima parte a capitolului 1 tratează problematica sistemelor senzitive la context. Sunt introduse și definite noțiuni precum context, sistem senzitiv la context și serviciu pe bază de localizare. Totodată, sunt identificate tipurile de context (direct și indirect) și categoriile de servicii pe bază de localizare (*push* și *pull*). Deoarece contextul poate fi direct sau indirect metodele folosite pentru determinarea acestuia sunt diferite. Avantajele și dezavantajele a trei asemenea metode (una directă și două indirecte) propuse de către autor urmează a fi analizate în capitolul 2 al tezei de doctorat printr-un studiu de caz. În ceea ce privește stadiul actual, sunt prezentate exemple de aplicații, servicii și librării software (*API*) din domeniul sistemelor senzitive la context precum: *Google Awareness API*, *Waze*, *Allrecipes Dinner Spinner* etc.

Serviciile *WEB* reprezintă o variantă viabilă pentru realizarea modulelor de comunicație în cadrul sistemelor senzitive la context cu arhitectură client-server. În consecință, capitolul 1 continuă prin definirea conceptului de serviciu *WEB*. Sunt prezentate tipurile de servicii *WEB* (*SOAP* și *RESTful*) împreună cu standardele și protocoalele relaționate (*WSDL*, *XML*, *XSD*, *JSON* etc.). Avantajele și dezavantajele celor două tipuri de servicii *WEB* sunt analizate în detaliu în capitolul 2 al tezei de doctorat. Referitor la stadiul actual, sunt furnizate câteva exemple concrete atât din categoria serviciilor *SOAP* cât și *RESTful*. Exemplele includ: *AWS Product Advertising API*, *Twitter REST API*, *Google Maps Directions API*, *Bing Trafic API* etc.

Dispozitivele *beacon BLE* sunt folosite într-o multitudine de aplicații din domeniul *IoT*. Acestea permit asocieri între evenimente din lumea digitală și mișcările utilizatorilor din lumea reală. Astfel, proximitatea unui utilizator față de un obiect din lumea reală poate declanșa o acțiune digitală. Având în vedere relevanța pentru domeniul sistemelor senzitive la context, subiectul dispozitivelor *beacon* este analizat în continuarea capitolului 1. Ca atare, este definita noțiunea de *beacon BLE* și sunt evidențiate standardele și profile relaționate (*iBeacon* și *Eddystone*). De asemenea, sunt prezentate trei modele de *beacon* existente pe piață împreună cu detalii despre caracteristicile tehnice ale acestora. Pentru fiecare dintre cele trei modele sunt incluse și caracteristicile statice (raport putere recepție în funcție de distanța de propagare a semnalului radio).

Sistemul experimental dezvoltat de către autor și prezentat în cadrul tezei de doctorat folosește un motor de inferență bazat pe reguli de producție. Capitolul 1 continuă prin abordarea tematicii motoarelor de inferență care este relevantă pentru realizarea demonstratorului. Așadar, sunt definite noțiuni esențiale precum: sistem expert, regulă, faptă, bază de cunoștințe, model de reprezentare a cunoașterii, motor de inferență și raționament. Sunt prezentate și două tipuri de raționament caracteristice motoarelor de inferență (*forward chaining* și *backward chaining*).

Cu privire la stadiul actual, în ultima vreme, motoarele de inferență au devenit parte integrantă a unor produse software din categoria sistemelor de management a proceselor de *business* (*BPMS – Business Process management System*). În acest sens, capitolul 1, evidențiază

câteva exemple consacrate de *BPMS* (*Drools*, *OpenRules*, *BizTalk*, *Oracle*) împreună cu motoarele de inferență caracteristice (*Drools Expert*, *OpenRules Rule Solver*, *BizTalk BRE*, *Oracle BRE*).

Totodată, motoarele de inferență stau și la baza unor produse software folosite pentru rezolvarea problemelor de planificare și alocare a resurselor critice în diferite industrii. Capitolul 1 prezintă câteva asemenea probleme cum ar fi planificarea orarului la nivel de universitate propusă în sesiunea 3 din cadrul competiției internaționale *ITC 2007 - Curriculum Course Scheduling*.

Motoarele de inferență au limbaje proprii care permit definirea regulilor de producție. Concomitent, mai există și posibilitatea folosirii tabelelor de decizie pentru definirea regulilor. Tabele de decizie au avantajul de a fi mai accesibile pentru personalul fără competențe de programare deoarece pot fi definite în *Excel*. În vederea exemplificării modului de utilizare a tabelelor de decizie, în capitolul 1 este propusă problema ipotetica de calculul al ratei dobânzii depozitelor la termen pentru o bancă. Pentru rezolvarea problemei din exemplu menționat anterior este folosit motorul *Drools Expert*.

Conform literaturii de specialitate, limbajul *UML* este folosit în cadrul ingineriei software pentru proiectarea și modelarea sistemelor de programe. Așadar, acesta reprezintă o alternativă viabilă cu privire la proiectarea și modelarea demonstratorului.

Limbajul se bazează pe reprezentări grafice denumite diagrame *ULM*. Acestea se împart în două categorii, respectiv: diagrame structurale și diagrame comportamentale. Diferențele dintre cele două categorii de diagrame sunt evidențiate în continuarea capitolului. Totodată, sunt prezentate în detaliu elementele caracteristice pentru următoarele tipuri de diagrame alese în vederea proiectării demonstratorului:

- diagrame de cazuri de utilizare;
- diagrame de clase;
- diagrame de secvențe.

În final, capitolul 1 definește noțiunea de metodă de proiectare software (*software design-pattern*). Sunt identificate câteva metode relevante pentru realizarea demonstratorului (*Singleton*, *Builder*, *Factory*, *DAO*, *MVC* și *MVVM*). Fiecare dintre acestea este însoțită de câte un exemplu concret de aplicare pentru limbajul de programare *Java*. Totodată, sunt evidențiate avantajele utilizării metodelor enumerate anterior precum:

- îmbunătățirea performatei aplicațiilor;
- reducerea timpului de execuție;
- o utilizare mai eficienta a memoriei;
- eliminarea ambiguităților din codul sursă;
- reducerea complexității codului sursă;
- sporirea transparentei codului sursă;
- reducerea efortului de dezvoltare și mentenanță.

Prezentarea metodelor este însoțită de mai multe exemple propuse de către autor.

Capitolul 2. Contribuții privind dezvoltarea și testarea unor metode pentru determinarea contextului

În subcapitolul 1.1.1 al tezei de doctorat a fost definită noțiunea de context. Au fost identificate și tipurile de context. Deoarece contextul poate fi indirect sau direct, atunci și metodele folosite pentru determinarea acestuia se împart în două categorii (metode indirecte și metode directe).

În cadrul acestui capitol, autorul propune trei metode (descrise pe larg în subcapitolul 2.2) pentru determinarea contextului după cum urmează:

- o metodă indirectă bazată pe datele de localizare ale utilizatorului obținute prin *GPS* (denumită în continuare metoda indirectă bazată pe localizare sau MIBL);
- o metoda directă bazată pe utilizarea de dispozitive *beacon* (denumită în continuare metoda directă fără *cache* sau MDFC);
- o metoda directă bazată pe utilizarea de dispozitive *beacon* împreună cu un *cache* local (denumită în continuare metoda directă cu *cache* sau MDCC).

Capitolul include o analiză comparativă a celor 3 metode (MIBL, MDFC, MDCC) pe fondul unui studiu de caz. Totodată sunt prezentate avantajele, dezavantajele și modul de aplicare a metodelor propuse. Analiza este realizată din punct de vedere al implementării sistemelor senzitive la context folosind baze de date relaționale și servicii *WEB*. În consecință, prima parte a capitolului tratează problematica folosirii serviciilor *WEB* pentru realizarea comunicației în cadrul sistemelor senzitive la context.

Pentru studiul de caz este considerată situația sistemelor care furnizează informații relevante utilizatorului în funcție de poziția geografica a acestuia (respectiv sisteme de tipul *Point-of-Interest Information*).

Rezultatele investigațiilor prezentate în cadrul acestui capitol au fost utile autorului pentru selectarea soluțiilor adecvate cu privire la:

- protocoalele și arhitecturile de comunicație utilizate în cadrul demonstratorului dezvoltat;
- metoda folosită pentru determinarea contextului în cadrul sistemului experimental realizat.

2.1. Analiza protocoalelor și arhitecturilor de comunicație folosite în cadrul sistemelor senzitive la context

Pentru sistemele senzitive la context cu arhitectură de tip client / server, comunicația între componentele *front-end* și *back-end* poate fi realizată prin intermediul serviciilor *WEB*. Conceptul de serviciu *WEB* a fost prezentat anterior în cadrul subcapitolului 1.1.2 al tezei de doctorat. Tot în subcapitolului 1.1.2 au fost introduse și descrise succint câteva arhitecturi, protocoale și tehnologii folosite în mod uzual la realizarea și exploatarea serviciilor *WEB*. Astfel, cu privire la tipurile de servicii *WEB* cunoscute, se poate defini următoarea clasificare:

- servicii *WEB SOAP*;
- servicii *WEB RESTful* / *XML* (mesajele între client și server sunt în format *XML*);

- servicii *WEB RESTful* / *JSON* (mesajele între client și server sunt în format *JSON*).

Acest subcapitol este consacrat unei analize comparative axată pe cele trei categorii de servicii *WEB* menționate anterior. Pornind de la un scenariu generic, frecvent întâlnit în cadrul sistemelor senzitive la context, este propus un experiment. Scopul experimentului este acela de a evidenția impactul componentelor de comunicație asupra comportamentului, performantei și evoluției în timp a unui sistem senzitiv la context.

Din considerente de spațiu, în cele ce urmează sunt evidențiate în mod sintetic rezultatele obținute în urma testului / simulărilor realizate. Mai multe detalii cu privire la aceste simulări sunt disponibile în cadrul tezei de doctorat.

În tipul testului au fost simulate 3 grupuri de utilizatori după cum urmează:

- un grup aferent variantei *SOAP*;
- un grup aferent variantei *REST* / *XML*;
- un grup aferent variantei *REST* / *JSON*.

Tabelul 2.6 conține rezultatele sintetice al testului pentru toate cele trei grupuri menționate anterior.

Tabel 2.6 Rezultatul centralizat pentru toate cele 3 grupuri

Group	*Samples*	*Min*	*Max*	*99% Line*	*95% Line*	*90% Line*	*Med.*	*Avg.*	*Error %*
SOAP	10	65	160	158	153	146	100	105	0.00%
REST / XML	10	30	89	86	78	67	46	48	0.00%
REST / JSON	10	7	45	43	37	29	16	20	0.00%

Rezultatele din tabelul 2.6 sunt prezentate sub formă grafică în figura 2.8.

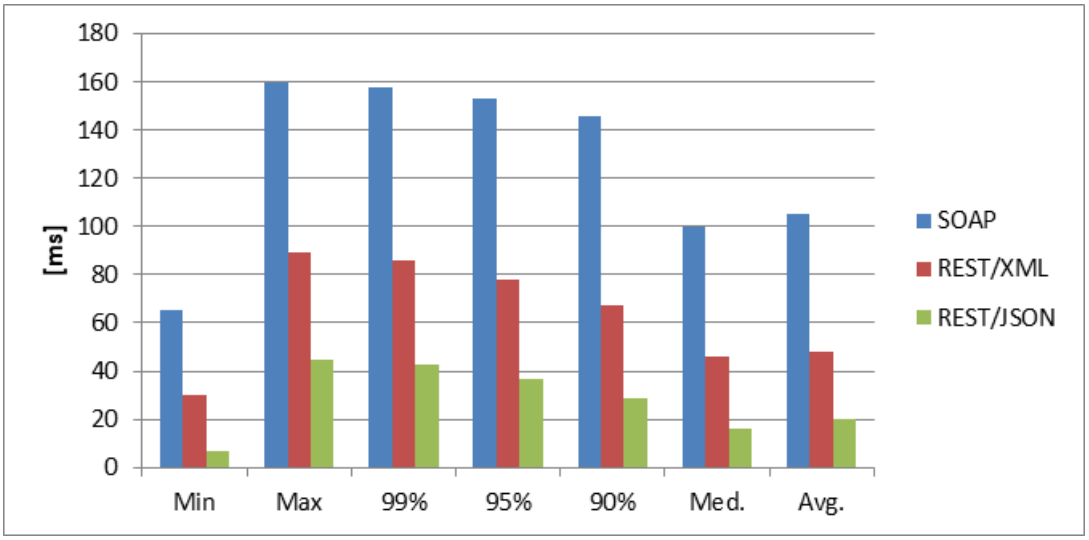

Fig. 2.8 Graficul agregat al rezultatelor testului pentru toate cele 3 grupuri

Semnificația indicatorilor folosiți în tabelul 2.6 și în figura 2.8 este:

- *Average* (*Avg.*): media aritmetică a timpilor de răspuns obținuți;
- *Median* (*Med.*): Valoarea timpului de răspuns pentru centila 50% (această valoare împarte mulțimea timpilor de răspuns în doua submulțimi egale);
- *90% Line*: Valoarea timpului de răspuns pentru centila 90% (90% dintre toți timpii de răspuns sunt mai mici sau egali cu această valoare);
- *95% Line*: Valoarea timpului de răspuns pentru centila 95%;
- *99% Line*: Valoarea timpului de răspuns pentru centila 99%;
- *Min*: Timpul minim de răspuns obținut;
- *Max*: Timpul maxim de răspuns obținut.

Graficul din figura 2.9 reprezintă răspunsul în timp pentru toate cele 3 grupuri.

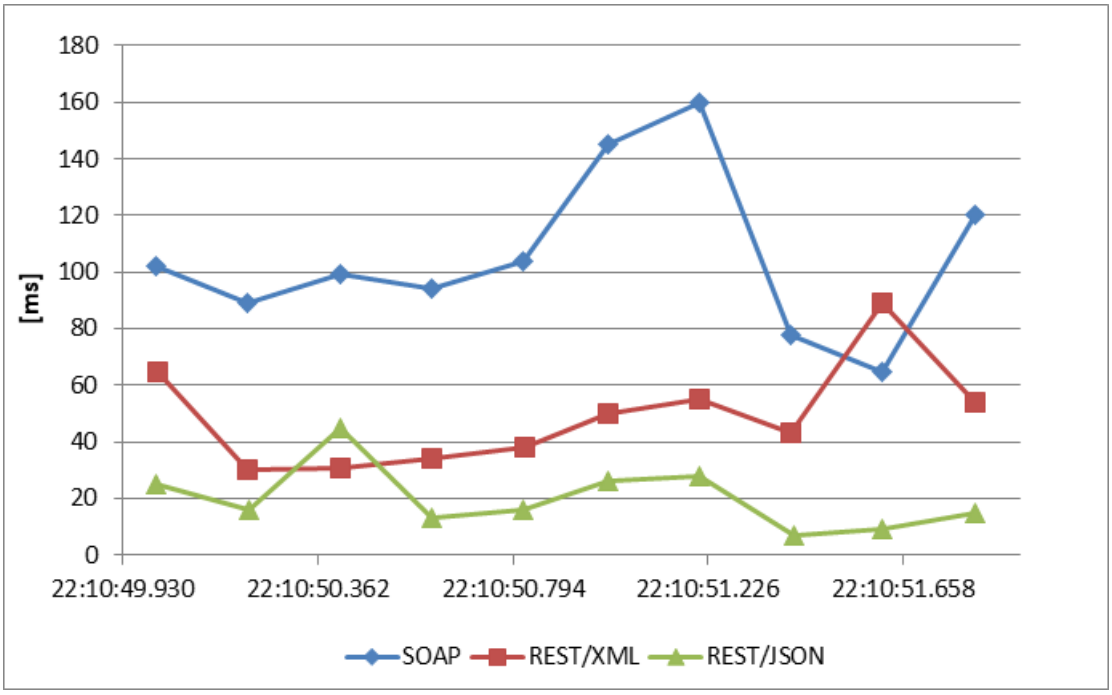

Fig. 2.9 Răspunsul în timp pentru cele 3 grupuri

În urma analizei se poate concluziona că din punct de vedere al performantei (respectiv al timpului de răspuns) pentru sistemele senzitive la context este indicată realizarea modulelor de comunicație folosind arhitectura *REST* / *JSON*.

Aceste concluzii sunt susținute și de literatura de specialitate. Astfel, în referințele [49], [50] și [51], serviciile *WEB RESTfull* împreună cu formatul *JSON* sunt considerate varianta optimă (*mobile-friendly*) din punct de vedere al timpului de răspuns pentru sistemele informaționale accesate prin intermediul terminalelor mobile. Totodată, conform cu rezultatele prezentate în referința [49], timpul de răspuns pentru procesarea mesajelor reprezentate în format *JSON* este mai scurt în raport cu alternativa *XML*.

2.2. Analiza metodelor pentru determinarea contextului

În cadrul prezentului subcapitol, autorul propune următoarele trei metode ce pot fi folosite pentru determinarea contextului:

- **MIBL** (Metoda Indirectă Bazată pe Localizare) care presupune determinarea contextului pe baza poziției geografice a utilizatorului obținută prin GPS și a teoremei cosinusului în triunghiurile sferice.
- **MDFC** (Metoda Directă Fără *Cache*) care implică determinarea contextului prin folosirea unor dispozitive de tip *beacon BLE* (prezentate în subcapitolul 1.1.3 al tezei de doctorat);
- **MDCC** (Metoda Directă Cu *Cache*) ce reprezintă o îmbunătățire a metodei MDFC prin utilizarea unei memorii locale de tip *cache* în cadrul aplicației client instalate pe terminalul mobil al utilizatorului. Rolul memoriei locale este acela de a diminua traficul în rețea prin prevenirea descărcării aceleași resurse de pe server de mai multe ori.

În cadrul prezentului subcapitol este prezentată și o analiză comparativă a celor trei metode pe fondul unui studiu de caz. Analiza realizată a fost axată pe următoare aspecte:

- integrarea metodelor propuse cu bazele de date relaționale;
- integrarea metodelor propuse cu serviciile *WEB* de tip *RESTfull*;
- comportamentul celor 3 metode din punct de vedere al traficului generat în rețea.

2.2.1. Metoda MIBL

Determinarea entităților aflate în vecinătatea utilizatorului pe baza informațiilor de localizare presupune următoarele. Cunoscând:

- poziția geografica a utilizatorului $U(Lat_U, Long_U)$ unde $Lat_U, Long_U \in \mathbb{R}$ și reprezintă coordonate de latitudine și longitudine exprimate în grade zecimale;
- pozițiile geografice ale tuturor entităților disponibile: $E(Lat_{Ei}, Long_{Ei})$ cu $Lat_{Ei}, Long_{Ei} \in \mathbb{R}$ (coordonate de latitudine și longitudine exprimate în grade zecimale) și $i = \overline{1 \dots n}$;
- vecinătatea definita prin suprafața geografica delimitata de cercul cu raza R (cunoscută) care are centrul dat de poziția utilizatorului: $V_U(U, R) = \{M(Lat_M, Long_M) | UM \leq R, Lat_M, Long_M, R \in \mathbb{R}\}$

trebuie determinate toate entitățile ale căror poziții E_i respectă condiția:

- $E_i \in V_U(U, R)$

respectiv entitățile care se află în vecinătatea V_U a utilizatorului.

Pe baza celor menționate anterior, metoda MIBL poate fi definită după cum urmează. Pentru fiecare moment t_i cu $i = \overline{0 \dots n}$, aplicația client instalată pe terminalul utilizatorului parcurge următoarea secvență:

1. determină poziția U_i a utilizatorului la momentul t_i folosind funcția de localizare a dispozitivului mobil (*GPS*);
2. apelează serviciul *WEB* de pe serverul sistemului furnizând parametrii U_i și valoarea predefinită R;

3. procesează răspunsul serverului pentru a extrage lista entităților ce reprezintă contextul utilizatorului la momentul t_i.

Corespunzător acestei secvențe se poate genera schema logică aferentă metodei MIBL, schemă ilustrată în figura 2.11.

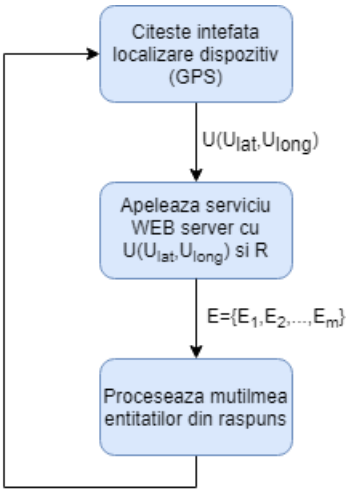

Fig. 2.11 Schemă logică MIBL în cadrul unei aplicații client

Sintetic, prin metoda MIBL, aplicația client va determină coordonatele utilizatorului folosind funcția de localizare a dispozitivului mobil (*GPS*). În vederea stabilirii contextului, aplicația client va realiza periodic cereri către server furnizând valorile parametrilor U_i și R. În ceea ce privește serverul, acesta va răspunde cu o listă ce va cuprinde entitățile (obiecte din lumea reală relevante pentru scopul utilizatorului) aflate în vecinătatea utilizatorului.

În verdea determinării entităților din vecinătatea utilizatorului prin metoda MIBL, se va utiliza teorema cosinusului în triunghiurile sferice.

2.2.2. Metoda MDFC

Pentru definirea metodei MDFC se poate porni de la aceleași ipoteze ca și în cazul MIBL, cu unele modificări detaliate în continuare.

Se consideră că obiectele din lumea reală au atașate dispozitive capabile să emită periodic un identificator unic prin intermediul unor semnale radio (de exemplu dispozitive de tip *beacon BLE*).

În ceea ce privește terminalul mobil (smartphone) al utilizatorului, se presupune că acesta are abilitatea de a detecta semnalele radio și de a identifica în mod unic orice dispozitiv / emițător.

Prin urmare, vecinătatea utilizatorului va depinde de distanța maximă de propagare a semnalului radio generat de emițător (cel atașat obiectului). În consecință, un obiect nu va putea fi detectat dacă se află la o distanță mai mare decât distanța maximă de propagare a semnalului generat de emițător. Sau, altfel spus, în această situație obiectul nu este considerat a fi în vecinătatea utilizatorului.

Ținând cont de cele menționate anterior, metoda MDFC poate fi formalizată după cum urmează. Pentru fiecare moment t_i cu $i = \overline{0 \dots n}$, clientul realizează următoarea secvență:

1. Determină identificatorii (*ID*) entităților aflate în vecinătatea utilizatorului la momentul t_i. Dacă nu este detectat nici un emițător (respectiv un *beacon*) atunci pașii 2 și 3 nu se mai execută;
2. Apelează serviciul *WEB* de pe serverul sistemului furnizând lista cu identificatorii (*ID*);
3. Procesează răspunsul serverului pentru a extrage lista entităților ce reprezintă contextul utilizatorului la momentul t_i.

Potrivit secvenței descrise mai sus, s-a elaborat schema logică pentru metoda MDFC prezentată în figura 2.12.

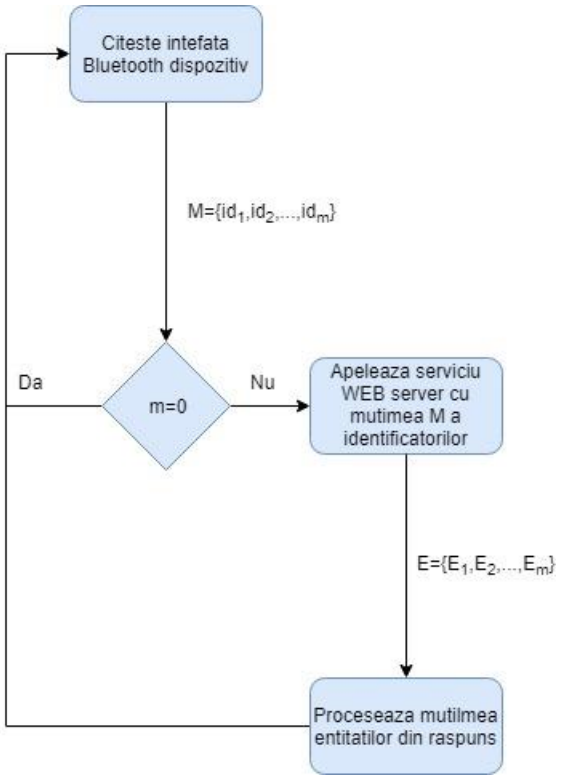

Fig. 2.12 Schemă logică MDFC în cadrul unei aplicații client

Din punct de vedere al integrării cu bazele de date relaționale, în cazul metodei MDFC, dispozitivele *beacon* trebuie configurate folosind valorile cheii primare din tabela în care sunt stocate datele entităților.

2.2.3. Metoda MDCC

Metoda MDDC reprezintă o variantă îmbunătățită a metodei MDFC. Diferențele între cele două metode se reflectă numai asupra modului de implementare al clienților, fără a fi necesare modificări asupra elementelor server.

Schema logică pentru metoda MDCC este prezentată în figura 2.13.

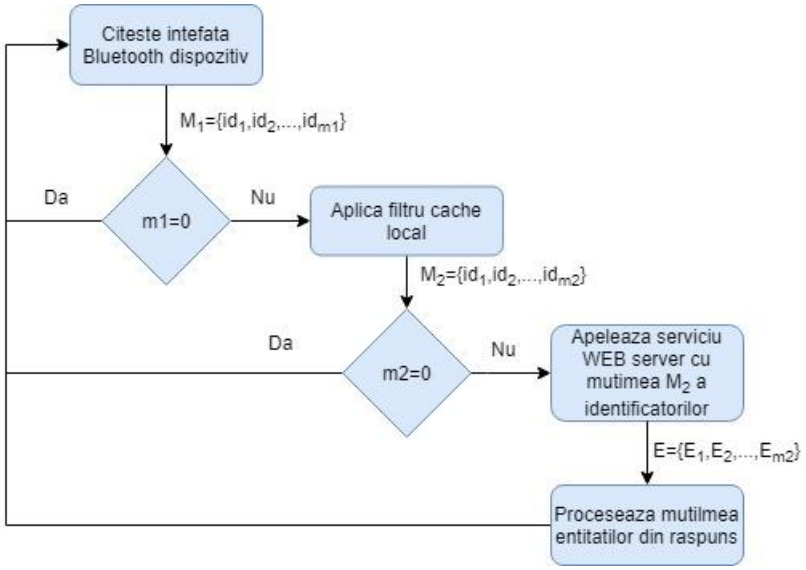

Fig. 2.13 Schemă logică MDCC în cadrul unei aplicații client

Utilizarea metodei MDDC poate conduce atât la reducerea numărului de apeluri cât și la diminuarea cantității de date vehiculate între client și server.

Astfel, în timpul unei sesiuni, prin metoda MDCC clientul poate stoca datele entităților obținute anterior de la server într-o memorie *cache* locală. În acest fel, pe durata sesiunii, date aferente unei entități anume nu vor fi transferate de mai multe ori (la momente de timp diferite ca răspuns la cereri diferite) dinspre server către client. De exemplu, dacă, la un anumit moment de timp, clientul detectează prezența unui emițător cu un *ID* existent deja în memoria *cache* locală atunci acesta nu mai este trimis ca parametru către server.

Ca o mențiune suplimentară, prin intermediul metodelor directe (MDFC și MDCC) pot fi tratate fără dificultate și situațiile care presupun entități mobile. Deoarece emițătorii sunt atașați fizic de obiecte, metodele cele două metode directe nu depind de cunoașterea pozițiilor entităților sau a utilizatorului.

2.2.4. Analiza posibilităţilor de integrare a metodelor propuse cu bazele de date relaţionale

În ceea ce privește posibilitățile de integrare cu bazele de date relaționale, au fost definite interogări *SQL* specifice metodelor propuse și a fost testat timpul de răspuns al acestora pentru o tabelă *MySQL* cu 1.000.000 de înregistrări. Rezultatele testelor, prezentate pe larg în cadrul subcapitolului 2.2.4 al tezei de doctorat, au indicat un timp de răspuns mult mai scurt în cazul metodelor directe (MDFC și MDCC).

2.2.5. Analiza posibilităţilor de integrare a metodelor propuse cu serviciile *WEB* de tip *RESTfull*

În vederea exemplificării modului de integrare a celor 3 metode cu serviciile *WEB*, a fost realizat un serviciu de tip *RESTfull* în limbajul de programare Java. În funcție de parametrii furnizați de către aplicațiile client, serviciul realizat poate fi folosit pentru toate cele 3 metode propuse.

2.2.6. Rezultate experimentale comparative obținute pentru cele 3 metode propuse

Acest subcapitol prezintă rezultatele experimentale obținute în urma unui studiu de caz axat pe determinarea impactului metodelor propuse (MIBL, MDFC, MDCC) asupra numărului de cereri și al cantității de date transferate între client și server.

Din considerente de spațiu, în cele ce urmează sunt evidențiate în mod sintetic rezultatele obținute în urma testului / simulărilor realizate. Mai multe detalii asupra acestor simulări sunt disponibile în cadrul tezei de doctorat.

Caracteristicilor dinamice obținute pe baza rezultatelor testului sunt prezentate în continuare.

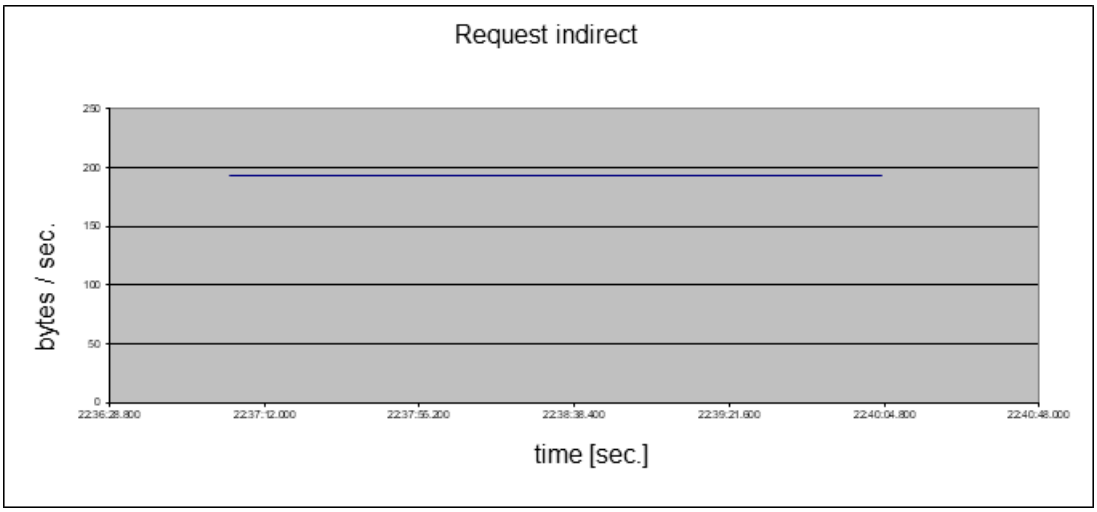

Fig. 2.20 Caracteristică dinamică trafic de date trimise în rețea pentru metoda MIBL

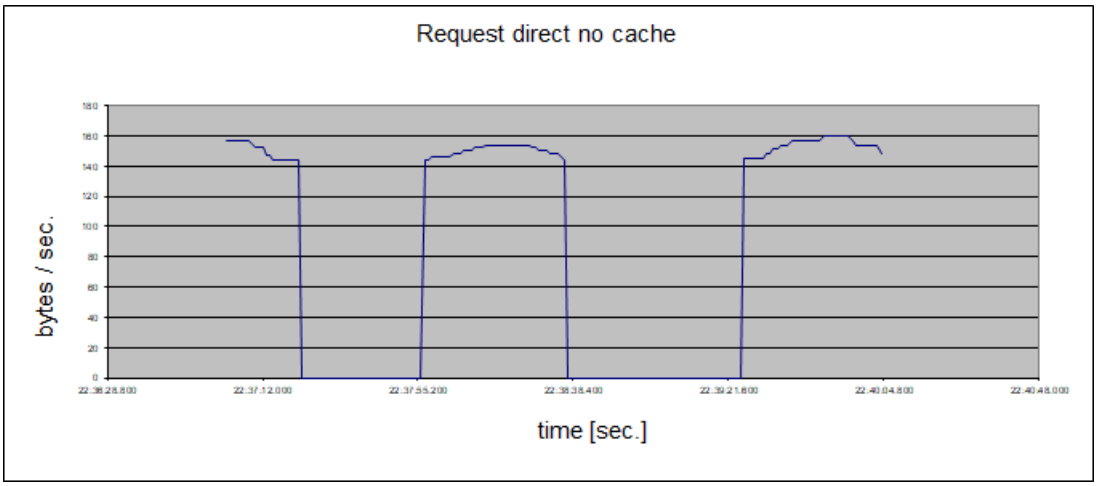

Fig. 2.21 Caracteristică dinamică trafic de date trimise în rețea pentru metoda MDFC

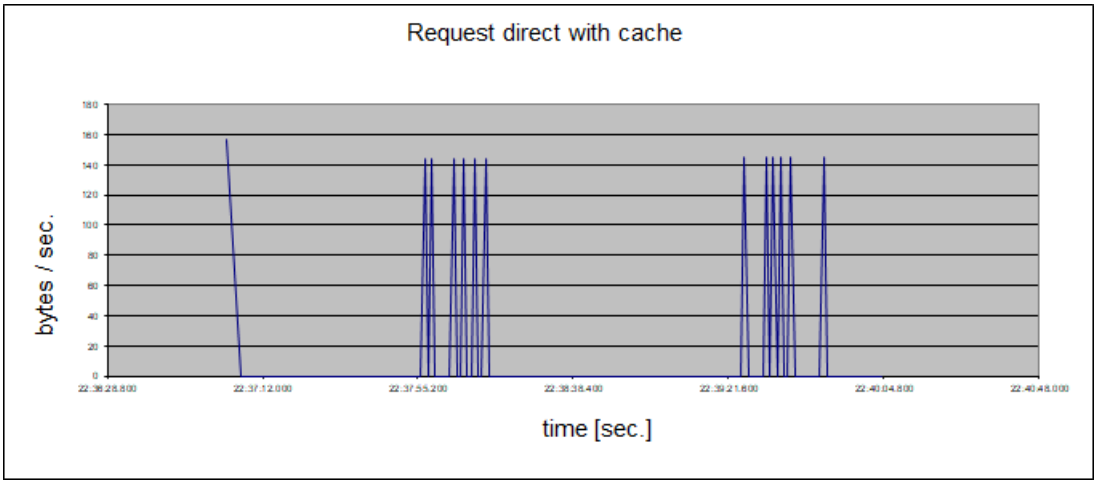

Fig. 2.22 Caracteristică dinamică trafic de date trimise în rețea pentru metoda MDCC

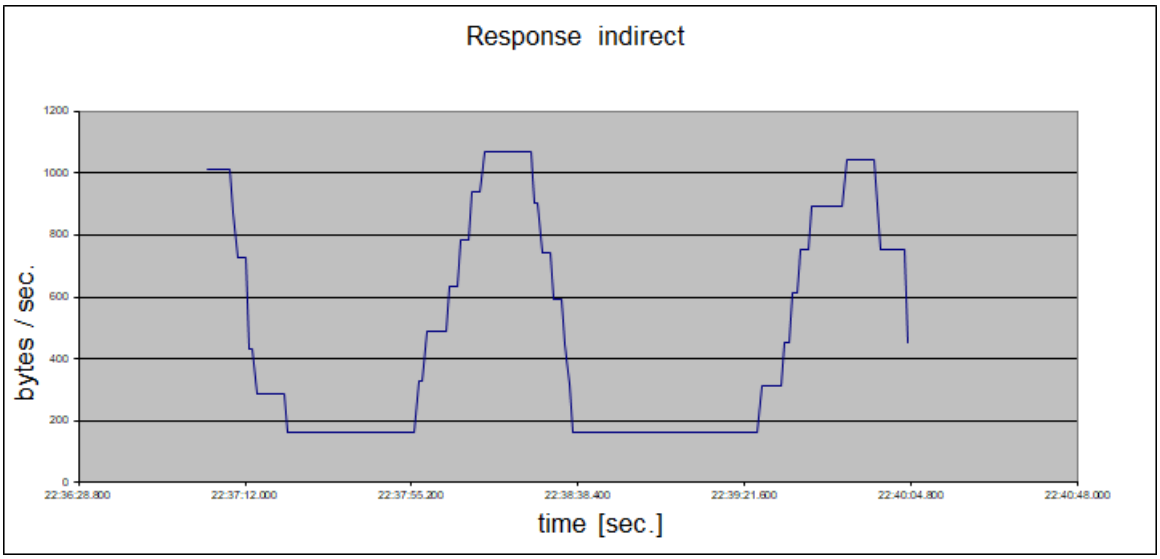

Fig. 2.23 Caracteristică dinamică trafic de date recepționate pentru metoda MIBL

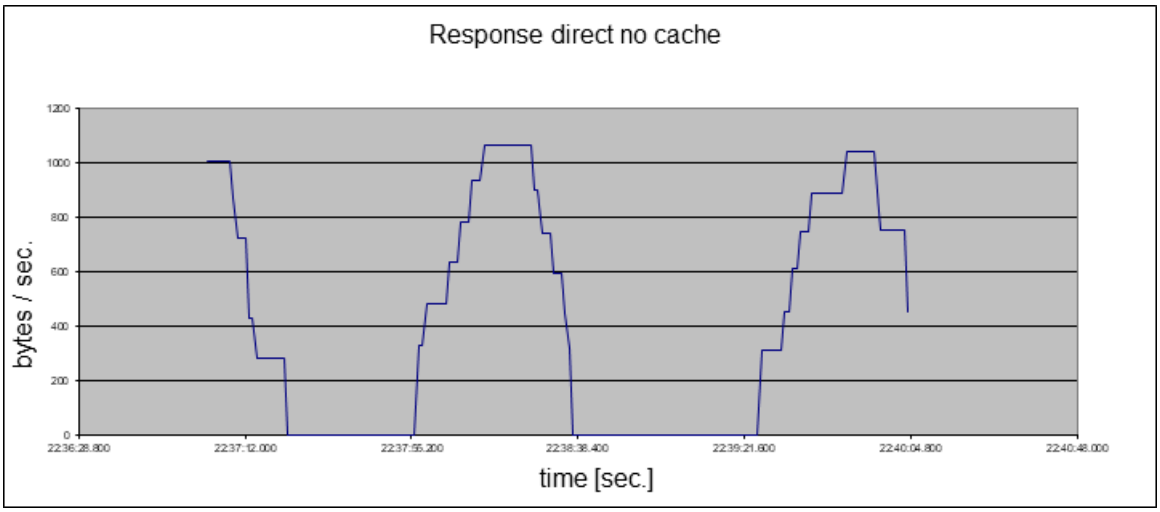

Fig. 2.24 Caracteristică dinamică trafic de date recepționate pentru metoda MDFC

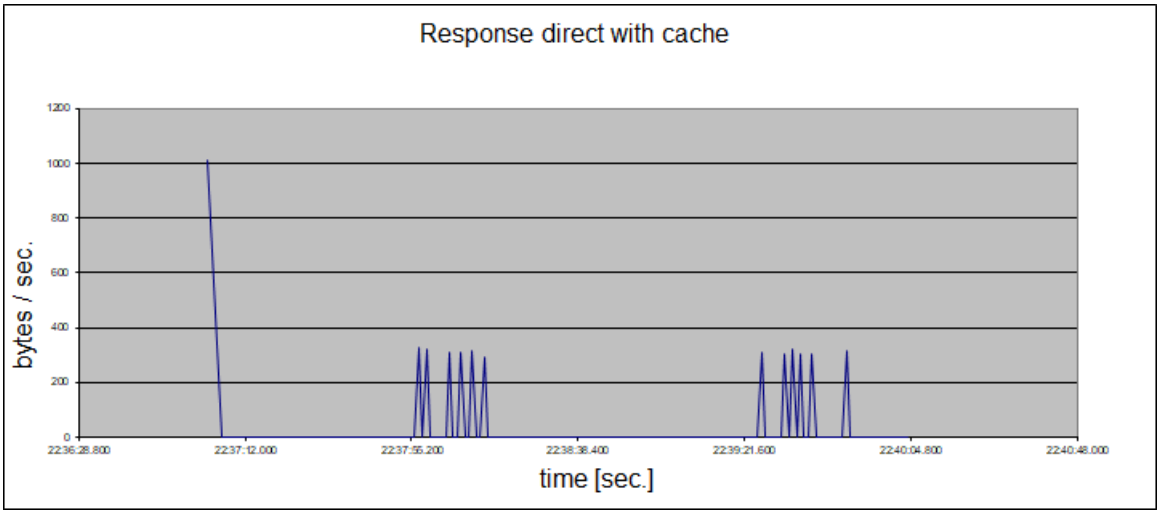

Fig. 2.25 Caracteristică dinamică trafic de date recepționate pentru metoda MDCC

Rezultatele centralizate obținute în urma testului sunt evidențiate în tabelul 2.13.

Tabel 2.13 Rezultatele centralizate ale testului

MIBL			MDFC			MDCC		
Număr de apeluri / cereri	*BYTES*		Număr de apeluri / cereri	*BYTES*		Număr de apeluri / cereri	*BYTES*	
	Trimiși	Primiți		Trimiși	Primiți		Trimiși	Primiți
180	34740	84179	97	14695	70650	13	1891	4762

Pe baza caracteristicilor din figurile 2.20, 2.21, 2.22, 2.23, 2.24, 2.25 și a datelor centralizate în tabelul 2.13 se pot trage următoarele concluzii:

- în cazul metodei MIBL pentru fiecare moment de timp t_i cu $i = \overline{1 \dots 180}$ a existat o cerere din partea clientului însoțită de un răspuns de la server;
- în cazul metodei MDFC numărul de cereri este mai mic față de MIBL. Pentru fiecare moment de timp t_i la care în vecinătatea utilizatorului nu există puncte de interes nu este transmisă nici o cerere către server;
- în cazul metodei MDCC numărul de cereri este mai mic față de MDFC. Acest lucru poate fi justificat prin utilizarea memoriei *cache* locale în cadrul clientului.

Așadar, din punct de vedere al numărului de cereri transmise către server cât și al traficului generat în rețea metoda MDCC este cea mai eficientă.

Totodată, trebuie menționat și faptul că pentru a onora o cerere din partea unui client serverul alocă anumite resurse de memorie și procesor. În cazul sistemelor senzitive la context, accesul clienților este concurențial resursele serverului fiind partajate între aceștia. Pe măsură ce numărul de clienți conectați simultan este mai mare și timpul de răspuns al serverului va crește. Drept urmare, din punct de vedere al optimizării este indicată utilizarea metodei MDCC.

2.3. Concluzii parţiale

În prima parte a capitolului sunt analizate trei variante de implementare ale serviciilor *WEB* în scopul determinării și selectării soluției optime cu privire la realizarea componentelor și modulelor de comunicație în cadrul demonstratorului. Cele trei variante luate în considerare sunt:

- servicii *WEB SOAP*;
- servicii *WEB RESTful* / *XML* (cu transferul datelor în format *XML*);
- servicii *WEB RESTful* / *JSON* (cu transferul datelor în format *JSON*).

Analiza performanțelor acestor variante este realizată pe fondul unui studiu de caz pentru un sistem care furnizează informații relevante utilizatorului în funcție de poziția geografica a acestuia (sistem de tipul *Point-of-Interest Information*).

Pentru realizarea studiului de caz au fost implementate următoarele două servicii *WEB* :

- un serviciu *SOAP*;
- un serviciu *RESTfull* capabil să transfere date reprezentate atât în formatul *JSON* cât și *XML*.

Cele două servicii WEB au fost folosite în cadrul unui experiment. Pe durata experimentului au simulate cereri din partea unor aplicații client prin intermediul sistemului *JMeter* și au fost monitorizați timpii de răspuns pentru fiecare dintre cele trei variante (*SOAP*, *RESTful / XML*, *RESTful / JSON*).

În urma efectuării experimentului, timpii de răspuns cei mai scurți au fost obținuți în cazul variantei *RESTfull / JSON*. Drept urmare, această variantă a fost aleasă pentru realizarea componentelor și modulelor de comunicație în cadrul demonstratorului.

În cea de a doua parte a capitolului, au fost propuse trei metode ce pot fi folosite pentru determinarea contextului și anume:

- MIBL (o metodă indirectă bazată pe datele de localizare ale utilizatorului obținute prin *GPS*);
- MDFC (o metodă directă bazată pe utilizarea de dispozitive *beacon BLE*);
- MDCC (o variantă îmbunătățită a MDFC bazată pe utilizarea unei memorii locale de tip *cache* în cadrul aplicației client instalate pe terminalul mobil al utilizatorului).

Fiecare dintre cele 3 metode este definită prin intermediul unei scheme logice.

Ulterior, sunt analizate posibilitățile de integrare a celor 3 metode cu bazele de date relaționale și cu serviciile *WEB* de tip *RESTfull*.

Referitor la bazele de date relaționale sunt definite interogări *SQL* specifice metodelor propuse și este testat timpul de răspuns al acestora pentru o tabelă cu 1.000.000 de înregistrări. Rezultatele testelor au indicat un timp de răspuns mult mai scurt în cazul metodelor directe (MDFC și MDCC).

În vederea exemplificării modului de integrare a celor 3 metode cu serviciile *WEB*, a fost realizat un serviciu de tip *RESTfull*. În funcție de parametrii furnizați de către aplicațiile client, serviciul realizat poate fi folosit pentru toate cele 3 metode propuse.

De asemenea, a fost realizat și un studiu de caz axat pe determinarea impactul metodelor propuse (MIBL, MDFC, MDCC) asupra numărului de cereri și al cantității de date transferate între client și server. Rezultatele studiului de caz au condus la concluzia că din punct de vedere al numărului de cereri transmise către server cât și al traficului generat în rețea metoda MDCC este cea mai eficientă. Drept urmare, aceasta metodă a fost selectată spre a fi folosită în cadrul demonstratorului.

Capitolul 3. Contribuții privind proiectarea și realizarea modulelor server ale unui sistem expert senzitiv la context pentru obiecte de patrimoniu

În cadrul tezei de doctorat, autorul formulează un concept nou, aplicabil în domeniul patrimoniului cultural. Totodată, autorul transpune în practică conceptul propus prin realizarea unui sistem experimental denumit în continuare și demonstrator.

Demonstratorul realizat este un sistem expert senzitiv la context, tranzacțional, distribuit și cu o arhitectură de tip client-server.

În prezentul capitol sunt tratate probleme ce țin de proiectarea și realizarea componentelor server ale demonstratorului.

Pentru început, în subcapitolul 3.1, este definit termenul demonstrator în raport cu obiectivele tezei de doctorat. Totodată sunt evidențiate atât caracteristice generale ale sistemelor expert, cât și modul de implementarea a acestora în cadrul demonstratorului realizat.

Subcapitolul 3.2 este alocat prezentării conceptul care stă la baza demonstratorului. Tot în cadrul acestui subcapitol sunt introduse și explicate anumite noțiuni relevante pentru demonstrator precum: obiecte de patrimoniu și patrimoniu cultural.

Ulterior, în subcapitolul 3.3, autorul propune arhitectura software a sistemului experimental.

Subcapitolul 3.4 include contribuțiile autorului privind proiectarea și realizarea componentelor server ale demonstratorului în conformitate cu arhitectura software definită anterior.

În final, subcapitolul 3.5 evidențiază concluziile parțiale corespunzătoare capitolului 3.

Aspectele referitoare proiectarea și realizarea părții de client a demonstratorului sunt abordate în capitolul 4.

3.1. Corelaţia dintre demonstrator şi sistemul expert senzitiv la context pentru obiecte de patrimoniu

Obiectivele tezei de doctorat au fost enunțate în introducere. Acestea sunt:

O1. Realizarea de cercetări cu privire la posibilitatea dezvoltării unui sistem expert senzitiv la context, destinat furnizării de conținut interactiv turiștilor pe durata vizitelor la siturile culturale.

O2. Transpunerea în practică a rezultatelor aferente obiectivului O1 prin realizarea unui sistem experimental, respectiv a unui demonstrator pentru sistemul expert senzitiv la context propus.

În raport cu cele două obiective menționate mai sus, termenul **demonstrator** se referă la un sistem expert senzitiv la context, a cărui funcționare a fost validată în condiții de laborator. Așadar, demonstratorul realizat este un sistem experimental care integrează toate componentele și a cărui configurație este similară în aproape toate aspectele cu cea a prototipului sau a sistemului final (respectiv a unui sistem expert senzitiv la context implementat la un muzeu).

Deoarece demonstratorul realizat este un sistem expert acesta înglobează anumite elemente specifice sistemelor de acest tip.

Schema ilustrată în figura 3.1, preluată și adaptată din referințele [20] și [22], evidențiază arhitectura generală a unui sistem expert. Această arhitectură a reprezentat punctul de plecare în activitatea de proiectare a demonstratorului.

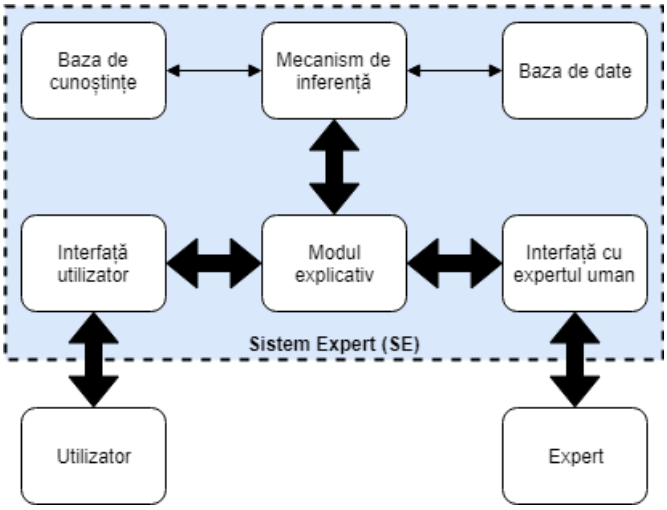

Fig. 3.1. Arhitectura generală a unui sistem expert [20] [22]

Conform figurii 3.1 și referințelor [20] și [22], arhitectura generală a unui SE (sistem expert) integrează următoarele elemente: mecanismul de inferenţă, baza de cunoştinţe, baza de date, modulul explicativ, interfaţa cu utilizatorul şi interfaţa cu expertul (inginerul de cunoştinţe).

Noțiuni precum motor de inferență, bază de cunoștințe, raționament, regulă și faptă au fost definite în cadrul subcapitolului 1.2 al tezei de doctorat. Acestea sunt folosite în cele ce urmează pentru descrierea elementelor din arhitectura generală a unui sistem expert evidențiată în figura 3.1.

- **Mecanismul de inferență** (respectiv motorul de inferență) reprezintă elementul principal din structura unui sistem expert. Acesta realizează inferențe (respectiv raționamente) și ia decizii pe baza regulilor [20] [22].
- **Baza de cunoștințe** conține regulile definite de către experți [20] [22].
- **Baza de date** poate conține fapte utilizate în cadrul regulilor [20] [22], rezultatele inferențelor (date rezultate în urma raționamentelor) și alte informații necesare funcționarii SE.
- **Modulul explicativ** permite justificarea raționamentului [20] [22] (succesiunea regulilor aplicate care a condus la o anumită concluzie).
- **Interfața cu utilizatorul** permite prezentarea concluziilor (rezultatelor raționamentului).
- **Interfața cu expertul uman** permite achiziția de cunoștințe [20] [22]. Prin intermediul acestei interfețe expertul uman poate adăuga reguli în baza de cunoștințe. Regulile sunt folosite ulterior de către mecanismul de inferență pentru a efectua raționamente.

Corelația dintre arhitectura demonstratorului realizat și schema generală a unui sistem expert este ilustrată în figura 3.2.

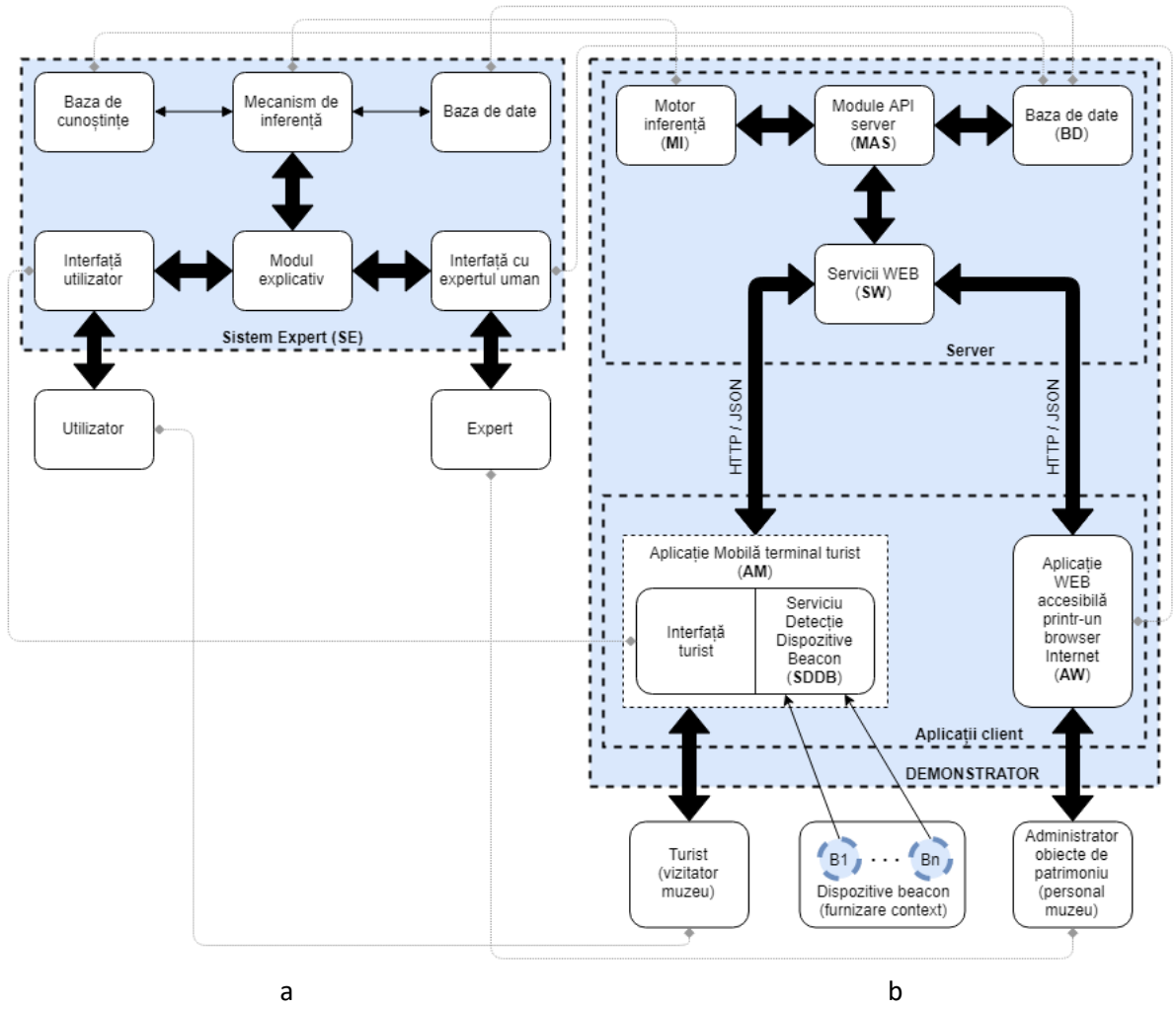

Fig. 3.2 Corelația dintre arhitectura unui sistem expert (a) și arhitectura demonstratorului (b)

După cum reiese și din figura 3.2, pe lângă elementele specifice arhitecturii unui sistem expert, demonstratorul integrează și componente relevante pentru îndeplinirea proprietății de senzitivitate la context. Totodată, demonstratorul combină arhitectura caracteristică sistemelor expert cu cea de tip client – server.

Corelația demonstrator – sistem expert este prezentată sintetic în tabelul 3.1. Mai multe detalii despre arhitectura demonstratorului și componentele acestuia sunt furnizate în subcapitolul 3.3 al prezentului capitol.

Tabel 3.1 Corelația demonstrator – sistem expert

SE	Demonstrator	Observații
Mecanism de inferență	Motor de inferență (**MI**)	Motorul de inferența folosit în cadrul demonstratorului este *Drools Expert*. Acesta este responsabil pentru realizarea raționamentelor în cadrul sistemului experimental.
Bază de cunoștințe	Bază de date (**BD**)	Cunoștințele sunt reprezentate prin intermediul regulilor de producție fiind stocate în cadrul bazei de date a demonstratorului. Pe lângă regulile de producție, baza de date conține atât informații necesare pentru funcționarea demonstratorului (de exemplu date despre utilizatori, obiecte
Bază de date		

		de patrimoniu etc.) cât și rezultatele inferențelor.
Modul explicativ	-	Motorul *Drools Expert* include un modul explicativ însă acesta nu este folosit în cadrul demonstratorului deoarece nu este necesar (respectiv se consideră că un turist nu va solicita explicarea raționamentului).
Interfață utilizator	Aplicație mobilă (**AM**)	Aplicația mobilă conține două elemente: - Interfața cu turistul care permite vizualizarea rezultatelor inferențelor; - Serviciul SDDB (Serviciu Detecție Dispozitive *Beacon*) necesar pentru îndeplinirea proprietății de senzitivitate la context a demonstratorului.
Interfață cu expertul uman	Aplicație *WEB* (**AW**)	Aplicația *WEB* reprezintă interfața expertului uman (administrator obiecte de patrimoniu) cu demonstratorul. Prin intermediul acestei aplicații sunt definite obiectele de patrimoniu și regulile de producție.
-	Servicii *WEB* (**SW**)	Demonstratorul este un sistem distribuit cu arhitectură de tip client – server. Aplicațiile client rulează local pe dispozitivele utilizatorilor (terminal turist și *browser* Internet). Comunicația între aplicațiile client și serverul demonstratorului se realizează prin intermediul unor servicii *WEB* de tip *RESTfull*.
-	Module *API* Server (**MAS**)	Modulele *API* server includ componentele necesare pentru: - Integrarea motorului de inferență în cadrul demonstratorului; - Integrarea bazei de date în cadrul demonstratorului (accesul la date); - Autentificarea utilizatorilor celor două aplicații (**AM** și **AW**).

3.2. Conceptul ce stă baza demonstratorului

Noțiunea de obiect de patrimoniu se regăsește chiar în titlul tezei de doctorat. Mai mult decât atât, conceptul propus este aplicabil în domeniul patrimoniului cultural. Drept urmare, cele două noțiuni, **obiect de patrimoniu** și **patrimoniu cultural**, sunt relevante pentru demonstrator fiind explicate în cele ce urmează.

Prin patrimoniul cultural înțelegem reprezentarea modului de viața al unei comunități, transmis din generație în generație, incluzând obiceiuri, practici, locuri și obiecte, expresii artistice și valori [1].

În ceea ce privește obiectele de patrimoniu, conform referinței [1], acestea se împart în 3 categorii și anume:

- medii artificiale (clădiri, ruine, vestigii arheologice etc.);
- medii naturale (rezervații naturale etc.);
- artefacte (cărți, documente, tablouri etc.).

Figura 3.3, preluata și adaptată din referința [1], prezintă în mod sugestiv elementele patrimoniului cultural.

Fig. 3.3 Elementele patrimoniului cultural

Un obiectiv principal în ceea ce privește managementul patrimoniului cultural constă în comunicarea semnificației și a necesității de prezervare a acestuia, atât membrilor comunității gazde, cât și vizitatorilor străini [2]. Patrimoniu cultural aparține tuturor, fiecare dintre noi având dreptul și responsabilitatea de a înțelege, aprecia și conserva valorile lui universale [2].

Pornind de la acest deziderat, se poate considera că o vizită la un sit cultural reprezintă un proces de asimilare pentru turist. În timpul vizitei, turistul își însușește informații relevante despre obiceiurile, practicile, locurile, obiectele, expresiile artistice și valorile relaționate cu respectivul sit cultural. Din această perspectiva, au fost identificați mai mulți factori ce pot influența în mod negativ procesul de asimilare a informațiilor de către turist. Astfel de factori pot fi: personalitatea, experiența, preferințele, interesele și starea de spirit ale turistului [3]. Analiza literaturii de specialitate sugerează faptul că un proces de asimilare / învățare interactiv și adaptiv poate conduce la un nivel crescut de conștientizare și la un interes sporit din partea turistului.

Sistemul experimental dezvoltat de către autor și prezentat în teza de doctorat primite crearea de asocieri virtuale între personaje digitale (*avatar*) și obiecte de patrimoniu. Fiecare obiect de patrimoniu din lumea reală poate avea un corespondent în cadrul sistemului. Totodată, pentru fiecare obiect de patrimoniu definit în sistem vor exista și unul sau mai multe personaje digitale asociate cu acesta.

Demonstratorul se bazează pe conceptul interacțiunilor virtuale între turiști și personajele digitale. Vizitatorii unui sit cultural nu sunt simpli spectatori. Aceștia vor putea sa *comunice* cu personajele digitale prin intermediul sistemului. În cadrul demonstratorului, un personaj digital are rolul de a comunica turiștilor informații despre obiectul de patrimoniu căruia îi este asociat.

Pentru a-și îndeplini rolul, personajele digitale prind *viață* pe terminalele de tip *smartphone* ale turiștilor și poartă conversații cu aceștia. Conversațiile sunt adaptive în sensul că pot evolua diferit în funcție de interacțiunea dintre *avatar* (respectiv personajul digital) și turist. Stimulul care declanșează interacțiunea depinde de contextul utilizatorului, sau altfel spus, de proximitatea vizitatorului față de un anumit obiect de patrimoniu la un moment dat.

După cum s-a precizat la începutul acestui capitol, demonstratorul este un sistem senzitiv la context. Contextul reprezintă totalitatea informațiilor necesare pentru a defini circumstanțele

utilizatorului [4]. Acesta include ansamblul elementelor din lumea reală care sunt relevante pentru scopul utilizatorului la un moment dat.

Scopul utilizatorului (turistului) în raport cu demonstratorul, este acela de a obține informații despre obiectele de patrimoniu. În consecință, elementele contextului în cazul demonstratorului sunt chiar obiectele de patrimoniu.

Demonstratorul este un **sistem automat**. Acesta monitorizează contextul utilizatorului pe durata unei vizite la un sit cultural. Ca urmare a modificărilor contextului, personaje digitale inițializează **automat** conversații cu utilizatorul pentru a-i transmite informații despre obiectele de patrimoniu.

După cum s-a mai menționat, în cadrul sistemului experimental, utilizatorii poartă conversații cu personaje digitale. Conversațiile evoluează în baza anumitor scenarii predefinite care includ cunoștințele unor experți. Experții pot fi chiar din rândul personalului care administrează situl cultural și obiectele de patrimoniu aferente acestuia. Relația dintre obiectele de patrimoniu, scenarii și conversații este explicată în subcapitolul 3.4.1 al tezei de doctorat.

În timp ce când folosesc demonstratorul, turiștii sunt implicați într-un proces de învățare. Aceștia asimilează informații despre obiectele de patrimoniu. Pentru a obține un interes sporit din partea utilizatorilor, procesul trebuie să fie unul interactiv. În acest scop este utilizat un motor de inferență. Prin urmare, demonstratorul este un **sistem bazat pe cunoștințe** (sau **sistem expert**), motorul de inferență fiind modulul din componență acestuia care permite furnizarea de informații în mod interactiv turiștilor. Noțiunile de sistem expert și motor de inferență se află în strânsă legătură și au fost definite în subcapitolul 1.2 al tezei de doctorat, fiind parțial reluate și la începutul acestui capitol.

Principiul de funcționare al demonstratorului este reprezentat sugestiv în figura 3.4.

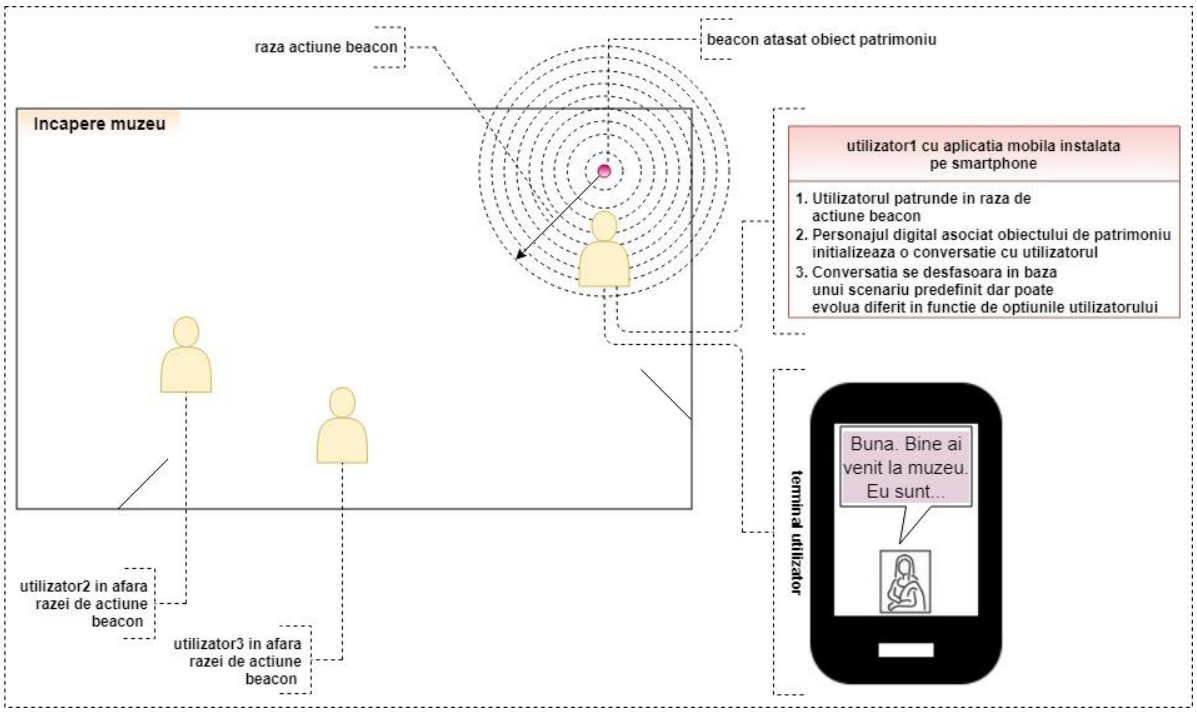

Fig. 3.4 Ilustrarea principiului de funcționare al demonstratorului

Conform figurii 3.4, principiul de funcționare al demonstratorului presupune determinarea contextului folosind dispozitive de tip *beacon*. Noțiunea de *beacon BLE* a fost definită în subcapitolul 1.1.3 al tezei de doctorat.

Pentru determinarea contextului este aplicată metoda MDCC (metoda directă cu *cache*) ale cărei caracteristici au fost definite în capitolul 2 al tezei de doctorat (subcapitolul 2.2).

3.3. Arhitectura software a demonstratorului

Modulele esențiale pentru funcționarea demonstratorului au fost evidențiate în subcapitolul 3.1 (figura 3.2 și tabelul 3.1). O parte dintre aceste module au fost menționate și în prezentarea conceptului demonstratorului (subcapitolul 3.2).

După cum s-a arătat în subcapitolul 3.1, demonstratorul combină arhitectura caracteristică sistemelor expert cu cea de tip client – server integrând totodată și componente relevante pentru îndeplinirea proprietății de senzitivitate la context.

Figura 3.5, evidențiază arhitectura demonstratorului (ilustrată anterior și în figura 3.2 pentru relevarea corelației demonstrator – sistem expert) care include 6 module software (două client și 4 server).

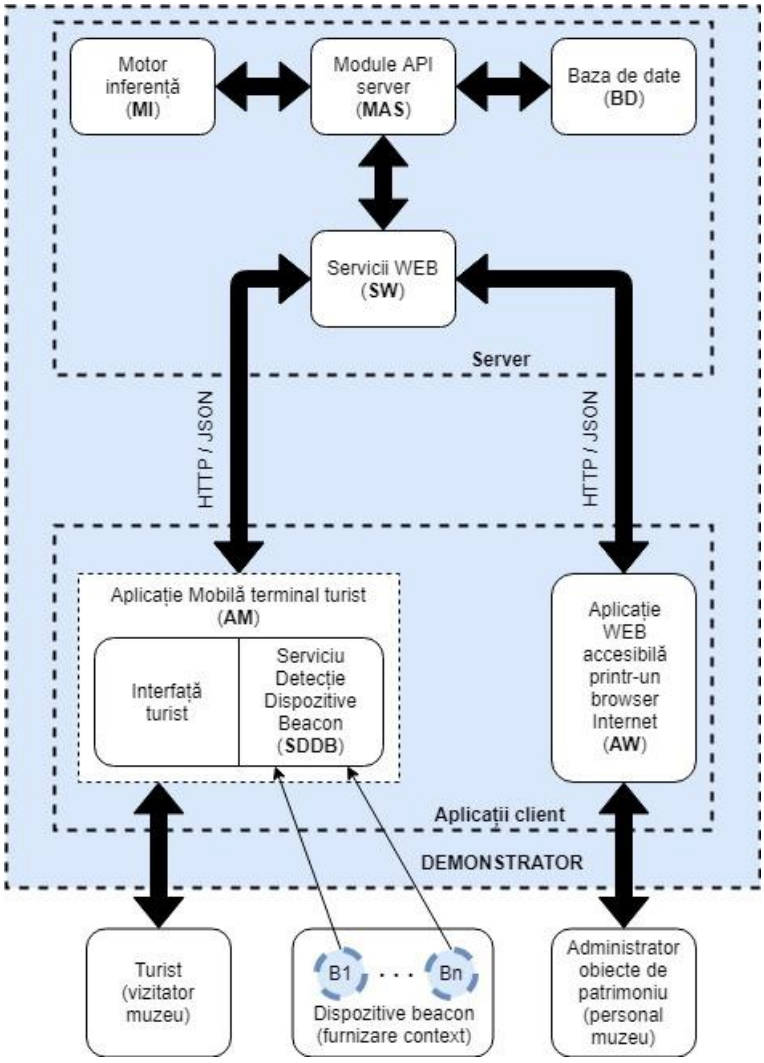

Fig. 3.5 Ilustrarea arhitecturii software a demonstratorului

În continuare sunt prezentate câteva detalii generice despre elementele din componența sistemului experimental ilustrate în figura 3.5.

- **Serverul demonstratorului** integrează următoarele 4 module evidențiate și în figura 3.5: baza de date (BD), modulele *API* server (MAS), motorul de inferență (MI) și serviciile *WEB* (SW).

❖ **Baza de date** (elementul BD din arhitectura demonstratorului reprezentată în figura 3.5) este unul dintre modulele server ale demonstratorului. Aceasta conține toate datele disponibile în cadrul sistemului experimental. Datele stocate aici corespund unor elemente precum: utilizatori, obiecte de patrimoniu, scenarii, conversații, mesaje etc. Demonstratorul folosește o bază de date relațională. Este de menționat faptul că regulile de producție sunt asimilate scenariilor iar mesajele (grupate în conversații) reprezintă rezultatul inferențelor. Mai multe detalii despre modulul BD sunt furnizate în subcapitolul 3.4.1.

❖ **Elementul MAS** (Module *API* Server) din arhitectura demonstratorului (reprezentată în figura 3.5), include toate componentele care asigură funcționalitatea serverului sistemului experimental. Noțiunea de componentă (sau componentă *EJB*) are sensul caracteristic platformei *Java EE* (*Enterprise Edition*) care a fost folosită pentru realizarea parții de server a demonstratorului.

Specificația *JSR-318* definește arhitectura *EJB* (*Enterprise JavaBeans*) aferentă aplicațiilor compatibile cu platforma *Java EE* [65]. Conform specificației *JSR-318*, arhitectura *EJB* este destinată aplicațiilor bazate pe componente (*component-based*) [65].

Sistemele și aplicațiile dezvoltate pe baza arhitecturii *EJB* sunt scalabile (*scalable*), tranzacționale (*transactional*) și accesibile unui număr mare de utilizatori în condiții de securitate (*multi-user secure*) [65].

Modulele API server (MAS) includ:

- domeniul demonstratorului;
- componenta de acces la cunoaștere;
- componentele de acces la date;
- componenta autentificare a utilizatorilor.

❖ **Motorul de inferență** (elementul MI din arhitectura demonstratorului reprezentată în figura 3.5) reprezintă un alt modul server al sistemului experimental. Conform conceptului enunțat în subcapitolul 3.2, acest modul este necesar pentru a conferi un caracter interactiv procesului de învățare în care sunt implicați turiștii. În cadrul demonstratorului este utilizat motorul de inferență *Drools Expert* împreună cu modelul regulilor de producție (pentru reprezentarea cunoștințelor). Motorul *Drools* este un modul independent. Pentru interacțiunea cu acesta, modulule MAS includ componenta de acces la cunoaștere.

❖ **Serviciile *WEB*** (blocul cu notația SW din arhitectura demonstratorului reprezentată în figura 3.5) ale demonstratorului asigură comunicația între aplicațiile client și server. Toate serviciile *WEB* din cadrul sistemului experimental sunt de tip *RESTful*. Acestea folosesc formatul *JSON* pentru reprezentarea resurselor și protocolul *HTTP* pentru transferul acestora în rețea.

➢ **Demonstratorul include două aplicații client**: aplicația mobilă (AM) și aplicația *WEB* (AW).

❖ **Aplicația mobilă** (blocul cu notația AM din arhitectura demonstratorului reprezentată în figura 3.5) este compatibilă cu sistemul de operare *Android* și reprezintă un modul client a demonstratorului. Demonstratorul este un sistem senzitiv la context. Din acest punct de vedere, aplicația mobilă joacă un rol esențial în determinarea contextului turiștilor. Aceasta include serviciul SDDB (reprezentat în figura 3.5) care folosește interfața *Bluetooth* a terminalului mobil pentru a detecta semnalele radio emise de către dispozitivele *beacon* atașate obiectelor de patrimoniu. Contextului utilizatorilor este determinat prin metoda MDCC prezentată în subcapitolul 2.2 al tezei de doctorat. Totodată, aplicația mobilă reprezintă și interfața utilizator care permite schimbul de mesaje între personajele digitale și turiști. În cadrul demonstratorului, mesajele reprezintă rezultatul inferențelor (sunt create în timpul execuției scenariilor ca urmare a declanșării regulilor) și sunt grupate în conversații.

❖ **Aplicația WEB** (elementul AW din arhitectura demonstratorului reprezentată în figura 3.5) rulează în *browser*, se bazează pe limbajele *Java*, *JavaScript* și *HTML* și reprezintă interfața dintre sistemul experimental și administratorii obiectelor de patrimoniu (experții umani). Aceasta permite definirea obiectelor de patrimoniu și a scenariilor în cadrul demonstratorului. Scenariile conțin regulile de producție folosite de către motorul de inferență.

Cu excepția aplicației *WEB*, a motorului de inferență (care reprezintă un modul independent) și a bazei de date (pentru care sa folosit *MySQL*), limbajul de programare adoptat la realizarea modulelor software ale demonstratorului este *Java*.

3.4. Modulele și componente server ale demonstratorului

Aceste subcapitol este consacrat prezentării unor aspecte relevante cu privire la proiectarea și realizarea modulelor și componentelor server ale demonstratorului.

3.4.1. Baza de date

Baza de date a demonstratorului este alcătuită din următoarele 7 tabele:

- tabela `heritage_objects` (stochează date despre **obiectele de patrimoniu**);
- tabela `scenarios` (stochează **regulile de producție** folosite de către motorul de inferență și date despre **personajele digitale**);
- tabela `conversations` (stochează **datele aferente conversațiilor** dintre personajele digitale și turiști);
- tabela `messages` (stochează **mesajele** create în timpul conversațiilor, mesaje care reprezintă rezultatele inferențelor);
- tabela `users` (stochează **datele utilizatorilor** care sunt necesare și în procesul de autentificare / autorizare);
- tabela `user_groups` (necesară pentru **autentificarea / autorizarea utilizatorilor**);
- tabela `user_images` (stochează **imaginile turiștilor** folosite în cadrul aplicației mobile).

Cele 7 tabele sunt reprezentate și în diagrama ER din figura 3.6 care reprezintă modelul relațional al demonstratorului.

Fig. 3.6 Diagrama modelului relațional al demonstratorului

Serverul de baze de date folosit în cadrul demonstratorului este *MySQL* (versiunea 5.7.16, 64 biți).

3.4.2. Modulele *API* server

Modulele *API* server, abreviate prin notația MAS în figura 3.5 (arhitectura demonstratorului) includ următoarele elemente:

- domeniul demonstratorului;
- componenta de acces la cunoaștere;
- componentele de acces la date;
- componenta autentificare a utilizatorilor.

Acestea sunt prezentate în subcapitolele următoare.

3.4.2.1. Domeniul demonstratorului

Prin domeniu se înțelege ansamblul tuturor elementelor necesare pentru a definii problema care trebuie rezolvată. În cazul demonstratorului, câteva exemple de elemente din domeniul problemei pot fi: **obiectele de patrimoniu**, **turiștii**, **conversațiile**, **mesajele** etc. Toate aceste elemente sunt reprezentate prin clase în cadrul domeniului demonstratorului.

Domeniul demonstratorului poate fi divizat în doua părți:

- **domeniul persistent** (include toate clasele care permit salvarea stării în baza de date);
- **domeniul cunoașterii** (include toate clasele folosite pentru definirea regulilor cu care lucrează motorul de inferență).

A. Domeniul persistent

În ceea ce privește domeniul persistent, fiecare clasă care intră în componența acestuia corespunde unei tabele din cadrul bazei de date. Cu alte cuvinte, starea instanțelor (obiectelor) unei clase din domeniul persistent poate fi salvată în baza de date. În cadrul demonstratorului, instanțele claselor domeniului persistent sunt entități JPA (*Java Persistence API*).

Prin intermediul *JPA*, o clasa *Java* obișnuită poate fi însoțită de metadate[45] [66] [67] [68] [69] [70] [71]. Metadatele descriu modul în care instanțele clasei pot fi salvate în baza de date. Acestea specifică corespondența între atributele claselor și câmpurile tabelelor. Procesul prin care este definită această corespondența poartă numele de mapare *ORM* (*Object Relational Mapping*). Conform referințelor [45], [66] și [67], tipurile de date definite în acest fel sunt denumite clase entități (*entity classes*), în timp ce instanțele acestora poartă denumirea de entități *JPA* (*JPA entities*).

Domeniul persistent al demonstratorului este alcătuit din următoarele clase:

- clasa `HeritageObject` corespunzătoare tabelei `heritage_objects` din baza de date;
- clasa `Scenario` corespunzătoare tabelei `scenarios` din baza de date;
- clasa `Conversation` corespunzătoare tabelei `conversations` din baza de date;
- clasa `Message` corespunzătoare tabelei `messages` din baza de date;
- clasa `User` corespunzătoare tabelei `users` din baza de date;
- clasa `UserImage` corespunzătoare tabelei `user_groups` din baza de date.

B. Domeniul cunoașterii

În ceea ce privește domeniul cunoașterii, acesta este alcătuit din clasele folosite în mod direct de către motorul de inferență notat cu abrevierea MI în figura 3.5 (unde este reprezentată arhitectura demonstratorului).

Motorul de inferența folosit în cadrul demonstratorului este *Drools Expert*.

Conform referințelor [25], [72], [73], [74], [75] și [76], limbajul caracteristic motorului de inferență *Drools Expert* este *DRL* (*Drools Rule Language*), limbaj care permite utilizarea de clase și sintaxă *Java* pentru declararea regulilor. Totodată, conform acelorași referințe, motorul *Drools Expert* utilizează algoritmul *Rete-OO* (*Rete Object Oriented*) acest aspect făcând posibilă adăugarea faptelor sub forma de obiecte la sesiunile de cunoaștere.

Drept urmare, a fost necesară definirea unor clase pentru a fi folosite în scopul reprezentării regulilor și faptelor în cadrul demonstratorului. Aceste clase nu au un caracter persistent fiind folosite exclusiv de către motorul de inferență în procesul decizional. Ele alcătuiesc domeniul cunoașterii demonstratorului și sunt evidențiate în digrama *UML* de clase din figura 3.7.

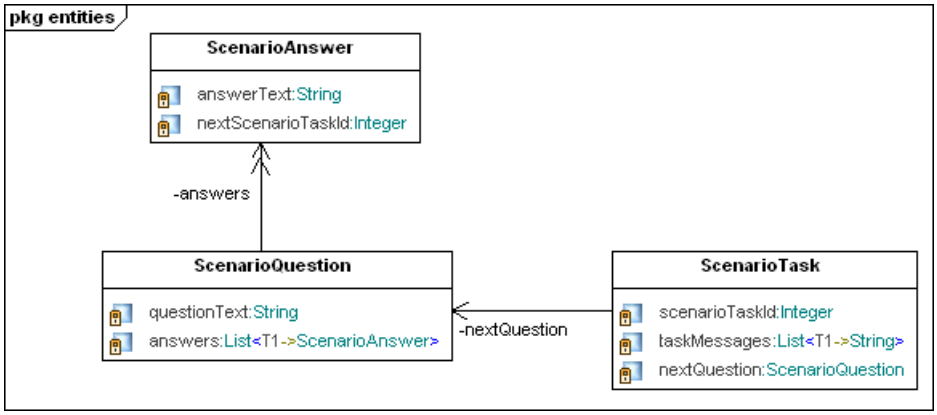

Fig. 3.7 Diagramă *UML* de clase pentru domeniul cunoașterii demonstratorului

Conform figurii 3.7 domeniul cunoașterii demonstratorului include 3 clase, respectiv: `ScenarioTask`, `ScenarioQuestion` și `ScenarioAnswer`. Pe baza acestor clase a fost definit un șablon ce poate fi folosit pentru declararea regulilor în cadrul demonstratorului. Acest șablon prezentat în figura 3.9.

```
rule $ruleName
  when                                          denumire regulă
    scenarioTask : ScenarioTask(getScenarioTaskId() == $scenarioTaskId)
  then                                                              id Task
    String[] messages = {
      $messageText1,                                text mesaj informativ
      $messageText2,
      ...
    };                                                  text întrebare
    String questionText = $quetionText;
    ScenarioAnswer[] answers = {                                text răspuns întrebare
      new ScenarioAnswer($answerText1, $childScenarioTaskId1),
      new ScenarioAnswer($answerText2, $childScenarioTaskId2),
      ...                                                       id task descendent
    };
    scenarioTask.setTaskMessages(Arrays.asList(messages));
    scenarioTask.setNextQuestion(new ScenarioQuestion(questionText, Arrays.asList(answers)));
end
```

Fig. 3.1 Șablon pentru declararea unei reguli în cadrul demonstratorului

3.4.2.2. Componenta de acces la cunoaștere

Componenta de acces la cunoaștere a MAS, este un element cu rol decizional în cadrul demonstratorului, care îndeplinește următoarele funcții principale:

- Creează sesiunile de cunoaștere pentru a determina taskurile care trebuie executate în baza opțiunilor utilizatorilor;
- Realizează transformările de la clasele din domeniul cunoașterii demonstratorului (`ScenarioTask`, `ScenarioQuestion`, `ScenarioAnswer`) la cele din domeniul persistent (`Message`).

Într-un sens mai larg, componenta de acces la cunoaștere decide cu privire la:

- Taskul care trebuie executat la un moment dat pentru un anumit scenariu;
- Mesajele care trebuie create în timpul execuției unui anumit task.

În continuare sunt prezentate câteva aspecte relevante pentru funcționarea componentei de acces la cunoaștere.

Noțiunea de **task** se referă la o clasa (respectiv clasa `ScenarioTask`) din cadrul domeniul cunoașterii demonstratorului. Prin intermediul unui task sunt generate mesaje (adresate turiștilor) referitoare la un obiect de patrimoniu, mesaje care constituie acțiunea unei reguli. Sintetizând, practic un task corespunde unei reguli.

După cum reiese din subcapitolul 3.4.1, regulile de producție sunt stocate în tabela `scenarios` a bazei de date.

Fiecare scenariu din tabela `scenarios` corespunde unui obiect de patrimoniu (în baza relației *one-to-many* dintre tabelele `heritage_objects` și `scenarios`), definește un personaj digital și este însoțit de un set de reguli. Interacțiunea dintre personajul digital și turiști se desfășoară în baza setului de reguli aferent scenariului. O regulă corespunde unui task. Declanșarea regulii conduce la executarea taskului și în consecință la crearea mesajelor definite prin intermediul acestuia.

În cadrul demonstratorului, între taskurile unui scenariu există o relație de precedență care dictează ordinea de execuție a acestora. Această relație poate fi reprezentată printr-un arbore orientat ale cărui noduri sunt taskurile. Pentru evidențierea acestui aspect, în tabelul 3.13 este prezentat codul unui posibil set de reguli ce ar putea fi definit pentru un scenariu al demonstratorului.

Tabel 3.13 Exemplu de posibil set de reguli din cadrul unui scenariu al demonstratorului

```
0    rule "Task 0: Start Mona Lisa scenario"
1      when
2        scenarioTask : ScenarioTask(getScenarioTaskId() == 0)
3      then
4        String[] messages = {
5          "Hello and welcome to the Louvre Museum.",
6          "I'm Mona Lisa. I'm a portrait painting by the italian artist Leonardo da Vinci.",
7          "I was once an item in the collection of King Francis the First of France."
8        };
9        String questionText = new StringBuilder()
10         .append("Would you like to know more about the woman in the painting?")
11         .append("Or perhaps you're more interested to find out about the painter")
12         .append("or about the painting itself?")
13         .toString();
14       ScenarioAnswer[] answers = {
15         new ScenarioAnswer("Hello Mona Lisa. Tell me about the woman in the painting.", 1),
16         new ScenarioAnswer("Hello Mona Lisa. I want to know more about the painter.", 2),
17         new ScenarioAnswer("Hello Mona Lisa. I want to find out about the painting.", 3)
18       };
19       scenarioTask.setTaskMessages(Arrays.asList(messages));
20       scenarioTask.setNextQuestion(new ScenarioQuestion(questionText, Arrays.asList(answers)));
21   end
22   rule "Task 1: About the woman in the painting"
23       when
24           scenarioTask : ScenarioTask(getScenarioTaskId() == 1)
25       then
26           String[] messages = {
27               "Unfortunately I don't know anything about the woman in the painting.",
28               "Bye"
29           };
30           scenarioTask.setTaskMessages(Arrays.asList(messages));
31   end
32   rule "Task 2: About the painter"
33       when
34           scenarioTask : ScenarioTask(getScenarioTaskId() == 2)
35       then
36           String[] messages = {
37               "Unfortunately I don't know anything about the painter.",
38               "Bye"
```

```
39          };
40          scenarioTask.setTaskMessages(Arrays.asList(messages));
41  end
42  rule "Task 3: About the painting"
43      when
44          scenarioTask : ScenarioTask(getScenarioTaskId() == 3)
45      then
46          String[] messages = {
47              "Unfortunately I don't know anything about the painting.",
48              "Bye"
49          };
50          scenarioTask.setTaskMessages(Arrays.asList(messages));
51  end
```

Arborele caracteristic pentru setul de reguli prezentat în tabelul 3.13 este ilustrat în figura 3.10.

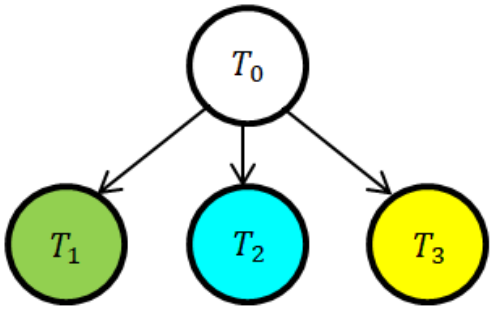

Fig. 3.10 Arbore orientat caracteristic setului de reguli din tabelul 3.13

În contextul demonstratorului, **arborele unui scenariului** reprezintă graful orientat, conex și aciclic (cu rădăcina în T_0) extras din setul de reguli (al scenariului).

Secvența de execuție a unui scenariului este definită printr-o succesiune de taskuri consecutive care formează un drum având originea în rădăcina arborelui scenariului și cealaltă extremitate într-una din frunze (respectiv noduri terminale ale arborelui scenariului). Deoarece un arbore orientat permite existența unui singur drum între rădăcină și un anumit nod terminal (respectiv frunză), numărul de secvențe de execuție posibile pentru un scenariu este egal cu numărul de frunze (aferent arborelui scenariului).

Referitor la situația prezentată în figura 3.10, secvențele de execuție posibile ale scenariului sunt: $T_0 \rightarrow T_1$, $T_0 \rightarrow T_2$ sau $T_0 \rightarrow T_3$.

Prin **execuția unui scenariu** înțelegem parcurgerea arborelui aferent acestuia în baza uneia dintre secvențele de execuție posibile.

Opțiunile utilizatorului pot influența alegerea secvenței de execuție. Prin convenție, **execuția scenariului începe mereu cu taskul T_0**. Așadar, pentru oricare scenariu din cadrul demonstratorului, setul de reguli trebuie să definească în mod obligatoriu un task T_0. Taskul care se execută în urma lui T_0 este unul dintre succesorii acestuia, fiind ales de către sistem pe baza opțiunii utilizatorului. Acest procedeu este repetat până când se ajunge într-un task terminal (respectiv într-o frunză a arborelui scenariului).

În cadrul demonstratorului, componenta de acces la cunoaștere a fost implementată prin intermediul clasei `KnowledgeService`. Diagrama de clase *UML* din figura 3.11 evidențiază caracteristicile clasei `KnowledgeService`.

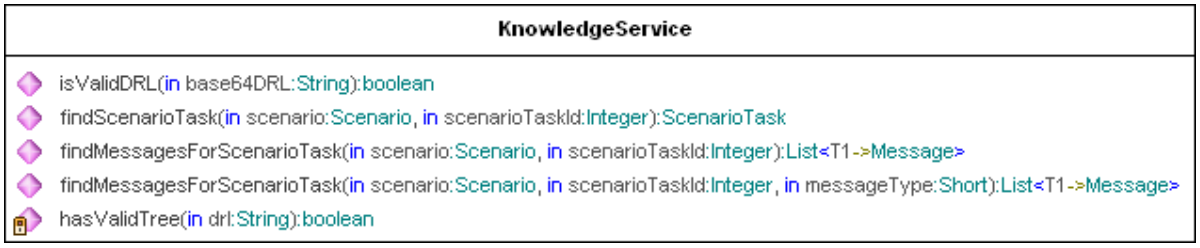

Fig. 3.2 Diagramă de clase *UML* pentru componenta de acces la cunoaștere

Componenta de acces la cunoaștere a fost realizată prin intermediul unui *EJB* (*Enterprise Java Bean*) de tip *Stateless*. Conform referințelor [68] și [70], componentele de tip *Stateless* nu pot fi asociate cu un anumit client deoarece nu păstrează starea între două cereri succesive (respectiv două apeluri consecutive ale metodelor componentei *EJB* provenite de la alte componente / clienți diferiți).

3.4.2.3. Componentele de acces la date

Componentele de acces la date aferente MAS (Module *API* server din figura 3.5) asigură toate operațiile de tip *CRUD* (*Create / Read / Update / Delete*) asupra bazei de date.

Accesul la date se face în baza șablonului (*design-pattern*) *DAO* descris în subcapitolul 1.4.4 al tezei de doctorat. Conform acestui șablon, obiectele *DAO* încapsulează complexitatea necesara operațiilor *CRUD* expunând o interfață comună și mai simplă pentru clienți.

Toate clasele de acces la date din cadrul demonstratorului sunt componente *EJB* (*Enterprise Java Bean*) de tip *Stateless*. Drept urmare, în continuare este folosit termenul de componentă *DAO* pentru a face referire la aceste clase.

Domeniul persistent al demonstratorului include mai multe clase (prezentate în subcapitolul 3.4.2.1). Pentru fiecare clasă din domeniul persistent există o componentă *DAO* responsabilă doar cu operațiile *CRUD* aferente clasei respective. De exemplu, tabela `conversations` din baza de date are un corespondent în domeniul persistent reprezentat prin intermediul clasei `Conversation`. Iar componenta `ConversationDao` este responsabilă cu toate operațiile necesare asupra tabelei `conversations` (ca de exemplu salvarea stării instanțelor clasei `Conversation` în baza de date).

În tabelul 3.15 este evidențiată corespondența între componentele *DAO*, clasele domeniului persistent și tabelele bazei de date.

Tabel 3.15 Corespondență componente DAO / clase domeniu persistent / tabele bază de date

Componentă *DAO*	Clasă domeniu persistent	Tabelă baza de date
`HeritageObjectDao`	`HeritageObject`	`heritage_objects`
`ScenarioDao`	`Scenario`	`scenarios`
`ConversationDao`	`Conversation`	`conversations`
`MessageDao`	`Message`	`messages`
`UserDao`	`User`	`users`
`UserImageDao`	`UserImage`	`user_images`

Diagrama de clase *UML* din figura 3.13 evidențiază toate componentele *DAO* disponibile pe partea de server a demonstratorului.

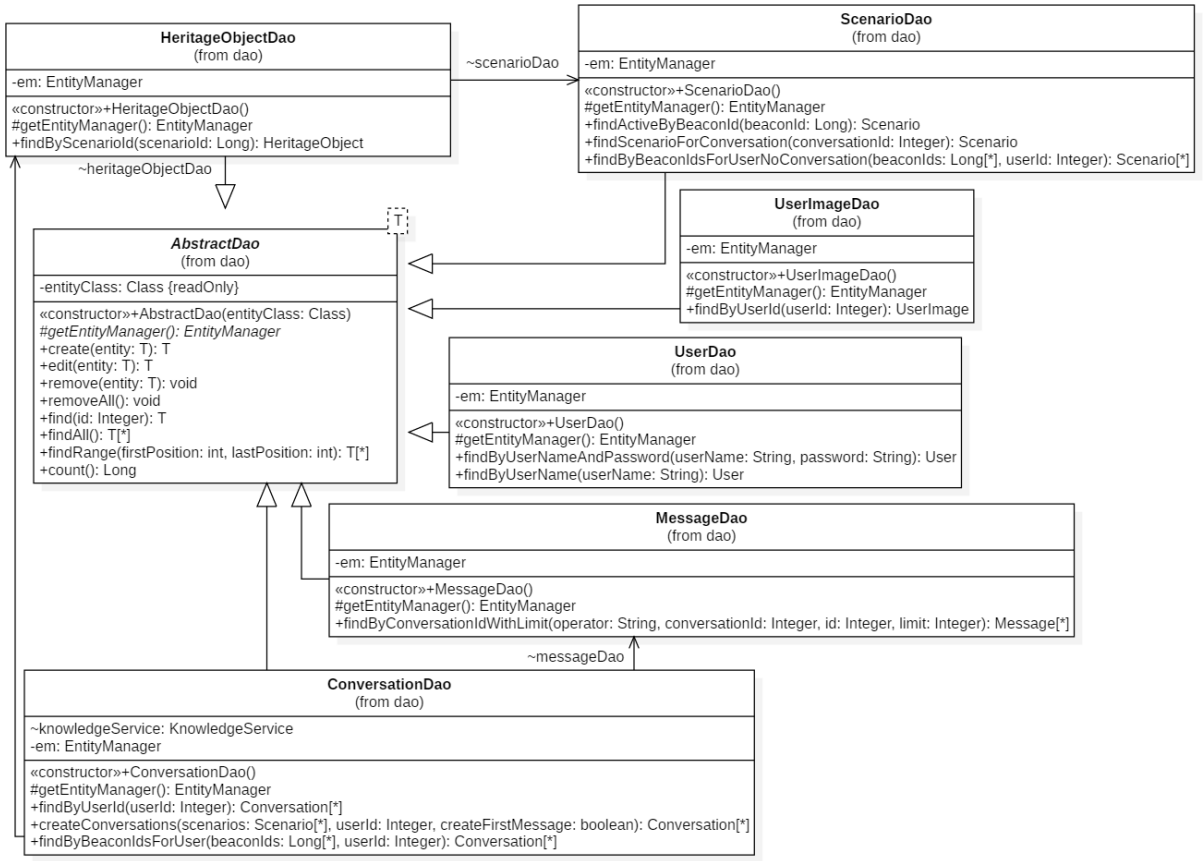

Fig.3.13 Diagramă UML de clase pentru componente *DAO* ale serverului demonstratorului

La începutul prezentului subcapitol, demonstratorului îi este asociată caracteristica de sistem tranzacțional. Caracterul tranzacțional al demonstratorului este relevat în contextul componentelor *DAO* deoarece acestea operează modificări asupra bazei de date.

O tranzacție reprezintă un set de operații sau activități relaționate care trebuie grupate și executate împreună [70]. Tranzacția presupune accesul partajat la una sau mai multe resurse comune (cum ar fi bazele de date) și implică un transfer de informații între participanții (părțile implicate în tranzacție) [70].

Orice tranzacție are un început și un sfârșit, iar operațiile efectuate între începutul și sfârșitul tranzacției devin parte a acesteia. O tranzacția se poate sfârși în două moduri, și anume:

- consemnată (*committed*), situație în care toate modificările realizate în interiorul tranzacției devin permanente;
- anulată sau derulată înapoi (*rolled back*), situație în care sistemul revine la starea anterioară (respectiv la starea în care se afla înainte de începerea tranzacției).

În cadrul platformei *Java EE* există mai multe categorii de tranzacții. Una dintre aceste categorii corespunde tranzacțiilor de tip *CMT* (*Container Managed Transactions*). Conform referinței [70], tranzacțiile de tip *CMT* sunt gestionate automat de către containerul *EJB* (*Enterprise Java Bean*) și presupun utilizarea în mod implicit a librăriei *API JTA* (*Java Transactions API*) care face parte din platforma *Java EE*.

În cadrul demonstratorului toate tranzacțiile sunt de tip *CMT*, managementul acestora fiind realizat exclusiv de către containerul *EJB* pentru toate componentele server.

3.4.2.4. Componenta de autentificare a utilizatorilor

Conform figurii 3.5, în arhitectura software a demonstratorului intră următoarele două aplicați client:

- o aplicație mobilă destinată turiștilor;
- o aplicație *WEB* destinată administratorilor obiectelor de patrimoniu;

care vor fi prezentate pe larg în capitolul 4 al tezei de doctorat.

Cele două aplicații comunică cu serverul sistemului experimental prin intermediul unor servicii *WEB RESTful* care urmează a fi prezentate în subcapitolul 3.4.3.

Serviciile *WEB* pot fi considerate puncte de acces la resursele serverului pentru utilizatorii aplicațiilor client. Drept urmare se întrevede necesitatea identificării utilizatorilor la nivelul serviciilor *WEB* înainte de li se permite accesul. Componenta de autentificare include metode care îndeplinesc acest rol.

3.4.3. Serviciile *WEB*

În cadrul demonstratorului, comunicația între aplicațiile client și server este realizată prin intermediul unor servicii *WEB* de tip *RESTful*.

Conform referinței [86], *REST*, prescurtarea pentru *Representational State Transfer*, reprezintă un tip de arhitectură software destinată sistemelor hipermedia distribuite care funcționează într-o rețea cum ar fi Internetul.

Arhitectura *REST* se bazează pe următorul set de principii, enunțate și în referințele [86], [87], [88], [89], [90] și [91]:

- **Folosirea resurselor**. În cadrul arhitecturii *REST* datele și informațiile sunt considerate resurse. O resursă poate fi un fișier media, un document, o instanță a unei clase etc.
- **Un model client – server**. Arhitectura *REST* implică transferul resurselor în baza unui model de comunicație client – server. Acesta presupune inițializarea comunicației de către client prin intermediul unei cereri. Iar ulterior clientul primește un răspuns din partea serverului.
- **Identificarea resurselor**. În cadrul arhitecturii *REST* fiecare resursă poate fi identificată în mod unic și universal. De cele mai multe ori arhitectura *REST* este folosită împreună cu protocolul *HTTP*. Iar identificarea resurselor în acest caz se face prin intermediul adresei *URL*.
- **Reprezentarea resurselor**. Conform arhitecturii *REST*, o resursă poate avea multiple reprezentări. Sau, altfel spus, exista mai multe formate și tipuri de conținut disponibile. Iar clientul și serverul pot negocia asupra tipului de conținut folosit.
- **Manipularea resurselor**. Resursele pot fi manipulate prin operații *CRUD* (*Create Read Update Delete*). În cazul utilizării împreună cu protocolul *HTTP* se folosesc metodele (denumite și verbe *HTTP*) disponibile: *GET*, *PUT*, *POST*, și *DELETE*. Corespondența între

operațiile *CRUD* și verbele *HTTP* este: *Create* -> *POST*, *Read* -> *GET*, *Update* -> *PUT*, *Delete* -> *DELETE*.

- **Comunicația fără păstrarea stării între apeluri** (***Stateless communication***). Serverul nu păstrează starea în raport cu un anumit client între cereri.

Suportul pentru dezvoltarea serviciilor *WEB* de tip *RESTful* în cadrul platformei *Java EE* este definit prin intermediul specificației *JAX-RS* 2.0 (*JSR-339: The Java API for RESTful Web Services*) disponibilă la următorul link: https://jcp.org/aboutJava/communityprocess/final/jsr339/index.html. Totodată, există mai multe implementări conforme cu *JAX-RS* 2.0. Una dintre acestea este *Jersey RESTful Web Services* (https://jersey.github.io) care a fost folosită pentru dezvoltarea serviciilor *WEB* ale demonstratorului.

În cadrul sistemului experimental, pentru fiecare clasă din domeniul persistent (respectiv resursă) există câte un serviciu *WEB* accesibil aplicațiilor client. Singura excepție de la aceasta regulă o face clasa `User` folosită intern în procesul de autentificare al utilizatorilor. De asemenea, este posibilă și descărcarea seturilor de reguli aferente scenariilor sub forma unor fișiere text.

Tabelul 3.20 evidențiază corespondență între serviciile *WEB* (clasele folosite pentru definirea acestora) ale demonstratorului și clasele domeniului persistent. Totodată, în tabelul 3.20 este indicat și identificatorul (adresa *URL* relativă) resursei pentru fiecare serviciu în parte.

Tabel 3.20 Corespondența între serviciile *WEB* și clasele domeniului persistent ale demonstratorului

Serviciu *WEB*	**Clasă domeniu persistent (resursă)**	**URL relativ**
`HeritageObjectService`	`HeritageObject`	/services/heritage-objects
`ScenarioService`	`Scenario`	/services/scenarios
`ConversationService`	`Conversation`	/services/conversations
`MessageService`	`Message`	/services/messages
`UserImageService`	`UserImage`	/services/user-images
`RulesService`	Folosit pentru a descărca fișiere care conțin seturile de reguli aferente scenariilor.	/services/rules

În cadrul demonstratorului, serviciile *WEB* definesc metode care sunt apelate de către aplicațiile client. Într-un sens mai larg, parametrii acestor metode pot fi considerate intrări ale serverului. Totodată, datele rezultatele în urma execuției metodelor reprezintă ieșiri ale serverului.

În figura 3.15 sunt ilustrate în mod sintetic toate metodele serviciilor WEB împreună cu intrările (respectiv parametrii metodelor) și ieșirile (respectiv datele rezultate în urma execuției metodelor) serverului.

Conform figurii 3.15, din punct de vedere funcțional, sistemul experimental incluzând serverul și aplicațiile client poate fi împărțit în 3 subsisteme, și anume:

- subsistemul de management al scenariilor (permite definirea scenariilor de către administratorii obiectelor de patrimoniu);
- subsistemul de selecție automata a scenariilor (permite crearea în mod automat a conversațiilor între turiști și personajele digitale pe baza scenariilor definite);

- subsistemul de execuție al scenariilor (permite crearea mesajelor în timpul execuției scenariilor).

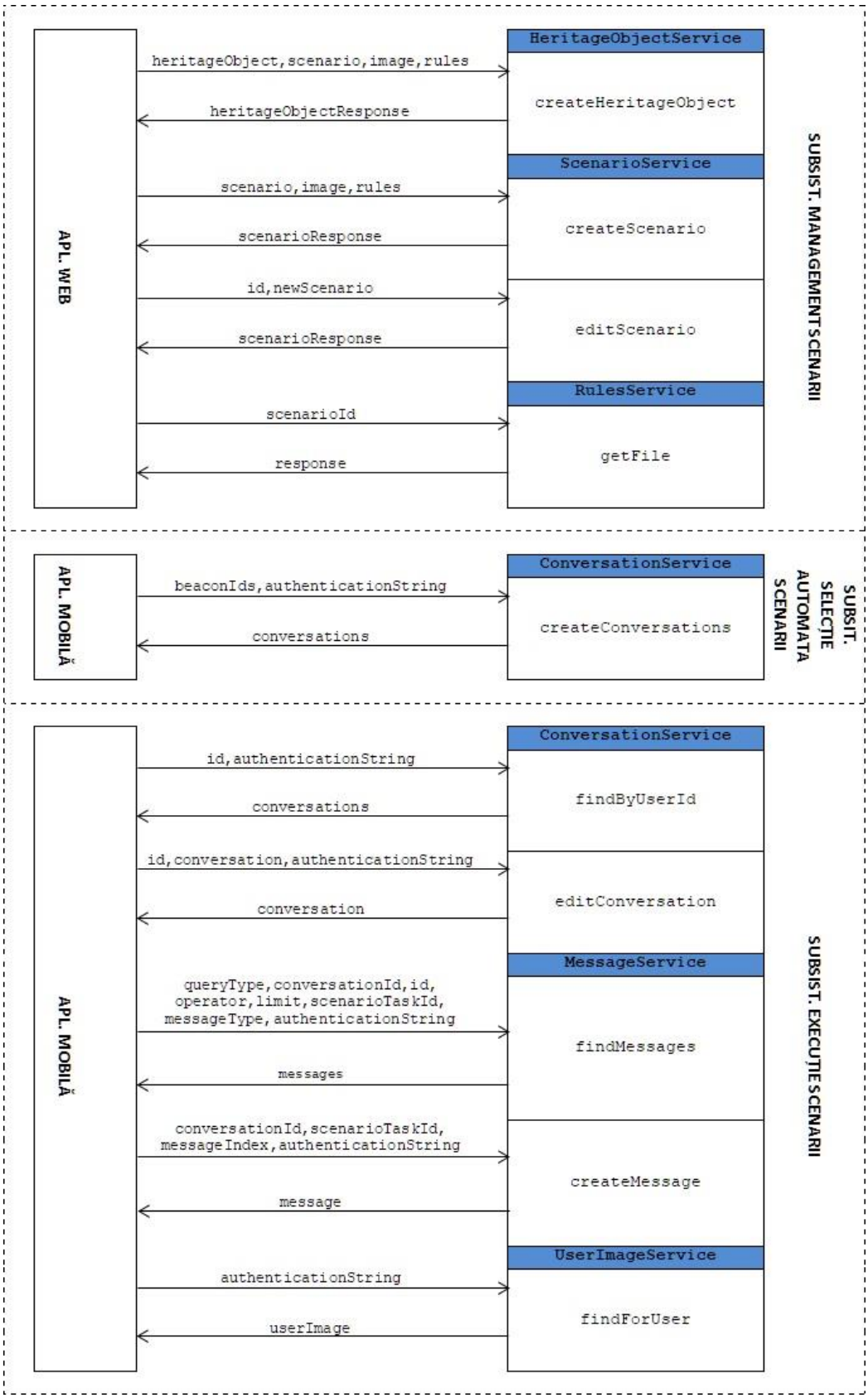

Fig. 3.15 Schemă subsisteme demonstrator

3.5. Concluzii parţiale

În capitolul 3 al tezei de doctorat sunt evidențiate și analizate probleme ce țin de proiectarea și realizarea componentelor server ale sistemului experimental. Demonstratorul a fost realizat pe baza platformei *Java EE* (*Enterprise Edition*). În ceea ce privește noțiunea de componentă (sau componentă *EJB*), explicată în cadrul capitolului, aceasta este folosită în sensul specific platformei *Java EE*.

Capitolul începe prin definirea termenului demonstrator în raport cu obiectivele tezei de doctorat. Totodată, sunt prezentate funcțiile generale ale sistemelor expert și este evidențiat modul de implementarea al acestora în cadrul demonstratorului.

Capitolul continuă prin expunerea unor noțiuni importante pentru domeniul de aplicare al demonstratorului. Astfel, sunt introduși și definiți termeni precum patrimoniu cultural și obiecte de patrimoniu și este prezentat conceptul care stă la baza sistemului experimental.

Ulterior, este propusă arhitectura software a demonstratorului care include următoarele 6 module software (2 client și 4 server):

- aplicația mobilă (client pentru sistemul de operare *Android* destinat utilizării de către turiști);
- aplicația *WEB* (client destinat rulării într-un *browser* de Internet de către administratorii obiectelor de patrimoniu);
- baza de date (modul server independent care stochează toate datele din cadrul sistemului experimental);
- motorul de inferență (modul server independent folosit în procesul decizional);
- MAS (Modulele *API* server care includ toate componentele *EJB* ce asigură funcționalitatea *back-end* a sistemului experimental);
- serviciile *WEB* (componente *EJB* server folosite pentru comunicație).

În continuarea capitolului, sunt prezentate în detaliu fiecare dintre modulele server enumerate anterior. Aspectele referitoare proiectarea și realizarea aplicațiilor client aferente demonstratorului nu fac obiectul capitolului 3 fiind abordate în capitolul 4.

Demonstratorul folosește o bază de date relațională pentru stocarea regulilor și a rezultatelor inferențelor. Drept urmare, sunt evidențiate toate elementele caracteristice bazei de date incluzând: **tabele**, **câmpuri**, **tipuri de date**, **referințe**, **constrângeri**. Modelul bazei de date este redat sub forma unei diagrame *ER*. Totodată, pentru fiecare tabelă în parte, este furnizat codul *SQL* folosit în scopul definirii acesteia.

Domeniul demonstratorului face parte din modulele *API* server (MAS). Acesta include atât clasele din **domeniul persistent** (clase care permit salvarea stării în baza de date), cât și pe cele din **domeniul cunoașterii** (folosite de către motorul de inferența). Ca atare, sunt descrise caracteristicile acestor clase însoțite și de anumite exemple din codul sursă al demonstratorului.

În cadrul sistemului experimental, componenta de acces la cunoaștere (parte integrantă a MAS) permite interacțiunea cu motorul de inferența în scopul determinării:

- taskurilor care trebuie executate la un moment dat pentru un anumit scenariu;
- mesajelor care trebuie create în timpul execuției unui anumit task.

În consecință, este prezentată această componentă. De asemenea sunt definite noțiuni importante pentru funcționarea componentei (respectiv **regulă**, **faptă**, **raționament**, **task**, **mesaj**, **secvența de execuție a scenariului**, **arbore scenariu** etc.) și este propus un șablon (construit pe baza claselor din domeniul cunoașterii) pentru a fi utilizat la definirea regulilor în cadrul sistemul experimental.

Conform conceptului enunțat, prin intermediul demonstratorului au loc interacțiuni între personajele digitale și utilizator (turist) în tipul vizitelor acestuia la situri culturale. Pentru modelarea interacțiunii între un utilizator și un *avatar*, autorul tezei de doctorat propune folosirea unui arbore orientat. Așadar, sunt introduse câteva noțiuni din teoria grafurilor, este evidențiată **relația de precedența între taskurile unui scenariu** și este definită **dependența regulă – task**.

Componentele de acces la date fac parte tot din MAS și asigură toate operațiile *CRUD* (*Create Read Update Delete*) asupra bazei de date a sistemului experimental. Prezentul capitol include o prezentare a acestor componente alături de câteva exemple din codul sursă al demonstratorului. Relațiile dintre componentele de acces la date sunt modelate prin intermediul unei diagrame de clase *UML*.

Conform arhitecturii software propuse, un ultim element al MAS este componenta de autentificare a utilizatorilor. Drept urmare, sunt prezentate câteva noțiuni teoretice legate de securitatea sistemelor de programe (respectiv **autentificare**, **autorizare**, **grup**, **rol**, **domeniu de securitate**) și este descris și modul de funcționare al acestei componente.

Partea finală a capitolului este alocată serviciilor *WEB* care reprezintă componente server folosite pentru comunicație. Acestea primesc cererile *HTTP* din partea clienților (aplicațiile client folosite de către utilizatori) și le onorează delegând responsabilități specifice către componentele interne (care fac parte din MAS) ale serverului. Interacţiunile dintre componentele MAS (modul în care componentele colaborează în vederea îndeplinirii cererilor din partea clienților) sunt modelate prin intermediul unor diagrame de secvențe *UML* incluse în anexa A.3 a capitolului.

În ceea ce privește tehnologiile folosite pentru realizarea componentelor server ale demonstratorului, pe parcursul capitolului 3 sunt evidențiate aspecte teoretice referitoare la:

- *Java EE* (platforma utilizată pentru dezvoltarea componentelor server);
- Arhitectura *EJB* (arhitectura care stă la baza aplicațiilor și sistemelor dezvoltate folosind platforma *Java EE*)
- *JPA* (un *API* bazat pe tehnici *ORM* care face parte din platforma *Java EE* și care a fost folosit pentru implementarea accesului la baza de date a sistemului experimental);
- *JTA* (un *API* caracteristic platformei *Java EE* care permite managementul și demarcarea tranzacțiilor).

Referitor la modul de realizare al componentelor server, din capitolul 3 mai pot fi desprinse următoarele concluzii:

- Prin folosirea tehnicilor *ORM*, a tipurilor generice și a proprietăților moștenirii claselor a fost redus efortul necesar dezvoltării componentelor de acces la date;
- Prin folosirea metodei *DAO* (*Data Access Object*) s-a realizat decuplarea și izolarea restului aplicației de elementele specifice accesului la baza de date. Acest lucru a condus la o claritate și transparență mai mare a codului și implicit la reducerea efortului de mentenanță;

- Prin folosirea arhitecturii *EJB* și a librăriilor *API* specifice platformei *JAVA EE* a fost redus efortul de dezvoltare pentru toate componentele server. Conform cu [71], platforma *JAVA EE* pune la dispoziția componentelor *EJB* mecanisme care permit managementul tranzacțiilor, accesul concurențial, utilizarea în comun a resurselor etc. Drept urmare, prin utilizarea acestei platforme mecanismele menționate anterior au fost disponibile în mod implicit fără a necesita un efort de dezvoltare.

Din punct de vedere al standardizării, la realizarea componentelor server ale demonstratorului, s-a ținut cont de următoarele specificații:

- *RFC4648 –> The Base16, Base32, and Base64 Data Encodings*;
- *JSR-345 –> Enterprise JavaBeans 3.2*;
- *RFC7231 –> Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*;
- *RFC7540 –> Hypertext Transfer Protocol Version 2 (HTTP/2)*;
- *RFC8446 –> The Transport Layer Security (TLS) Protocol Version 1.3*;
- *JSR-342 –> Java Platform, Enterprise Edition 7*;
- *JSR-337 –> Java SE 8*;
- *JSR-335 –> Lambda Expressions for the Java Programming Language*;
- *JSR-317 –> Java Persistence 2.0*;
- *RFC8259 –> The JavaScript Object Notation (JSON) Data Interchange Format*;
- *JSR-907 –> Java Transaction API*;
- *ISO/IEC 9075-1:2016 –> Database languages - SQL - Part 1: Framework (SQL/Framework)*;
- *RFC3986 –> Uniform Resource Identifier (URI): Generic Syntax*;
- *JSR-94 –> Java Rule Engine API*;
- *RFC5280 –> Internet X.509 Public Key Infrastructure Certificate*;
- *JSR-339 –> JAX-RS 2.0: The Java API for RESTful Web Services*;
- RFC2388 –> *Returning Values from Forms: multipart/form-data*.

Capitolul 4. Contribuții privind proiectarea și realizarea aplicațiilor client ale unui sistem expert senzitiv la context pentru obiecte de patrimoniu

Capitolul 4 conține contribuțiile autorului cu privire la proiectarea și realizarea aplicațiilor client ale demonstratorului.

Demonstratorul sistemului expert dezvoltat include două aplicații client, respectiv:

- aplicația mobilă destinată vizitatorilor (turiștilor)
- aplicația *WEB* dedicată administratorilor obiectelor de patrimoniu.

Ca atare, capitolul 4 începe prin prezentarea aplicației mobile.

Ulterior, în cea de a doua parte a capitolului, sunt definite caracteristicile aplicației *WEB*.

După prezentarea celor două aplicații, în cea de a treia parte a capitolului, sunt evidențiate rezultatele obținute ca urmare a 4 noi experimente. Cele 4 experimente au vizat determinarea răspunsului demonstratorului la două tipuri de intrări (treaptă și rampă) în cazul utilizării metodelor MDFC și MDCC (propuse de către autor și descrise în subcapitolul 2.2 al tezei de doctorat). În timpul experimentelor au fost monitorizate următoarele 3 variabile: traficul generat în rețea, timpul de răspuns și a resursele de memorie alocate de către serverul demonstratorului.

În finalul capitolului sunt evidențiate concluziile parțiale.

4.1. Contribuții privind proiectarea și realizarea aplicației mobile a demonstratorului

Conform arhitecturii software a demonstratorului prezentată în subcapitolul 3.3 al tezei de doctorat, sistemul experimental include o aplicație client destinată turiștilor. Aplicația client a fost realizată în limbajul de programare *Java*, este compatibilă cu sistemul de operare *Android* și poate fi instalată pe terminalele de tip smartphone ale utilizatorilor pentru a fi folosită în timpul vizitelor la siturile culturale.

Prezentul subcapitol tratează probleme ce țin de proiectarea și realizarea aplicației client destinată turiștilor. Această este referită și prin termenii de **aplicație mobilă** sau **client *Android*** pe parcursul subcapitolului 4.1.

Aplicația mobilă are următoarele 7 cazuri de utilizare:

- AM1 – *Vizualizează Conversații*;
- AM2 – *Vizualizează Informații Cont*;
- AM3 – *Editează Date Autentificare*;
- AM4 – *Pornește Modul Interactiv*;
- AM5 – *Oprește Modul Interactiv*;
- AM6 – *Vizualizează Mesaje Conversație*;
- AM7 – *Creează Următorul Mesaj*,

unde notația AMi cu $i = \overline{1..7}$ reprezintă codul de identificare alocat fiecărui caz de utilizare, conform cerinței ce reiese din șablonul prezentat în tabelul 1.18 (subcapitolul 1.3 al tezei).

Cazurile de utilizare ale aplicației mobile enumerate mai sus (respectiv AM1 … AM7) au fost definite respectând șablonul din subcapitolul 1.3 al tezei de doctorat (tabelul 1.18). Acestea reprezintă specificația pe baza cărei a fost realizată aplicația mobilă și sunt reprezentate sugestiv prin intermediul diagramei *UML* din figura 4.1 de mai jos.

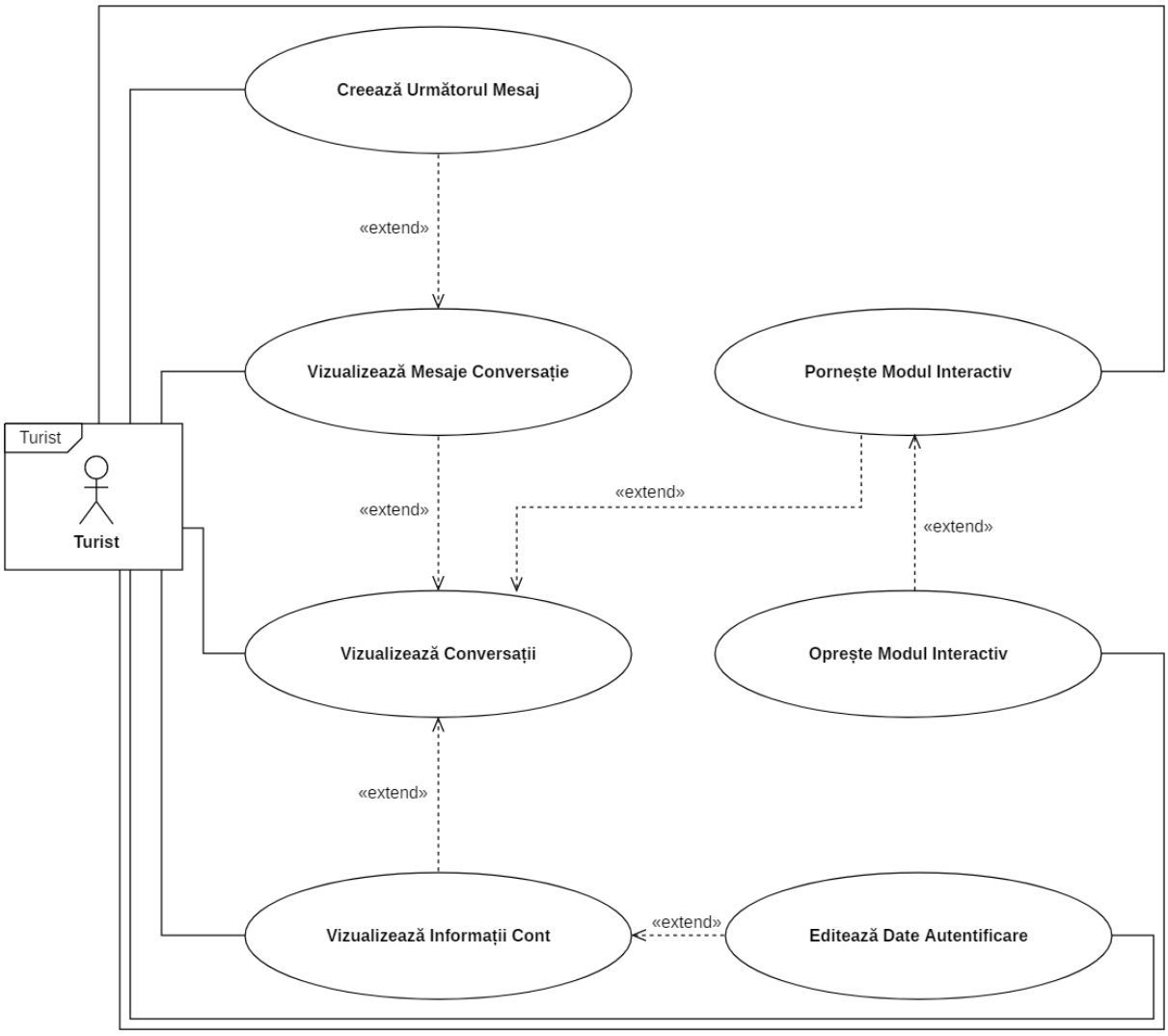

Fig. 4.1 Diagrama *UML* a cazurilor de utilizare aferente aplicației mobile

Aplicația mobilă dezvoltată conține următoarele trei ecrane:

- ecranul destinat vizualizării conversațiilor (referit pe scurt prin termenul **ecran conversații**);
- ecranul destinat vizualizării mesajelor (referit pe scurt prin termenul **ecran mesaje**);
- ecranul destinat vizualizării informațiilor contului de utilizator (referit pe scurt prin termenul **ecran info-cont**).

Folosindu-se de aceste ecrane utilizatorii aplicației mobile pot declanșa oricare dintre cele 7 cazuri de utilizare (respectiv AM1 … AM7) menționate anterior.

În contextul aplicației mobile dezvoltate, un **ecran** corespunde unui element de interfață grafică de tipul unei ferestre care ocupă întregul display al terminalului mobil la un moment dat.

Ecranul poate conține și alte elemente de interfață (butoane, căsuțe text etc.) prin intermediul cărora utilizatorul poate interacționa cu aplicația.

Poziționarea clientului *Android* în cadrul subsistemelor demonstratorului este reprezentată prin intermediul schemei din figura 4.2 ce reia o parte a schemei din figura 3.15 (subcapitolul 3.4.3 al tezei de doctorat). În figura 4.2 este evidențiată și corespondența între metodele serviciilor *WEB* și cazurile de utilizare (în sensul că o metodă a unui serviciu *WEB* poate fi folosită într-unul sau mai multe cazuri de utilizare). În ceea ce privește cazul de utilizare AM5, acesta nu necesită apelarea unui serviciu *WEB*.

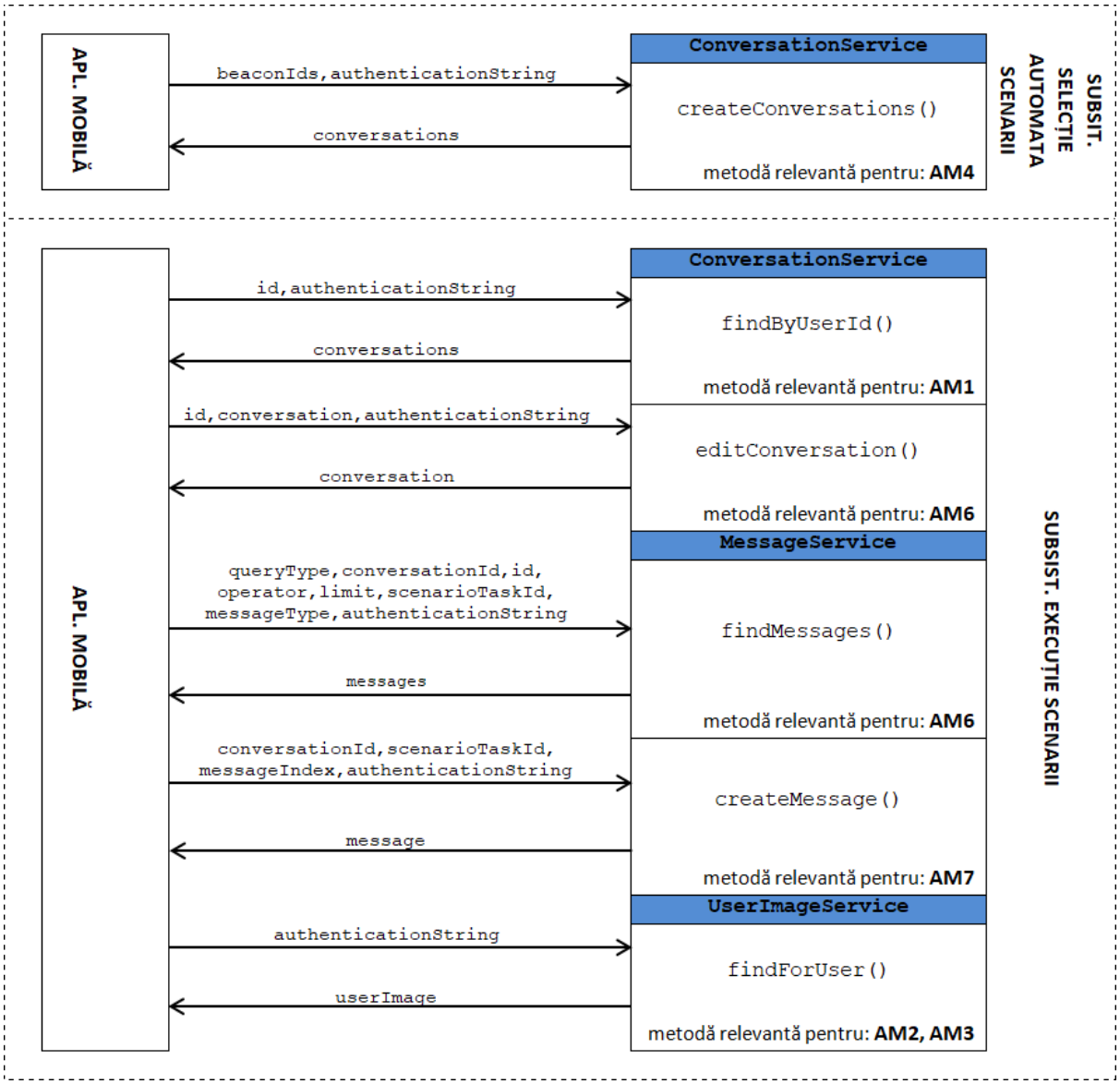

Fig. 4.2 Poziționarea aplicației mobile în cadrul subsistemelor demonstratorului

Aplicația mobilă joacă un rol esențial în determinarea contextului turiștilor. Folosindu-se de interfața *Bluetooth* a terminalului mobil, aceasta detectează semnalele radio emise de către dispozitivele *beacon* atașate obiectelor de patrimoniu. În baza modificărilor contextului, personajele digitale inițializează automat conversații cu utilizatorul pentru a-i transmite informații despre obiectele de patrimoniu. Contextului utilizatorilor este determinat prin metoda MDCC prezentată în capitolul 2 (subcapitolul 2.2.3 al tezei de doctorat).

Sistemul experimental monitorizează contextul utilizatorului doar atunci când aplicația mobilă funcționează în modul interactiv. Cu alte cuvinte modul interactiv corespunde modului de funcționare automat. Drept urmare, prezentul subcapitol acordă o importanța mare cazului de utilizare AM4 – *Pornește Modul Interactiv*.

De asemenea, este de menționat faptul că aplicația mobilă prezintă câteva similarități cu aplicațiile de mesagerie (cum ar fi aplicații *SMS*, *WhatsApp*, *Skype* etc.). S-a adoptat această abordare deoarece clientul *Android* facilitează comunicația între turiști și personajele digitale, așa cum, în mod asemănător, aplicațiile de mesagerie permit trimiterea de mesaje text între utilizatori. Totodată, conform studiilor de piață aplicațiile de mesagerie au o mare popularitate în rândul publicului. Drept urmare, este de așteptat ca utilizatorii să fie familiarizați cu anumite elemente caracteristice modului de utilizare specific acestor aplicații.

Cele 7 cazuri de utilizare (respectiv AM1…AM7 care reprezintă specificația aplicației mobile) împreună cu anumite detalii care țin de realizarea clientului *Android* sunt prezentate în cadrul tezei de doctorat dar au fost omise din acest rezumat pe considerente de spațiu.

Ca o ultimă mențiune, în urma dezvoltării clintului *Android* aceasta a fost instalat și testat pe un terminal *Huawei P20 Lite*. Pe durata testelor au fost realizate anumite capturi de ecran direct de pe terminalul *Huawei*, o parte dintre acestea fiind prezentate în continuare.

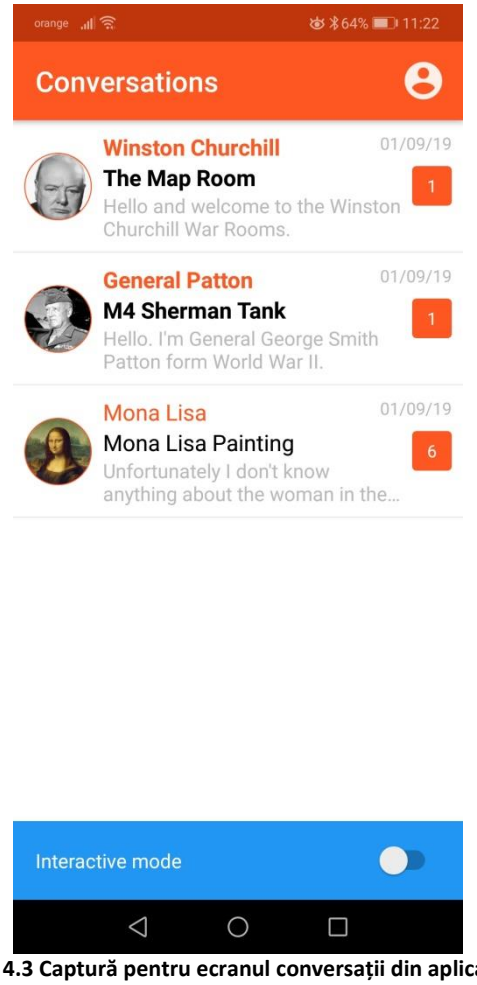

Fig. 4.3 Captură pentru ecranul conversații din aplicația mobilă

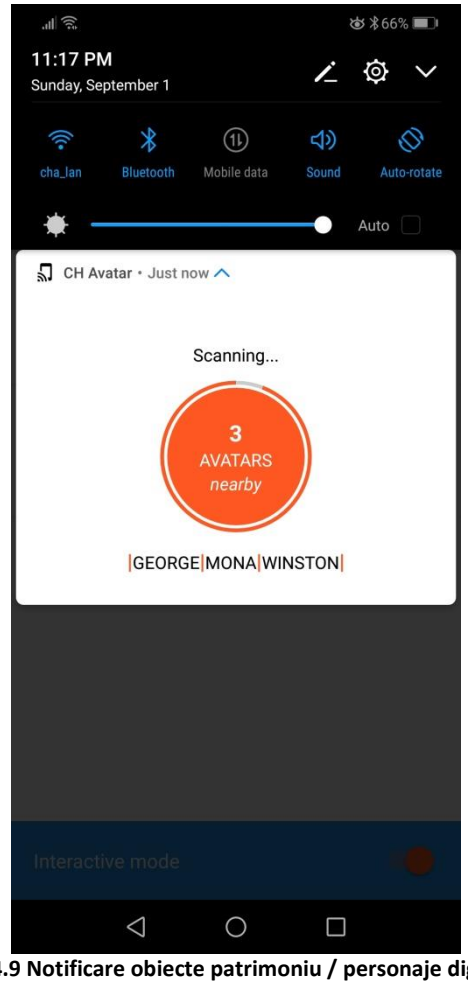

Fig. 4.9 Notificare obiecte patrimoniu / personaje digitale

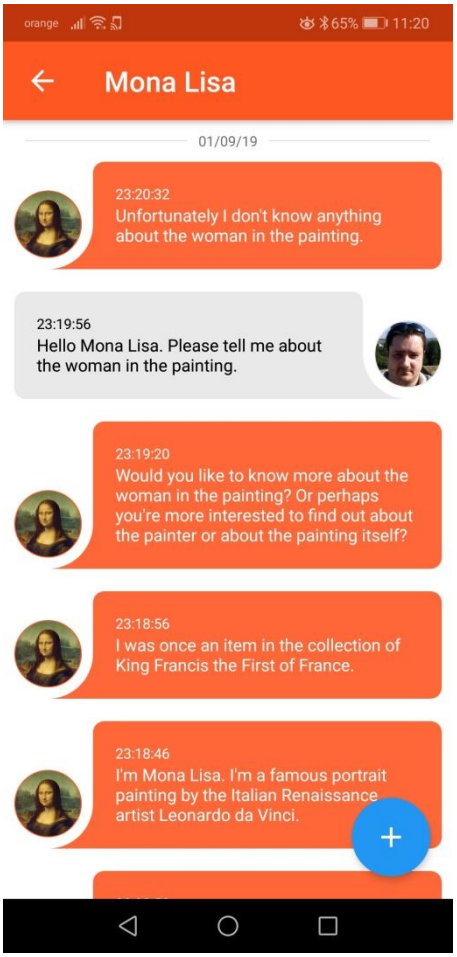

Fig. 4.10 Captură ecran mesaje din aplicația mobilă

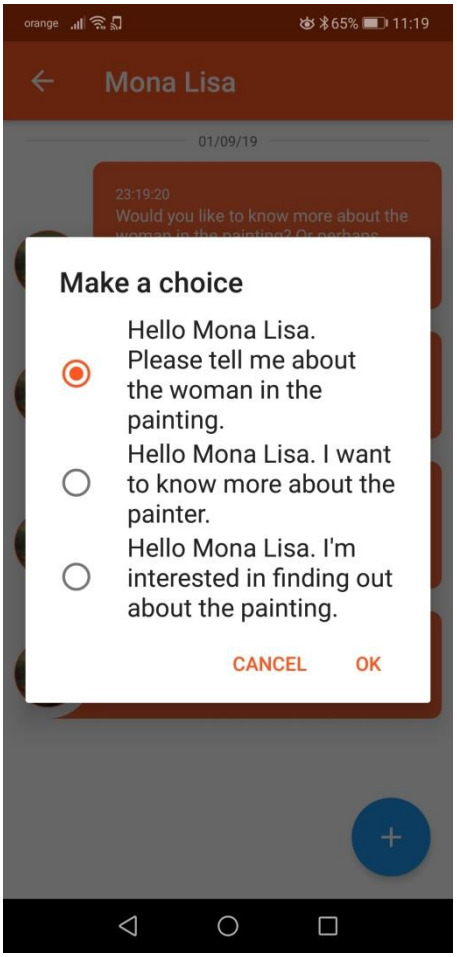

Fig. 4.11 Captură fereastră de dialog cu opțiunile utilizatorului din aplicația mobilă

4.2. Contribuții privind proiectarea și realizarea aplicației *WEB* a demonstratorului

Conform arhitecturii software a demonstratorului (prezentată în subcapitolul 3.3 al tezei de doctorat), sistemul experimental include o aplicație client destinată administratorilor obiectelor de patrimoniu (respectiv interfața sistemului cu expertul uman evidențiată și în figura 3.2 din subcapitolul 3.1 al tezei de doctorat). Aplicația client poate fi accesată dintr-un *browser* de *Internet* compatibil și se bazează pe limbajele *Java*, *JavaScript* și *HTML*. Prin intermediul aplicației pot fi definite obiectele de patrimoniu și scenariile în cadrul demonstratorului.

Prezentul subcapitol tratează probleme ce țin de proiectarea și realizarea aplicației client destinată administratorilor obiectelor de patrimoniu. Aceasta este referită și prin termenii de **aplicație *WEB*** sau **client *WEB*** dea lungul subcapitolului 4.2.

Pentru clientul *WEB* au fost definite următoarele 7 cazuri de utilizare:

- AW1 – *Log In*;
- AW2 – *Adaugă Obiect Patrimoniu*;
- AW3 – *Adaugă Scenariu*;
- AW4 – *Activează Scenariu*;

- AW5 – *Dezactivează Scenariu*;
- AW6 – *Descarcă Reguli Scenariu*;
- AW7 – *Log Out*,

unde notația AWi cu $i = \overline{1 \dots 7}$ reprezintă codul de identificare alocat fiecărui caz de utilizare, conform cerinței care reiese din șablonul prezentat în tabelul 1.18 (subcapitolul 1.3 al tezei de doctorat).

Similar modului de abordare adoptat pentru aplicația mobilă, cazurile de utilizare ale clientului *WEB* (AW1 … AW7) au fost definite respectând șablonul din subcapitolul 1.3 al tezei de doctorat (tabelul 1.18). Acestea sunt reprezentate sugestiv prin intermediul diagramei *UML* din figura 4.15.

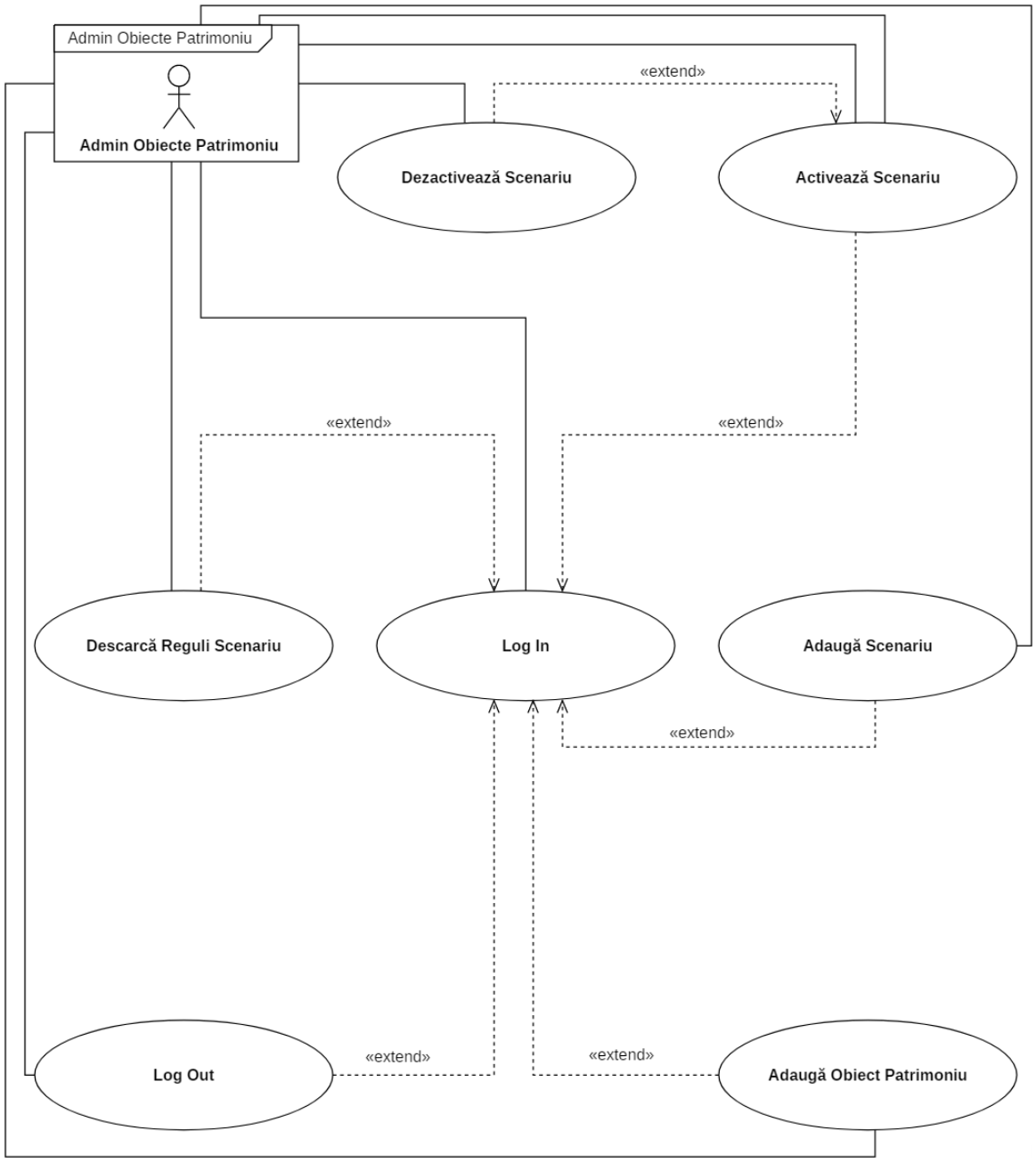

Fig. 4.15 Diagrama cazurilor de utilizare aferente aplicație *WEB*

Aplicația *WEB* dezvoltată conține următoarele două pagini:

- **pagina de *Log In***;
- **pagina principală a aplicației**.

Cu excepția autentificării (AW1 – *Log In*), toate cazurile de utilizare aferente clientului *WEB* (AW2 … AW7) pot fi declanșate din pagina principală a aplicației.

Poziționarea aplicației *WEB* în cadrul subsistemelor demonstratorului este reprezentată prin intermediul schemei din figura 4.16 (care reia o parte a schemei din figura 3.15 – subcapitolul 3.4.3 al tezei de doctorat) de mai jos în care este evidențiată și corespondența între metodele serviciilor *WEB* și cazurile de utilizare (în sensul că o metodă a unui serviciu *WEB* poate fi folosită într-unul sau mai multe cazuri de utilizare). În ceea ce privește, cazurile de utilizare AW1 (*Log In*) și AW7 (*Log Out*) nu necesită apelarea unui serviciu *WEB*.

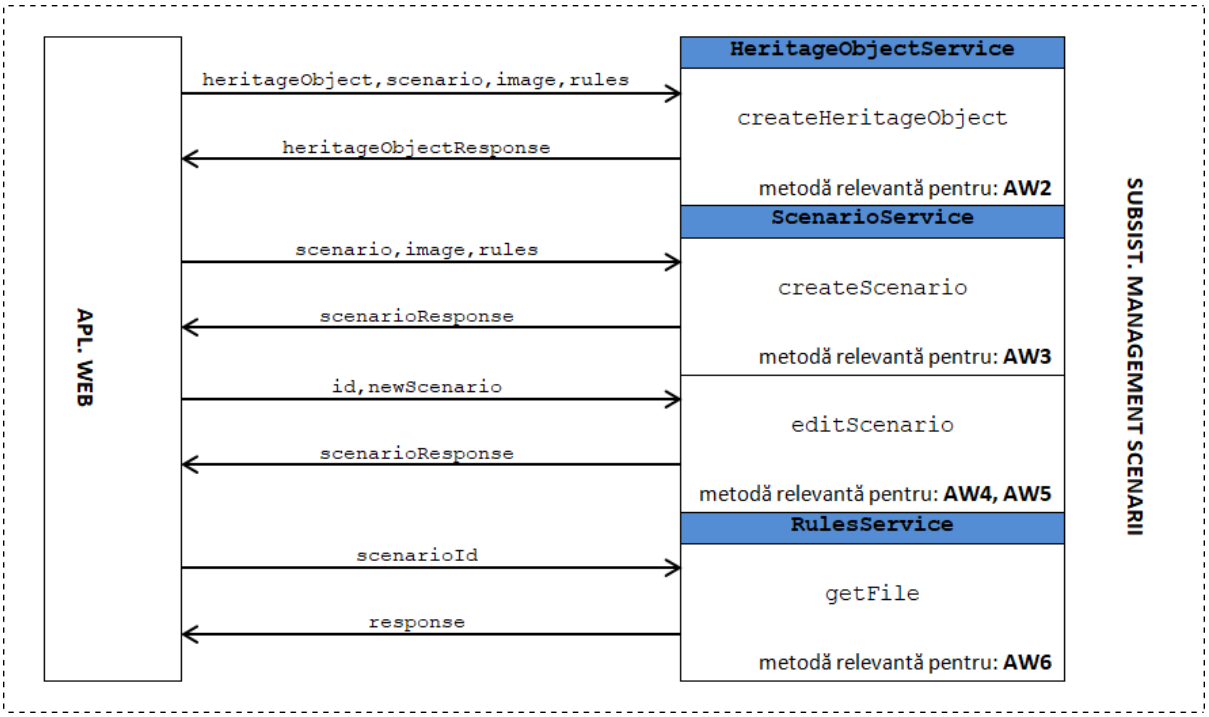

Fig. 4.16 Poziționarea aplicației *WEB* în cadrul subsistemelor demonstratorului

Cele 7 cazuri de utilizare (respectiv AW1…AW7 care reprezintă specificația aplicației *WEB*) împreună cu anumite detalii care țin de realizarea clientului *WEB* sunt prezentate în cadrul tezei de doctorat dar au fost omise din acest rezumat pe considerente de spațiu.

Este de menționat faptul că în urma dezvoltării aplicației *WEB* aceasta a fost testată prin intermediul unui *browser* (*Firefox Quantum* 69.0.3 varianta pe 64 de biți pentru *Windows*). Pe durata testelor au fost realizate anumite capturi de ecran ale paginilor clientului *WEB*, o parte dintre acestea fiind prezentate în cele ce urmează.

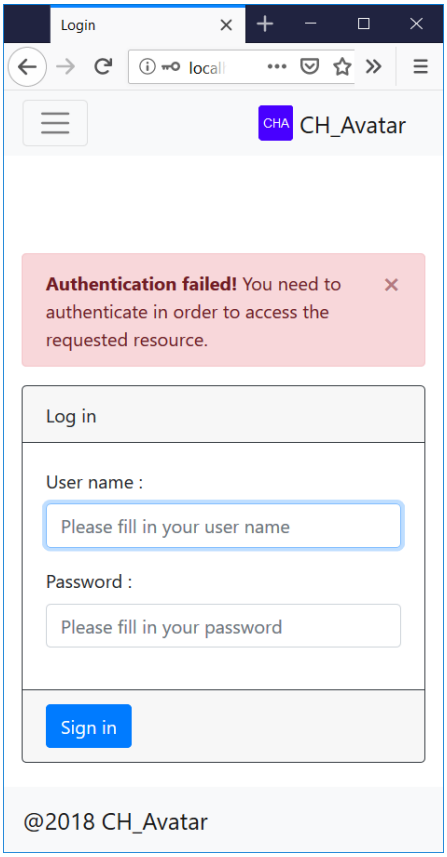

Fig. 4.17 Pagina *Log In* aplicație *WEB*

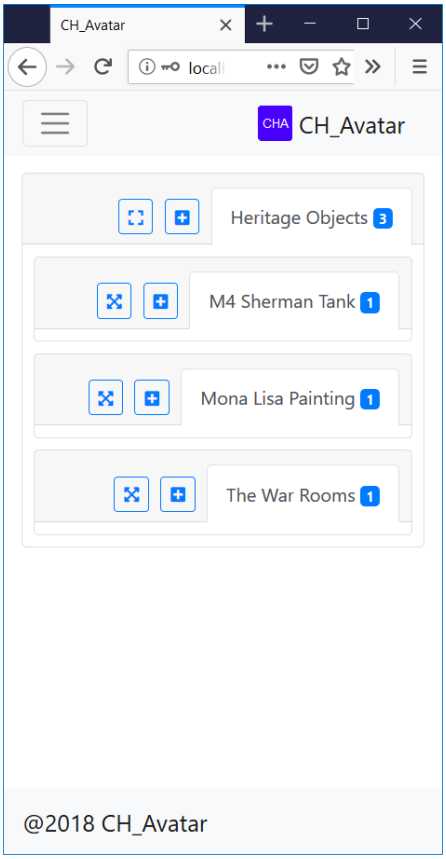

Fig. 4.18 Pagina principală aplicație *WEB*

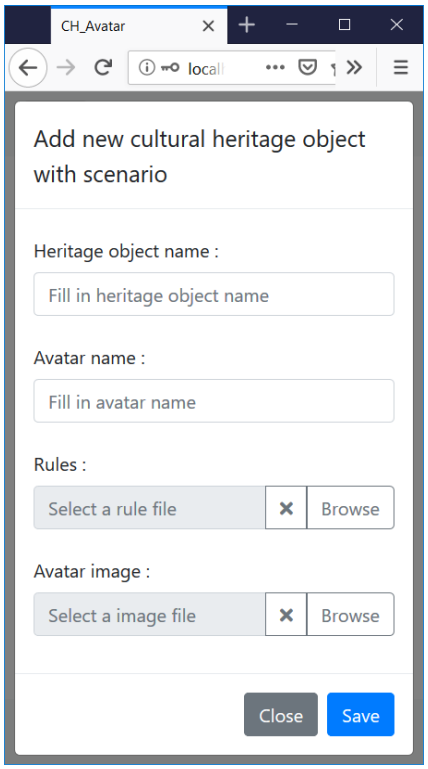

Fig.4.20 Captură fereastră de dialog pentru adăugarea unui nou obiect de patrimoniu

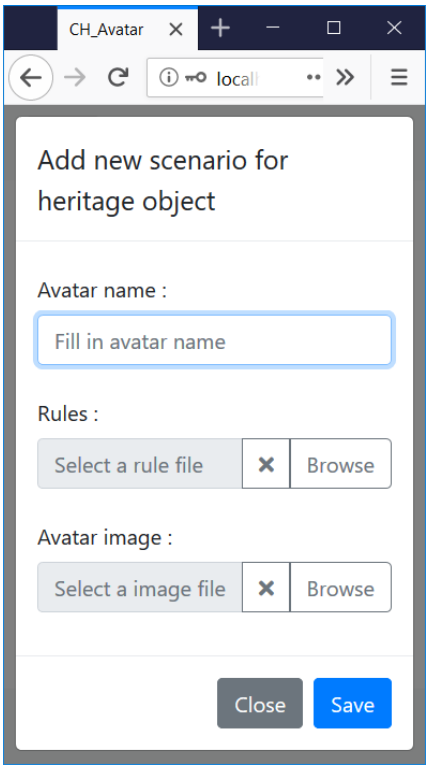

Fig. 4.21 Captură fereastră de dialog pentru adăugarea unui nou scenariu

4.3. Rezultate experimentale comparative obţinute ca urmare a testării metodelor *MDCC* şi *MDFC* în cazul demonstratorului

Demonstratorul (aferent sistemului expert) este un sistem senzitiv la context, tranzacțional, distribuit și cu o arhitectură de tip client-server. Sistemele cu aceste caracteristici trebuie să fie accesibile unui număr mare de utilizatori simultan, respectiv să permită accesul **concurențial** al acestora prin partajarea și utilizarea în comun a resurselor.

Prin intermediul aplicațiilor client, utilizatorii transmit cereri *HTTP* către serverul sistemului experimental. Pentru a onora o cerere din partea unui client serverul alocă resurse de memorie și procesor. În ceea ce privește comunicația între aplicațiile client și server, aceasta generează trafic în rețea.

Resursele disponibile serverului nu sunt nelimitate, iar în condițiile unei sarcini mari (respectiv a unui număr mare de utilizatori conectați simultan) timpul mediu de răspuns al sistemului (durata medie necesară serverului pentru a onora o cerere) poate crește. Drept urmare, pentru asemenea sisteme se impune o utilizare optimă a resurselor. O modalitate de optimizare constă în reducerea numărului de cereri transmise către server.

Deoarece demonstratorul implementează un sistem senzitiv la context, cea mai mare parte a traficului în rețea este realizat în scopul determinării contextului utilizatorilor.

În capitolul 2 al tezei de doctorat (subcapitolul 2.2) au fost definite 3 metode pentru determinarea contextului propuse de către autor, respectiv: MIBL, MDFC și MDCC. Conform rezultatelor experimentale prezentate în subcapitolul 2.2, metodele directe (MDFC și MDCC) conduc la un număr **mai mic** de cereri transmise în rețea comparativ cu cea indirectă.

Prezentul subcapitol evidențiază rezultatele unui nou set de 4 experimente în care au fost monitorizate 3 variabile, respectiv traficul generat în rețea, timpul de răspuns și resursele de memorie alocate de către serverul demonstratorului ca urmare a utilizării celor două metode directe. Deoarece numărul de utilizatori simultani este factorul principal de natură să influențeze cele 3 variabile monitorizate, acest număr este considerat mărime de intrare a sistemului pe durata experimentelor. De asemenea, având în vedere faptul că folosirea unor utilizatori reali (turiști cu aplicația mobilă instalată) nu a fost posibilă, intrarea a fost simulată prin intermediul unei singure aplicații client cu mai multe fire de execuției (respectiv unul pentru fiecare utilizator considerat). Codul sursă aferent acestei aplicații a fost scris de către autor în limbajul de programare *Java* și se regăsește în teza de doctorat. Referitor la evoluția intrării, în timpul simulărilor au fost folosite funcții de tip treapta și rampă.

Experimentele au avut în vedere răspunsul sistemului la două tipuri de intrări pentru cele două metode directe (respectiv MDFC și MDCC) după cum urmează:

- E1 – intrare treaptă în cazul folosirii metodei MDFC;
- E2 – intrare treaptă în cazul folosirii metodei MDCC;
- E3 – intrare rampă în cazul folosirii metodei MDFC;
- E4 – intrare rampă în cazul folosirii metodei MDCC.

Un aspect important cu privire la cele două metode (MDFC și MDCC) constă în faptul că diferențele între acestea se reflectă numai asupra modului de implementare al clienților. Indiferent de metoda folosita, componentele server ale demonstratorului nu necesită modificări.

În cazul metodei MDCC, numărul de cereri este redus prin alocarea și gestionarea de către client a unei zone de memorie de tip *cache*. În această zonă de memorie clientul păstrează rezultatele cererilor către server în scopul refolosirii ulterioare a acestora. Utilizarea unui *cache* local în cazul metodei MDCC previne descărcarea aceleași resurse de pe server de mai multe ori.

Pentru monitorizarea traficului generat în rețea pe durata experimentelor a fost folosit sistemul de analiză *Wireshark* (versiunea 3.0.3 pe 64 de biți pentru *Windows*). Referitor la resursele de memorie alocate de către server, acestea au fost monitorizate prin intermediul unui instrument dedicat (*NetBeans Profiler* care este disponibil în cadrul mediului de dezvoltare *NetBeans* versiunea 8.2 utilizat și pentru realizarea componentelor server ale demonstratorului).

În ceea ce privește valorile parametrilor folosiți în cadrul experimentelor, acestea sunt:

- pentru intrarea treaptă (experimentele E1 și E2) în cazul ambelor metode: $k = \overline{1 \dots 100}$, $t_k = k * \Delta t, \Delta t = 1s, U_k(t_k) = 100$;
- pentru intrarea rampă (experimentele E3 și E4) în cazul ambelor metode: $k = \overline{1 \dots 100}$, $t_k = k * \Delta t, \Delta t = 1s, U_k(t_k) = k$.

unde U reprezintă numărul utilizatorilor. Cu alte cuvinte, pentru intrarea treaptă, a fost menținut un număr constant de utilizatori (100 de utilizatori) în vecinătatea unui dispozitiv *beacon* pe toată durata experimentelor. Iar pentru intrarea rampă numărul utilizatorilor a fost variat în baza dreptei descrise prin $U_k(t_k) = k$.

Rezultatele celor patru experimente sunt evidențiate prin intermediul graficelor incluse în anexa A.3 a tezei de doctorat. Graficele din anexa A.3 nu sunt prezentate în cadrul acestui rezumat din considerente de spațiu. Cu titlu de exemplu, în figurile 4.25, 4.26, 4.27 și 4.28 sunt ilustrate 4 caracteristici care evidențiază răspunsul sistemului pentru experimentele E2 (metoda MDCC cu intrare treaptă) și E4 (metoda MDFC cu intrare rampă).

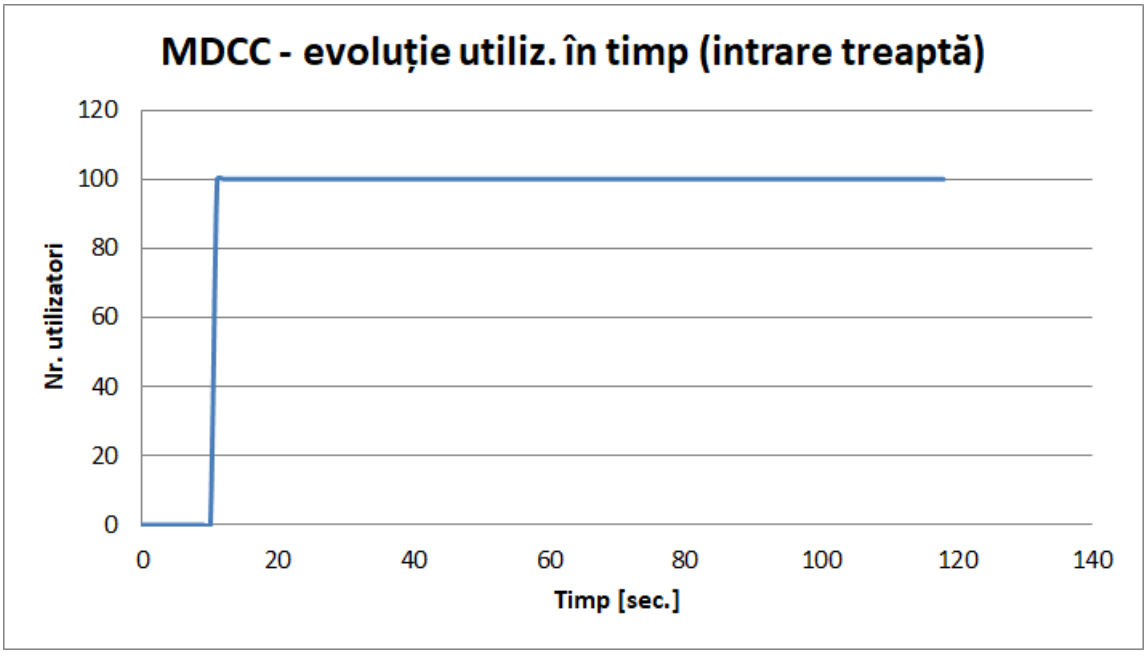

Fig. 4.25 MDCC – intrare treaptă

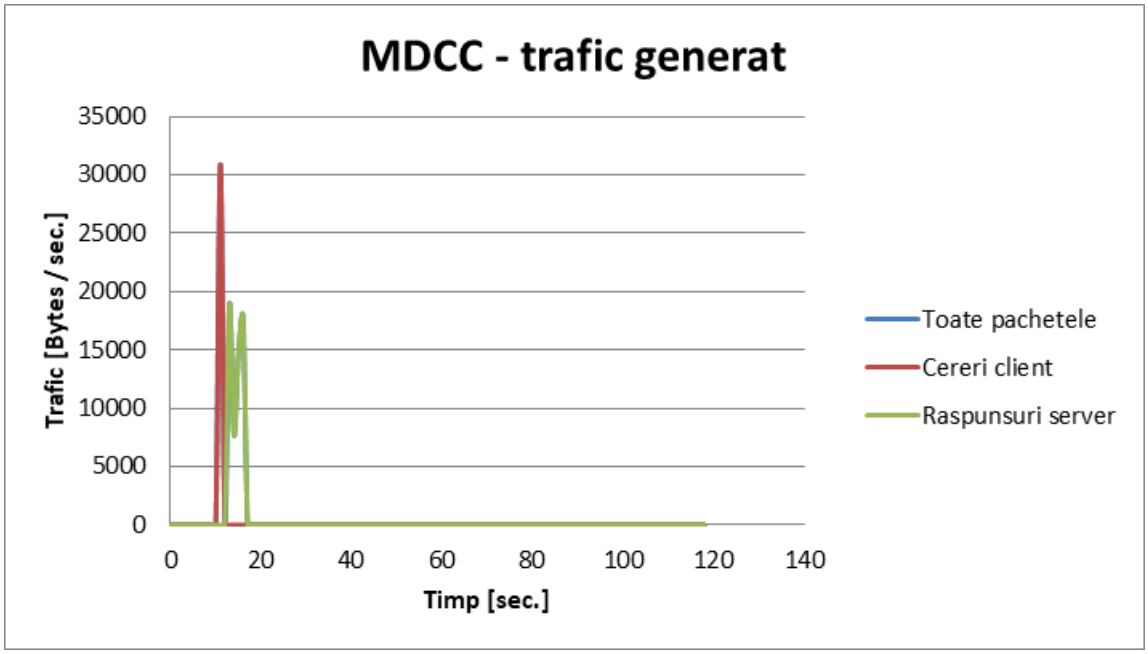

Fig. 4.26 MDCC – răspunsul sistemului experimental (trafic generat în rețea) pentru intrare treaptă

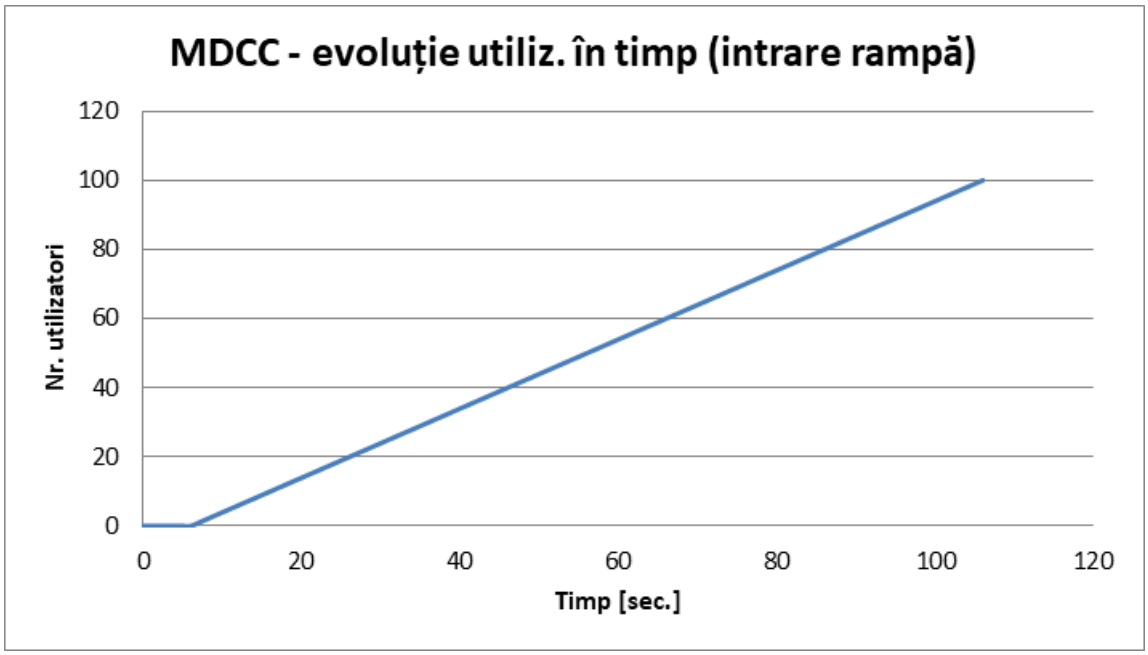

Fig. 4.27 MDCC – intrare rampă

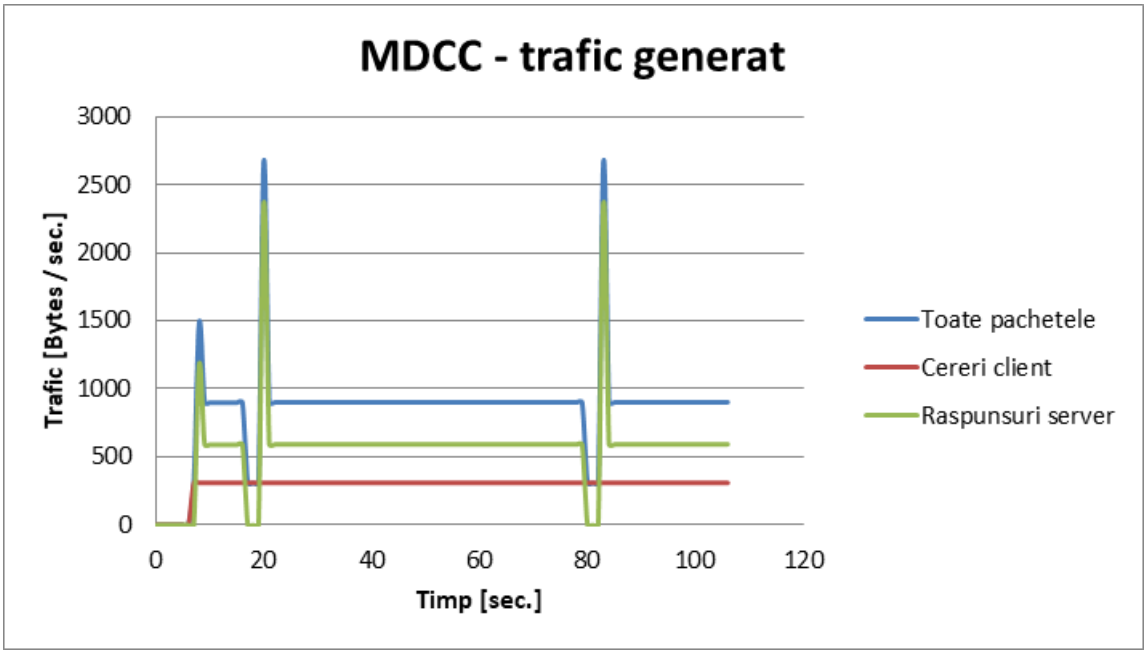

Fig. 4.28 MDCC – răspunsul sistemului experimental (trafic generat în rețea) pentru intrare rampă

În ceea ce privește dependența intrare – ieșire dintre numărul de utilizatori și traficul generat în rețea, atât pe baza caracteristicilor prezentate în figurile de mai sus (4.25, 4.26, 4.27 și 4.28) cât și a celor ilustrate în anexa A.3 a tezei de doctorat, se pot evidenția următoarele aspecte:

- în cazul utilizării metodei MDFC pentru o intrare treaptă răspunsul sistemul este tot o funcție treaptă;
- în cazul utilizării metodei MDCC pentru o intrare treaptă (figura **Error! Reference source not found.**) răspunsul sistemul este o funcție impuls (figura **Error! Reference source not found.**);
- în cazul utilizării metodei MDFC pentru o intrare rampă răspunsul sistemul este tot o funcție rampă;
- în cazul utilizării metodei MDCC pentru o intrare rampă (figura **Error! Reference source not found.**) răspunsul sistemul este o funcție treaptă (figura **Error! Reference source not found.**).

Cu alte cuvinte, făcând abstracție de câteva abateri punctuale care țin de latența transmisiei datelor în rețea și de timpul de răspuns al serverului, în absența unui *cache* local (respectiv în cazul metodei MDFC) comportamentul sistemului este asemănător cu al unui element **proporțional**. Pe de alta parte, prin utilizarea unei memorii locale de tip *cache* (respectiv în cazul metodei MDCC) comportamentul sistemului devine asemănător cu al unui element **derivativ**.

În ceea ce privește timpul de răspuns și cantitatea de memorie alocată de către server, conform graficelor din anexa A.3 a tezei de doctorat, acestea sunt **mai mici în cazul metodei MDCC** pentru ambele tipuri de intrări.

4.4. Concluzii parțiale

Capitolul 4 tratează probleme ce țin de proiectarea și realizarea aplicațiilor client ale demonstratorului. Aplicațiile client reprezintă interfața sistemului expert cu utilizatorii.

Demonstratorul integrează următoarele două aplicații client:

- Aplicația mobilă care este destinată turiștilor. Aceasta a fost dezvoltată în limbajul de programare *Java*, este compatibilă cu sistemul de operare *Android* și poate fi instalată pe terminalele de tip smartphone ale utilizatorilor pentru a fi folosită în timpul vizitelor la siturile culturale.
- Aplicația *WEB* care este destinată experților din domeniul patrimoniului cultural (de exemplu personalul care se ocupă de administrarea obiectelor de patrimoniu expuse în cadrul unui muzeu). Această aplicație poate fi accesată dintr-un *browser* de Internet compatibil și se bazează pe limbajele *Java*, *JavaScript* și *HTML*. Prin intermediul aplicației *WEB* pot fi definite obiectele de patrimoniu și scenariile în cadrul demonstratorului. Scenariile conțin cunoștințele experților furnizate sub forma regulilor de producție. Acestea sunt stocate în baza de date pentru a fi folosite ulterior de către motorul de inferență.

Pentru proiectarea celor două aplicații a fost utilizat limbajul *UML* descris în capitolul 1 (subcapitolul 1.3) al tezei de doctorat.

Dezvoltarea aplicațiilor a fost realizată folosind limbajele *JAVA* și *JavaScript*. Codul sursă aferent celor două aplicații este disponibil în arhiva (fișierul *arhiva_cod_sursa.rar*) de pe CD-ul ce însoțește teza de doctorat.

Capitolul 4 începe cu tratarea problemelor legate de proiectarea aplicației mobile. Astfel, este prezentată specificația aplicației mobile care este alcătuită din 7 cazuri de utilizare. Acestea sunt evidențiate prin intermediul unei diagrame *UML* (diagramă de cazuri de utilizare) și definite folosind șablonul adoptat în cadrul subcapitolului 1.3.

Aplicația mobilă joacă un rol important în funcționarea sistemului experimental. Prin intermediul aplicației mobile, demonstratorul monitorizează contextul utilizatorilor, iar pe baza modificărilor contextului, personajele digitale inițializează automat conversații cu utilizatorii pentru a le transmite informații despre obiectele de patrimoniu.

În vederea monitorizării contextului aplicația mobilă trebuie să funcționeze în modul interactiv. Drept urmare, capitolul 4 acordă o mare importanță cazului de utilizare *Pornește Modul Interactiv* prezentat în subcapitolul 4.1.1.4.

În cadrul aplicației mobile, modul interactiv este implementat prin intermediul serviciului SDDB (Serviciu Detecție Dispozitive *Beacon*). Schema logică care stă la baza funcționarii acestui serviciu este inclusă și descrisă în cadrul subcapitolul 4.1.1.4.

Aspectele referitoare la realizarea aplicației mobile (respectiv arhitectura aplicației, platforma și librăriile software folosite, mediul de dezvoltare utilizat etc.) sunt prezentate în subcapitolul 4.1.2.

Capitolul 4 continuă cu prezentarea aplicației *WEB*. Aceasta reprezintă interfață experților din domeniul patrimoniului cultural cu sistemul experimental și are 7 cazuri de utilizare. Atât cazurile de utilizare cât și relațiile dintre acestea sunt puse în evidență prin intermediul unei diagrame *UML*. Ca și în cazul aplicației mobile, pentru definirea cazurilor de utilizare este folosit tot șablonul adoptat în subcapitolul 1.3.

Dintre cazurile de utilizare ale aplicație *WEB* sunt de un deosebit interes cele care permit definirea obiectelor de patrimoniu (subcapitolul 4.2.1.2) și a scenariilor (subcapitolul 4.2.1.3). În cadrul demonstratorului, orice obiect de patrimoniu este însoțit de minim un scenariu. Prin

intermediul scenariului sunt definite datele personajului digital asociat obiectului de patrimoniu. Mai mult decât atât, scenariu include și setul de reguli în baza căruia se vor desfășura conversațiile între personajul digital și turiști. Mai specific, cele două cazuri de utilizare menționate anterior caracterizează modul în care sunt definite cunoștințele expertului uman în cadrul demonstratorului.

Aspectele referitoare la realizarea aplicației *WEB* (respectiv arhitectura acesteia, platforma și librăriile software folosite, mediul de dezvoltare utilizat etc.) sunt tratate în subcapitolul 4.1.2.

Capitolul 4 se încheie cu prezentarea unui nou set de experimente realizate în urma finalizării demonstratorului. Scopul experimentelor a constat în determinarea răspunsurilor demonstratorului pentru două tipuri de intrări (respectiv treaptă și rampă) în cazul utilizării metodelor MDFC și MDCC (definite în subcapitolul 2.2). În timpul experimentelor au fost monitorizate **traficul generat în rețea**, **timpul de răspuns** și **resursele de memorie** alocate de către serverul demonstratorului.

Deoarece numărul de utilizatori simultani este factorul principal de natură să influențeze cele 3 variabile monitorizate, acesta a fost considerat intrare a sistemului. De asemenea, având în vedere faptul că nu a fost posibilă folosirea unor utilizatori reali (turiști cu aplicația mobilă instalată), intrarea a fost simulată prin intermediul unei singure aplicații client cu mai multe fire de execuției (unul pentru fiecare utilizator considerat). Rezultatele experimentale au indicat o utilizare mai eficienta a resurselor și un timp de răspuns mai mic în cazul metodei MDCC pentru ambele tipuri de intrări. Aceste rezultate pot fi justificate prin utilizarea zonei de memorie de tip *cache* în cazul metodei MDCC.

Capitolul 5. Concluzii generale, contribuţii, diseminarea rezultatelor şi direcţii viitoare de cercetare

Acest ultim capitol al tezei de doctorat își propune să creeze o imagine de ansamblu asupra problemelor soluționate.

Pornind de la o sinteză a concluziilor raportate la obiectivele asumate (subcapitolul 5.1), trecând prin evidențierea contribuțiilor semnificative (subcapitolul 5.2) și diseminarea în publicații a rezultatelor obținute (subcapitolul 5.3), capitolul 5 se încheie cu enumerarea unor posibilități de continuare a cercetărilor inițiate în teza de doctorat (subcapitolul 5.4).

5.1. Concluzii generale

Acest subcapitol se constituie într-un rezumat al concluziilor întregii teze de doctorat. Concluziile, obiectivele și capitolele tezei se află în strânsă legătură. Drept urmare, sinteza prezentată în subcapitolul curent evidențiază corelația dintre acestea.

Obiectivele tezei de doctorat au fost enunțate în introducere. Acestea sunt:

O1. Realizarea de cercetări și investigații cu privire la posibilitatea dezvoltării unui sistem expert senzitiv la context, destinat furnizării de conținut interactiv turiștilor pe durata vizitelor la siturile culturale;

O2. Transpunerea în practică a rezultatelor aferente obiectivului O1 prin realizarea unui sistem experimental, respectiv a unui demonstrator pentru sistemul expert senzitiv la context propus.

Obiectivele propuse au fost îndeplinite.

În cele ce urmează sunt evidențiate concluziile referitoare la îndeplinirea obiectivului O1:

O1.1. A fost realizat un studiu cu privire la stadiul actual în domeniul sistemelor senzitive la context (subcapitolul 1.1). Ca urmare a studiului s-a determinat posibilitatea de utilizare a serviciilor *WEB* în cadrul sistemelor senzitive la context cu arhitectură client-server (subcapitolul 1.1.2) și au fost identificate dispozitivele de tip *beacon BLE*. Acestea pot fi folosite împreună cu terminalele mobile de tip *smartphone* în vederea determinării contextului utilizatorului. Totodată, a fost selectat modelul de *beacon IBKS 105* pentru a fi utilizat în cadrul demonstratorului. Caracteristicile tehnice ale acestuia sunt prezentate în subcapitolul 1.1.3.

O1.2. Au fost investigate mai multe motoare de inferență furnizate de către diferiți dezvoltatori (*Red Hat*, *ORACLE*, *OpenRules*, *MICROSOFT*, *IBM*) în vederea stabilirii posibilităților de integrare în cadrul demonstratorului propus spre realizare. Pe baza rezultatelor evidențiate în subcapitolul 1.2 a fost selectat motorul *Drools Expert*.

O1.3. A fost realizat un studiu al literaturii de specialitate cu privire la limbajul *UML* folosit ulterior pentru proiectarea și modelarea sistemului experimental. Ca urmare a studiului au fost identificate tipurile de diagrame *UML* adecvate pentru proiectarea demonstratorului. Acestea se regăsesc în subcapitolul 1.3.

O1.4. În vederea utilizării la realizarea sistemului experimental, au fost analizate mai multe metode de proiectare și dezvoltare software (*design-patterns*) precum *DAO*, *Singleton*, *Factory* etc. Caracteristicile acestora au fost evidențiate prin intermediul unor exemple propuse de către autor în subcapitolul 1.4.

O1.5. Au fost analizate 3 variante de implementare ale serviciilor *WEB* identificate anterior (vezi punctul O1.1). Pe baza timpului de răspuns, a fost selectata varianta *RESTful* / *JSON* pentru realizarea componentelor de comunicație în cadrul demonstratorului. Rezultatele experimentale obținute sunt disponibile în subcapitolul 2.1.

O1.6. Au fost propuse 3 metode pentru determinarea contextului (respectiv MIBL, MDCC, MDFC). Au fost analizate posibilitățile de integrare ale celor 3 metode cu bazele de date relaționale (subcapitolul 2.2.4) și cu serviciile *WEB* de tip *RESTful* / *JSON* (subcapitolul 2.2.5). Rezultatele experimentale au indicat un timp mult mai mare de răspuns în cazul metodei MIBL în raport cu celelalte două metode.

O1.7. A fost realizat un experiment în verdea determinării numărului de cereri și a cantității de date transmise în rețea pentru cele 3 metode menționate la punctul O1.6. Rezultatele experimentale incluse în subcapitolul 2.2.6, au evidențiat un trafic de date mai mic în cazul metodei MDCC față de metodele MDFC și MIBL. Pe baza acestor rezultate și a concluziilor anterioare (vezi punctul O1.6) a fost adoptată metoda MDCC pentru a fi folosită în cadrul demonstratorului.

Obiectivul O2 al tezei de doctorat a fost concretizat în mare parte prin realizarea sistemului experimental (respectiv a demonstratorului). Concluziile generale referitoare la îndeplinirea acestui obiectiv sunt enumerate în continuare:

O2.1. Au fost evidențiate caracteristicile generale ale sistemelor expert și modul de implementarea a acestora în cadrul demonstratorului (subcapitolul 3.1);

O2.2. A fost definit conceptul care stă la baza demonstratorului (subcapitolul 3.2);

O2.3. A fost definită arhitectura demonstratorului (subcapitolul 3.3). Demonstratorul combină arhitectura caracteristică sistemelor expert cu cea de tip client – server integrând totodată și componente relevante pentru îndeplinirea proprietății de senzitivitate la context;

O2.4. Au fost proiectate și realizate baza de date și componentele server ale demonstratorului (subcapitolul 3.4). Atât pentru activitățile de proiectare cât și pentru cele de realizare s-a pus un mare accent pe standardizare. Totodată, s-a urmărit minimizarea efortului necesar pentru dezvoltare prin folosirea: tehnicilor *ORM*, a tipurilor generice, a proprietăților moștenirii claselor, a arhitecturii *EJB* și a metodelor de *design software*.

O2.5. Au fost proiectate și realizate aplicațiile client ale demonstratorului (subcapitolele 4.1 și 4.2);

O2.6. În urma finalizării demonstratorului, au fost realizate 4 noi experimente. Pe durată experimentelor au fost monitorizate traficul generat în rețea, timpul de răspuns și resursele de memorie alocate de către serverul demonstratorului. Rezultatele și concluziile experimentelor se regăsesc în subcapitolul 4.3.

5.2. Contribuții originale ale tezei de doctorat

Pe parcursul tezei de doctorat au fost prezentate elemente de originalitate care se regăsesc cu precădere în capitolele 2, 3 și 4.

În cele ce urmează este prezentată o sinteză a contribuții cu o relevanță aparte.

1. A fost realizat un studiu cu privire la stadiul actual în domeniul sistemelor senzitive la context. Ca urmare a studiului au fost identificate aplicații, servicii și librării software relevante pentru acest domeniu precum: *Google Awareness API*, *Google Beacon Platform*, *Allrecipes Dinner Spinner*, *Waze* etc. Totodată, au fost remarcate dispozitivele de tip *beacon BLE* care sunt utilizate într-o multitudine de aplicații din domeniul sistemelor senzitive la context și *IoT*. Dintre modelele analizate a fost ales produsul *IBKS 105* compatibil cu profilul *Eddystone-UID* (definit de *Google*) pentru a fi folosit în cadrul demonstratorului.
2. Au fost propuse următoarele 3 metode pentru determinarea contextului: MIBL, MDFC și MDCC. Au fost prezentate modalitățile de integrare ale metodelor propuse (MIBL, MDFC și MDCC) cu bazele de date relaționale și serviciile de tip *WEB RESTful*. În acest sens au fost definite interogări *SQL* pentru metodele propuse. De asemenea, a fost realizat și un test pentru determinarea timpului de răspuns caracteristic interogărilor definite. În cadrul testului a fost folosit serverul *MySQL* și o tabelă populată cu 1.000.000 de înregistrări. Rezultatele testului au indicat un timp de răspuns mult mai mare în cazul interogării specifice metodei MIBL.
3. A fost realizat un studiu de caz în verdea determinării numărului de cereri și a traficului generat în rețea în cazul celor 3 metode propuse (MIBL, MDFC și MDCC). Ca urmare a studiului de caz a rezultat faptul că metoda MDCC generează cel mai mic număr de cereri transmise în rețea. Pe baza acestui rezultat și al concluziei de la punctul 2 a fost selectata metoda MDCC pentru a fi folosită în cadrul demonstratorului.
4. A fost realizat un studiu cu privire la servicii *WEB*. Au fost furnizate exemple de servicii *WEB* consacrate precum: *Product Advertising API* (*Amazon*), *Google Maps Static API*, *Twitter REST API* etc. De asemenea, a fost efectuat și un experiment în vederea determinării timpului de răspuns pentru următoarele 3 variante de servicii *WEB*: *SOAP*, *RESTful* / *XML*, *RESTful* / *JSON*. Pe baza rezultatelor experimentale a fost aleasă varianta *RESTful* / *JSON* pentru a fi folosită în scopul implementării modulelor de comunicație din cadrul demonstratorului.
5. A fost realizat un studiu cu privire la stadiul actual în domeniul sistemelor expert și al motoarelor de inferența bazate pe reguli de producție. Ca urmare a studiului au fost remarcate diferite domenii de aplicare și tipuri de probleme ce pot fi rezolvate cu ajutorul motoarelor de inferența. Totodată, au fost identificate mai multe motoare bazate pe reguli precum: *Drools Expert*, *Oracle Business Rules Engine*, *OpenRules Rule Solver*, *BizTalk Business Rules Engine*, *IBM ODM Decision Server Rules*. Dintre acestea a fost selectat motorul *Drools Expert* pentru a fi folosit la realizarea demonstratorului. Spre deosebire de celelalte variante, acesta este *open-source* și poate fi folosit independent (nu ca parte integrantă a unei suite de programe).
6. A fost realizat un studiu asupra metodelor de proiectare software (*design patterns*). În baza studiului au fost identificate câteva metode relevante pentru realizarea demonstratorului (ca de exemplu *DAO*, *MVVM*, *MVC*, *Singleton* etc.) și au fost evidențiate avantajele acestora

(respectiv reducerea timpului de execuție, utilizarea eficientă a memorie, diminuarea efortului de dezvoltare și întreținere etc.) prin intermediul unor exemple concrete de aplicare folosind limbajul de programare Java.

7. A fost definit conceptul care stă la baza demonstratorului.
8. A fost definită arhitectura software a demonstratorului.
9. A fost proiectată și realizată baza de date a demonstratorului. Modelul bazei de date a fost definit prin intermediul unei diagrame *ER*.
10. A fost proiectat și realizat domeniul demonstratorului (domeniul cunoașterii și domeniul persistent). Proiectarea / modelarea domeniului a fost realizată prin diagrame de clase *UML*.
11. Au fost proiectate și realizate componentele server (componenta de acces la cunoaștere, componentele de acces la date, componenta de autentificare a utilizatorilor, serviciile *WEB*) ale demonstratorului. Proiectarea / modelarea componentelor server a fost realizată prin diagrame *UML* de clase și de secvențe.
12. Au fost definite cazurile de utilizare pentru aplicația mobilă a demonstratorului.
13. A fost realizată aplicația mobilă a demonstratorului.
14. Au fost definite cazurile de utilizare pentru aplicația *WEB* a demonstratorului.
15. A fost realizată aplicația *WEB* a demonstratorului.
16. Au fost realizate 4 experimente în vederea determinării răspunsului demonstratorului la două tipuri de intrări (treaptă și rampă) în cazul utilizării metodelor MDFC și MDCC. În timpul experimentelor au fost monitorizate traficul generat în rețea, timpul de răspuns și resursele de memorie alocate de către serverul demonstratorului. Rezultatele experimentale au indicat o utilizare mai eficienta a resurselor și un timp de răspuns mai mic în cazul metodei MDCC pentru ambele tipuri de intrări. Totodată, s-au constatat următoarele:
    a. În lipsa utilizării unui *cache* de către client (cazul metodei MDFC) răspunsul sistemului este proporțional;
    b. Prin utilizarea unui *cache* de către client (cazul metodei MDFC) răspunsul sistemului este derivativ.

5.3. Diseminarea rezultatelor cercetării

Rezultatele obținute de către autor în domenii relevante tezei de doctorat au fost diseminate prin participarea la conferințe și sesiuni de comunicări științifice și prin publicarea de articole dintre care sunt de menționat:

1. *Sistem expert senzitiv la context pentru obiecte de patrimoniu*
   Bărbos M.
   Sesiunea de Comunicări a Cercurilor Științifice Studențești în Domeniile - Inginerie Electrica, Electronica, Control și Calculatoare – SCCSS-IEECC, Sibiu - 5-6 iunie 2019
   http://www.sccss.ro
2. *CH-AVATAR: A Context-Aware Platform for an Adaptive Cultural Heritage Experience*
   Bărbos M.
   The 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 21-23 septembrie, 2017, București, România
   DOI: 10.1109/IDAACS.2017.8095197
   https://ieeexplore.ieee.org/document/8095197/

Indexat în *Clarivate Analytics Web of Science*
Indexat în *Elsevier Scopus*

3. *UbiPOL: A Platform for Context-Aware Mobile Device Applications in Policy Making*
Bărbos M., Lee H., Pop E., Campos L.M.
UBICOMM 2011, The 5th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, 20-25 Noiembrie, 2011, Lisabona, Portugalia
ISBN-13: 978-161208171, *IARIA XPS Press*
https://www.thinkmind.org/download.php?articleid=ubicomm_2011_11_20_10113
Indexat în *Elsevier Scopus*
4. *E-services Access System Using Wireless Communications Networks*
Pop E.**, Bărbos M.**
IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR), 28-30 Mai, 2010, Cluj-Napoca, România
DOI: 10.1109/AQTR.2010.5520801
https://ieeexplore.ieee.org/document/5520801/
Indexat în *Clarivate Analytics Web of Science*
Indexat în *Elsevier Scopus*
5. *Ubiquitous Participation Platform for POLicy Makings (UbiPOL): A Research Note*
Irani Z., Lee H., Weerakkody V., Kamal M., Topham S., Simpson G., Balci A., Medeni T.D., Gábor A., Kő A., Küçükpehlivan A., Dabanli A., Sağiroğllu C., Saygin Y., Nergiz E., Hintoglu A., Campos L.M., Correia P., Luis J.P.S., Rebahi Y., Onofrei A.A., Pop E., **Bărbos M.**, Iancu M.
International Journal of Electronic Government Research, Volume 6, Issue 1, 2010
DOI: 10.4018/jegr.2010102006
https://www.igi-global.com/viewtitlesample.aspx?id=38965&ptid=38827&t=Ubiquitous%20Participation%20Platform%20for%20POLicy%20Makings%20(UbiPOL):%20A%20Research%20Note
Indexat în *Elsevier Scopus*
6. *WEB Services for Ubiquitous Mobile Device Applications*
Bărbos M., Pop E.
SERVICE COMPUTATION 2010 : The 2nd International Conferences on Advanced Service Computing, 21-26 Noiembrie, 2010, Lisabona, Portugalia
ISBN-13: 978-1612081052, *IARIA XPS Press*
https://www.thinkmind.org/download.php?articleid=service_computation_2010_3_10_20098
Indexat în *Clarivate Analytics Web of Science*
7. *Client Server System for e-Services Providing in Mobile Communications Networks*
Pop E., **Bărbos M.**, Lupu R.
World Congress on Engineering 2008, 2-4 Julie, 2008, Londra, Anglia
ISBN-13: 978-9881701244, *Newswood Academic Publications*
Indexat în *Clarivate Analytics Web of Science*
8. *Technical overview on designing wireless remote control steering mechanisms for small ships and scaled model ships*
Bărbos M., Cristescu C.
2008 IEEE International Conference on Automation, Quality and Testing, Robotics, 22-25 Mai 2008, Cluj-Napoca, România
DOI: 10.1109/AQTR.2008.4588929
https://ieeexplore.ieee.org/document/4588929/
Indexat în *Clarivate Analytics Web of Science*
Indexat în *Elsevier Scopus*
Citat în brevetul internațional US9073480B2
(https://patents.google.com/patent/US9073480)

5.4. Direcţii posibile de continuare a cercetărilor

Demonstratorul realizat de către autor pe durata stagiului doctoral implementează funcțiile sistemului expert senzitiv la context pentru obiecte de patrimoniu și se afla la convergența mai multor domenii prioritare de cercetare. Acesta integrează tehnologii din domeniul inteligenței artificiale și al sistemelor senzitive la context pentru furnizarea de conținut interactiv turiștilor în timpul vizitelor la siturile culturale.

În continuare sunt evidențiate câteva posibile direcții de continuare a cercetărilor din prezenta teză de doctorat:

1. Integrarea unor tehnologii *TTS* (*Text-to-Speech*): În cadrul demonstratorului, informațiile despre obiectele de patrimoniu sunt furnizate sub forma unor mesaje text care trebuie citite de către utilizator. Prin intermediul tehnologiilor *TTS* sistemul ar putea transforma textul din cadrul mesajelor în vorbire.
2. Extinderea tipului de conținut furnizat: Pe lângă mesajele text, sistemul ar putea furniza informații turiștilor și prin intermediul altor mijloace multimedia (imagini, fișiere audio-video, etc.). Acest lucru ar presupune extinderea modelului relațional în scopul acomodării noilor tipuri de date.
3. Modelarea relațiilor de precedență dintre taskurile scenariilor prin utilizarea mai multor tipuri de grafuri: În cadrul demonstratorului, interacțiunea dintre un turist și un personaj digital a fost modelată prin intermediul unui arbore orientat. În timpul interacțiunii, turistul poate influența modul în care este parcurs arborele, însă nu poate reveni asupra unei decizii anterioare. Utilizarea altor tipuri de graf ar putea oferi mai multă flexibilitate din acest punct de vedere.
4. Integrarea unor tehnologii din domeniul realității augmentate: În opinia autorului, un sistem de tipul demonstratorului ar putea fi îmbunătățit prin dezvoltarea unui client compatibil cu dispozitive de tip *smart-glasses* (ca de exemplu *Glass* produs de *Google*). Asemenea dispozitive facilitează realitatea augmentată modificând în mod artificial percepția vizuală a utilizatorului asupra obiectelor din mediul înconjurător. Utilizarea acestor dispozitive în cadrul unui sistem de tipul demontatorului ar permite suprapunerea de informații suplimentare despre obiectele de patrimoniu în câmpul vizual al turistului.
5. Utilizarea tabelelor de decizie: Regulile de producție folosite de către motorul de inferență din cadrul demonstratorului sunt stocate într-o bază de date. În general, sintaxa regulilor de producție este apropiată de limbajul uman, acestea fiind ușor de înțeles de către experți din diferite domenii. Cu toate acestea, atunci când este posibil, ar fi mai indicată utilizarea tabelelor de decizie pentru definirea cunoștințelor. Tabele de decizie au avantajul de a fi mai accesibile pentru personalul fără competențe de programare deoarece acestea pot fi definite în *Excel*. În cazul folosirii tabelelor de decizie, regulile nu vor mai fi stocate într-o baza de date acestea fiind salvate pe server în fișiere *Excel*.

Bibliografie

1. What is Cultural Heritage. *CiD - Culture in Development.* [Interactiv] [Citat: 23 03 2019.] http://www.cultureindevelopment.nl/cultural_heritage/what_is_cultural_heritage.

2. **International Council on Monuments and Sites - ICOMOS** Charters adopted by the general assembly of ICOMOS. [Interactiv] [Citat: 23 03 2019.] https://www.icomos.org/images/DOCUMENTS/Charters/INTERNATIONAL_CULTURAL_TOURISM_CHARTER.pdf.

3. **Bărbos, M.** *CH-AVATAR: A Context-Aware Platform for an Adaptive Cultural Heritage Experience.* București : IEEE, 2017. DOI: 10.1109/IDAACS.2017.8095197.

4. **Abowd, G. D. și Dey, A. K.** *Towards a better understanding of context and context-awareness.* Londra : Springer, 1999. HUC'99 - The first international symposium on Handheld and Ubiquitous Computing. ISBN:3-540-66550-1.

5. **Bărbos, M., și alții, și alții.** *UbiPOL: A Platform for Context-aware Mobile Device Applications in Policy Making.* Lisabona : IARIA XPS Press, 2011. UBICOMM 2011 - The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. ISBN:978-161208171-7.

6. **Gu, Tao, și alții, și alții.** *An Ontology-based Context Model in Intelligent Environments.* 2004. PROCEEDINGS OF COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE.

7. **Schilit, B. N. și Theimer, M. M.** *Disseminating active map information to mobile hosts.*, s.l. : IEEE, 1994, IEEE Network, Vol. 8. DOI: 10.1109/65.313011.

8. **Virrantaus, K., și alții, și alții.** *Developing GIS-supported location-based services.* s.l. : IEEE, 2001. Proceedings of the Second International Conference on Web Information Systems Engineering. ISBN: 0-7695-1393-X.

9. **Pop, E. și Bărbos, M.** E-services Access System Using Wireless Communications Networks,. *Proceedings of 2010 IEEE International Conference on Automation, Quality and Testing, Robotics.* s.l. : IEEE, 2010. ISBN-13: 978-1424467228.

10. **Pop, E., Bărbos, M. și Lupu, R.** Client Server System for E-Services Mobile Access in Business Environment. *Proceedings of the IADIS International Conference on e-Commerce 2008.* s.l. : IADIS Press, 2008.

11. **World Wide Web Consortium (W3C)** Web Services Glossary. [Interactiv] W3C, 2004. [Citat: 26 03 2019.] https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/.

12. **Richardson, L. și Ruby, S.** *RESTful Web services.* s.l. : O'Reilly Media, 2007. ISBN 9780-596-52926-0.

13. **Bărbos, M. și Pop, E.** WEB Services for Ubiquitous Mobile Device Applications. *The 2nd International Conferences on Advanced Service Computing.* s.l. : IARIA XPS Press, 2010.

14. **Pointr Labs Ltd.** Beacons: Everything you need to know. *Pointr Labs Web site.* [Interactiv] [Citat: 27 03 2019.] https://www.pointrlabs.com/posts/beacons-everything-you-need-to-know/.

15. **Bluetooth SIG Inc.** Bluetooth Radio Versions. *Bluetooth SIG Web site.* [Interactiv] Bluetooth SIG Inc. [Citat: 27 03 2019.] https://www.bluetooth.com/bluetooth-technology/radio-versions.

16. *Physical web.* [Interactiv] [Citat: 27 03 2019.] https://google.github.io/physical-web/.

17. **Statler, S.** *Beacon Technologies: The Hitchhiker's Guide to the Beacosystem.* s.l. : Apress, 2016. ISBN:978-1-4842-1889-1.

18. **Bluetooth SIG Inc.** Bluetooth Market Update 2018. *Bluetooth SIG Web site.* [Interactiv] Bluetooth SIG Inc. [Citat: 04 04 2019.] https://www.bluetooth.com/markets/market-report.
19. **Accent Systems.** Connecting the world with bluetooth beacons. [Interactiv] [Citat: 01 04 2019.] https://accent-systems.com/beacons/.
20. **Dumitrache, I.** *Ingineria Reglării Automate.* s.l. : Editura Politehnica Press, 2005. ISBN-10: 973-8449723.
21. **Jackson, P.** *Introduction To Expert Systems 3rd edition.* Boston : Addison-Wesley Longman Publishing Co., 1998. ISBN:0201876868.
22. **Dumitrache, I.** *Automatica, Volumul II.* s.l. : Editura Academie Române, 2009. ISBN-13: 978-9732718827.
23. **Frederick, H.R., Waterman, D. și Lenat, D.** *Building Expert Systems.* s.l. : Addison Wesley, 1983. ISBN 0-201-10686-8.
24. **Oracle.** Fusion Middleware User's Guide for Oracle Business Rules. [Interactiv] [Citat: 01 04 2019.] https://docs.oracle.com/cd/E12839_01/user.1111/e10228.pdf.
25. **Drools Dev. Team.** Drools Expert User Guide. [Interactiv] [Citat: 01 04 2019.] http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf.
26. **OpenRules, Inc.** RULE SOLVER - Constraint Programming with OpenRules 6.3.1 - USER MANUAL. [Interactiv] [Citat: 01 04 2019.] http://openrules.com/pdf/RulesSolver.UserManual.pdf.
27. **Microsoft.** BizTalk Server - Rule Engine. [Interactiv] [Citat: 2019 04 01.] https://docs.microsoft.com/en-us/biztalk/core/rule-engine.
28. **IBM.** IBM ODM - Engine execution modes - RetePlus mode. [Interactiv] [Citat: 01 04 2019.] https://www.ibm.com/support/knowledgecenter/en/SS7J8H/com.ibm.odm.dserver.rules.designer.run/optimizing_topics/con_opt_execmodes_reteplus.html.
29. **OptaPlanner Team.** OptaPlanner User Guide. [Interactiv] [Citat: 03 04 2019.] https://docs.optaplanner.org/7.4.1.Final/optaplanner-docs/pdf/index.pdf.
30. **Queen's University of Belfast.** ITC 2007 - Track 3 - Curriculum Course Scheduling - The Problem Model. [Interactiv] [Citat: 04 04 2019.] http://www.cs.qub.ac.uk/itc2007/curriculmcourse/course_curriculm_index_files/problemmodel.htm.
31. **Queen's University of Belfast.** ITC 2007 - Track 1 - Examination Timetabling - The Problem Model. [Interactiv] [Citat: 04 04 2019.] http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index_files/examproblemmodel.htm.
32. **Société française de Recherche Opérationnelle et Aide à la Décision - ROADEF.** ROADEF/EURO Challenge 2012: Machine Reassignment. [Interactiv] [Citat: 04 04 2019.] http://www.roadef.org/challenge/2012/files/problem_definition_v1.pdf.
33. **Weilkiens, T.** *Systems Engineering with SysML/UML: Modeling, Analysis, Design.* s.l. : Elsevier Inc., 2007. ISBN-13: 978-0123742742.
34. **Ionita, A. D.** *Modelarea UML în ingineria sistemelor de programe.* s.l. : ALL, 2003. ISBN-10: 9735714639.
35. **Zito, A., Dingel, J. și Diskin, Z.** *Package Merge in UML 2: Practice vs. Theory?, Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006 Genova, Italy, October 1-6, 2006 Proceedings.* s.l. : Springer, 2006. ISBN-13: 978-3540457725.
36. **Lavagno, L., Martin, G. și Selic, B.V.** *UML for Real: Design of Embedded Real-Time Systems.* s.l. : Springer, 2003. ISBN-13: 978-1402075018.

37. **Douglass, B. P.** *Real-Time UML Workshop for Embedded Systems.* s.l. : Elsevier Inc., 2007. ISBN-13: 978-0750679060.
38. **Pilone, D. și Pitman, N.** *UML 2.0 IN A NUTSHELL : A Desktop Quick Reference.* s.l. : O'Reilly Media, 2009. ISBN-13: 978-0596007959.
39. **Grässle, P., Baumann, H. și Baumann, P.** *UML 2.0 in Action: A Project-Based Tutorial.* s.l. : Packt Publishing, 2005. ISBN-10: 1904811558.
40. **Holt, J.** *UML for Systems Engineering.* s.l. : Institution of Engineering and Technology, 2004. ISBN-13: 978-0863413544.
41. **Cockburn, A.** *Writing Effective Use Cases.* s.l. : Addison-Wesley, 2000. ISBN-13: 978-0201702255.
42. **Gamma, Erich, et al., et al.** *Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley Professional, 1994. ISBN-13: 978-0-201-63361-0.
43. **Jeanne, Boyarsky și Scott, Selikoff.** *Oracle Certified Professional Java SE 8 Programmer II Study Guide.* s.l. : Sybex, 2015. ISBN-13: 978-1-119-06790-0.
44. **Kamalmeet, Singh, Adrian, Ianculescu și Lucian-Paul, Torje.** *Design Patterns and Best Practices in Java: A comprehensive guide to building smart and reusable code in Java 1st Edition.* s.l. : Packt Publishing, 2018. ISBN-13: 978-1-786-46359-3.
45. **M., Keith și M., Schincariol.** *Pro JPA 2: Mastering the Java Persistence API.* s.l. : Apress, 2010. ISBN-13: 978-1-4302-1956-9.
46. **Bloch, Joshua.** *Effective Java Second Edition.* s.l. : Addison-Wesley, 2008. ISBN-13: 978-0-321-35668-0.
47. **Oracle.** Core J2EE Patterns - Data Access Object. *Oracle Web Site.* [Interactiv] [Citat: 20 Martie 2019.] https://www.oracle.com/technetwork/java/dataaccessobject-138824.html.
48. **Panda, Debu, Rahman, Reza și Lane, Derek.** *EJB 3 in Action.* s.l. : Manning Publications, 2007. ISBN 1-933988-34-7.
49. **Barbeau, S.J. și Saloranta, T.** *Performance evaluation of transit data formats on a mobile device, 21st World Congress on Intelligent Transport Systems, ITSWC 2014: Reinventing Transportation in Our Connected World.* Detroit : s.n., 2014.
50. **Barbeau, S.J., și alții, și alții.** *A location-aware framework for intelligent real-time mobile applications, IEEE Pervasive Computing.* s.l. : IEEE, 2011, Vol. 10. DOI: 10.1109/MPRV.2010.48.
51. *Transportation Research Record: Journal of the Transportation Research Board.* **Barbeau, S.J., Wint, P.L. și Georggi, N.L.** 1, s.l. : SAGE Publishing, 2010, Vol. 2143. doi:10.3141/2143-21.
52. **Jones, C.B.** *Geographical Information Systems and Computer Cartography.* s.l. : Routledge, 1997. ISBN-13: 978-0582044395.
53. **Delbourgo, R.** *Trigonometry: Notes, Problems And Exercises.* s.l. : WS Professional, 2017. ISBN-13: 978-9813207103.
54. **Chen, C.J.** *Physics of Solar Energy.* s.l. : John Wiley & Sons, 2011. ISBN-13: 978-1118044599.
55. **Lawhead, J.** *Learning Geospatial Analysis with Python Second Edition.* s.l. : Packt Publishing, 2015. ISBN-13: 978-1783552429.
56. **Harrison, G. și Feuerstein, S.** *MySQL Stored Procedure Programming.* s.l. : O'Reilly Media, 2006. ISBN-13: 978-0596100896.
57. **Schwartz, B., Zaitsev, P. și Tkachenko, V.** *High Performance MySQL Third Edition.* O'Reilly Media : s.n., 2012. ISBN-13: 978-1449314286.
58. **Bradford, R.** *Effective MySQL: Optimizing SQL Statements.* s.l. : McGraw-Hill, 2011. ISBN-13: 978-0071782807.

59. **Cabral, S. și K. Murphy.** *MySQL Administrator's Bible.* s.l. : Wiley Publishing, 2009. ISBN-13: 978-0470416914.
60. **McLaughlin, M.** *MySQL Workbench: Data Modeling & Development.* s.l. : McGraw-Hill, 2013. ISBN-13: 978-0071791892.
61. **Oracle.** MySQL 8.0 Reference Manual. [Interactiv] [Citat: 27 09 2019.] https://dev.mysql.com/doc/refman/8.0/en/.
62. **Mamoulis, N.** *Spatial Data Management.* 2012 : Morgan & Claypool. ISBN-13: 978-1608458325.
63. **Urbano, F. și Basille, M.** Spatial is not Special: Managing Tracking Data in a Spatial Database. *Spatial Database for GPS Wildlife Tracking Data: A Practical Guide to Creatinga Data Management System with PostgreSQL/PostGIS and R.* s.l. : Springer, 2014.
64. **Chen, R. și Xie, J.** Open Source Databases and Their Spatial Extensions. *Open Source Approachesin Spatial Data Handling.* s.l. : Springer, 2008.
65. **EJB 3.1 Expert Group.** JSR-318 Enterprise JavaBeans 3.1 Final Release Specification. *Java Community Process.* [Interactiv] 05 09 2009. [Citat: 24 07 2019.] https://download.oracle.com/otndocs/jcp/ejb-3.1-fr-eval-oth-JSpec.
66. **Schincariol, M. și Keith, M.** *Pro JPA 2, Second Edition: A definitive guide to mastering the Java Persistence API.* s.l. : Apress, 2013. ISBN-13: 978-1430249269.
67. **Keith, M., Schincario, M. and Nardone, M.** *Pro JPA 2 in Java EE 8,Third Edition: An In-Depth Guide to Java Persistence APIs.* s.l. : Apress, 2018. ISBN-13: 978-1484234198.
68. **Goncalves, A.** *Beginning Java EE 6 Platform with GlassFish 3, Second Edition.* s.l. : Apress, 2010. ISBN-13: 978-1430228899.
69. **Yang, D.** *Java Persistence with Jpa 2.1.* s.l. : Outskirts Press, 2013. ISBN-13: 978-1478700197.
70. **Rubinger, A. L. and Burke, B.** *Enterprise JavaBeans 3.1, Sixth Edition: Developing Enterprise Java Components.* s.l. : O'Reilly Media, 2010. ISBN-13: 978-0596158026.
71. **Wetherbee, J., et al., et al.** *Beginning EJB 3: Java EE7 Edition.* s.l. : Apress, 2013. ISBN-13: 978-1430246923.
72. **Salatino, M., De Maio, M. și Aliverti, E.** *Mastering JBoss Drools 6.* s.l. : Packt Publishing, 2016. ISBN-13: 978-1783288625.
73. **Bali, M.** *Drools JBoss Rules 5.X Developer's Guide.* s.l. : Packt Publishing, 2013. ISBN-13: 978-1782161264.
74. **Amador, L.** *Drools Developer's Cookbook.* s.l. : Packt Publishing, 2012. ISBN-13: 978-1849511964.
75. **Ary, J.** *Instant Drools Starter.* s.l. : Packt Publishing, 2013. ISBN-13: 978-1782165545.
76. **Browne, P.** *JBoss Drools Business Rules.* s.l. : Packt Publishing, 2009. ISBN-13: 978-1847196064.
77. **Bârză, S. și Morogan, L. M.** *Algoritmica grafurilor.* s.l. : Fundatia Romania de Maine, 2008. ISBN-13: 978-9731631479.
78. **Bondy, J.A. și Murty, U.S.R.** *Graph Theory.* s.l. : Springer, 2008. ISBN-13: 978-1846289699.
79. **Paraschiv, N. și Popescu, M.** *Sisteme distribuite de supervizare și control.* s.l. : Editura Universității Petrol-Gaze din Ploiești, 2014. ISBN-13: 978-9737195579.
80. **Jendrock, E., și alții, și alții.** *The Java EE 6 Tutorial Fourth Edition.* s.l. : Addison-Wesley, 2010. ISBN-13: 978-0137081851.
81. **Sikora, M.** *EJB 3 Developer Guide.* s.l. : Packt Publishing, 2008. ISBN-13: 978-1847195609.
82. **Reese, R. M.** *EJB 3.1 Cookbook.* s.l. : Packt Publishing, 2011. ISBN-13: 978-1849682381.
83. **Yener, M. și Theedom, A. Theedom.** *Professional Java EE Design Patterns.* s.l. : John Wiley & Sons, Inc., 2015. ISBN-13: 978-1118843413.

84. **Heffelfinger, D. R.** *Java EE 7 Development with NetBeans 8.* s.l. : Packt Publishing, 2015. ISBN-13: 978-1783983520.
85. **Sriganesh, R. P., Brose, G. și Silverman, M.** *Mastering Enterprise JavaBeans 3.0.* s.l. : Wiley Publishing, Inc., 2006. ISBN-13: 978-0471785415.
86. **Fielding, R. T.** Teză de doctorat. *Architectural Styles and the Design of Network-based Software Architectures.* s.l. : UCI - UNIVERSITY OF CALIFORNIA IRVINE, 2000.
87. **Brock, J, Gupta, A. și Wielenga, G.** *Java EE and HTML5 Enterprise Application Development.* s.l. : McGraw-Hill Education, 2014. ISBN-13: 978-0071823142.
88. **Patni, S.** *Pro RESTful APIs: Design, Build and Integrate with REST, JSON, XML and JAX-RS.* s.l. : Apress, 2017. ISBN-13: 978-1484226643.
89. **Kalali, M. și Mehta, B.** *Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON.* s.l. : Packt Publishing, 2013. ISBN-13: 978-1782178125.
90. **Burke, B.** *RESTful Java with JAX-RS 2.0: Designing and Developing Distributed Web Services, Second Edition.* s.l. : O'Reilly Media, 2013. ISBN-13: 978-1449361341.
91. **Gulabani, S.** *Developing RESTful Web Services with Jersey 2.0.* s.l. : Packt Publishing, 2014. ISBN-13: 978-1783288298.
92. **Google.** Services overview. *Android Developers.* [Interactiv] [Citat: 5 10 2019.] https://developer.android.com/guide/components/services.
93. **Meier, R. și Lake, I.** *Professional Android Fourth Edition.* s.l. : John Wiley & Sons, 2018. ISBN-13: 978-1118949528.
94. **Mainkar, P.** *Expert Android Programming.* s.l. : Packt Publishing, 2017. ISBN-13: 978-1786468956.
95. **Google.** Guide to app architecture. *Android Developers.* [Interactiv] [Citat: 1 11 2019.] https://developer.android.com/jetpack/docs/guide.
96. **Darwin, I.** *Android Cookbook - Problems and Solutions for Android Developers.* s.l. : O'Reilly Media, 2017. ISBN-13: 978-1449374433.
97. **Stroud, A.** *Android Database Best Practices.* s.l. : Addison-Wesley, 2016. ISBN-13: 978-0134437996.
98. **Smyth, N.** *Android Studio 3.0 Development Essentials – Android 8 Edition.* s.l. : Payload Media, 2017. ISBN-13: 978-1977540096.
99. **Deitel, P., Deitel, H. și Wald, A.** *Android 6 for Programmers An App-Driven Approach Third Edition.* s.l. : Pearson Education, 2016. ISBN-13: 978-0134289366.
100. **Simpson, J.E.** *XPath and XPointer Locating Content in XML Documents.* s.l. : O'Reilly Media, 2002. ISBN-13: 978-0596002916.
101. **Williams, I.** *Beginning XSLT and XPath Transforming XML Documents and Data.* s.l. : Wiley Publishing, 2009. ISBN-13: 978-0470477250.
102. **Kay, M.** *XSLT 2.0 and XPath 2.0 4th Edition.* s.l. : Wiley Publishing, 2008. ISBN-13: 978-0470192740.