Business Information Systems Research Center
Centrul de cercetări in Informatică Economică

Proiect PN-III-P2-2.1-PED-2016-1140

ENTERKNOW:

Aplicație ENTERPRISE-Aware bazată pe reprezentarea hibridă și formală a cunoștințelor întreprinderii

Director proiect: **Prof. dr. Robert Andrei Buchmann**

Membri: Lect. Dr. Ana-Maria Ghiran, Mihai Cinpoeru, Alisa Harkai

Rezumat Etapa 2017

Prezență on-line:

http://austria.omilab.org/psm/content/enterknow/info

(în portalul comunității OMiLAB - Open Models Initiative Laboratory)

http://enterknow.granturi.ubbcluj.ro

1. FORMULAREA PROBLEMEI

Convergența dintre practicile de *dezvoltare agilă* a aplicațiilor software și *paradigma de dezvoltare bazată pe modele* ("Model-Driven Engineering") a condus în mod natural către metodologia modelării agile. Totuşi, în mod tradiţional modelarea conceptuală porneşte de la prezumţia că limbajul de modelare este unul fix, stabil, oferind un *spațiu semantic standardizat*. La baza prezumției se află considerentul că toate cerințele de dezvoltare a aplicațiilor software ar trebui să fie subordonate conceptual acestui spațiu semantic, de regulă cel oferit de limbaje orientate spre ingineria software (UML).

În cercetarea de faţă inversăm această subordonare, poziţionând limbajul şi instrumentul de modelare în subordinea cerinţelor sistemelor model-driven. Framework-ul angajat, AMME - *Agile Modelling Method Engineering* (Karagiannis, 2015) - permite acest lucru din perspectiva "*modelarea este reprezentare de cunoștințe"*, în sensul că un limbaj de modelare poate fi adaptat pentru a capta prin mijloace diagramatice acele cunoștințe legate de specificul întreprinderii sau domeniului de activitate care sunt relevante pentru artefactele software ce se vor implementa pe baza modelelor. În consecinţă, instrumentul de modelare devine unul de construire a unor reţele semantice cu manifestare grafică, reţele ce pot fi exportate în baze de cunoştinţe pe care le numim aici "hibride" pentru a sublinia integrarea de surse de informaţii eterogene: conţinut al diagramelor create cu limbajul de modelare agil, date relevante execuţiei (din surse moştenite sau deschise), legături semantice între elemente modelate şi date sau resurse relevante execuţiei, reguli/axiome pentru execuţia de inferenţe la nivelul bazei de cunoştinţe.

Proiectul are ca obiectiv implementarea unui demonstrator ce foloseşte o astfel de bază de cunoştinţe hibride în loc de bază de date. Ca rezultat generalizabil, se urmăreşte şi consolidarea unei metode de inginerie software inovatoare derivată din această abordare. Deoarece prin metoda propusă parametrizarea semantică a artefactelor software înlocuiește practicile model-driven tradiționale (de ex., generarea de cod), devine necesară și o revizuire a ciclurilor de dezvoltare consacrate ce se bazau pe relația strânsă între modele conceptuale UML și schema unor baze de date relaționale (Maciaszek, 2002). Un întreg ciclu de dezvoltare de tip *roundtrip engineering* ar putea fi conceput ca și generalizare a metodei aici propuse – totuși, în acest proiect ne limităm la obiectivele legate de demonstrator, luând în considerare doar avantajele fluxului unidirecțional de cunoștințe dinspre baza de modele din back-end către componenta front-end, folosind un mediu de reprezentare a cunoștințelor procesabile de către mașini, bazat pe RDF (W3C, 2018).

Etapa 2017 a vizat implementarea unei prime iteraţii a demonstratorului, fiind prevăzute definirea scenariului de aplicare, proiectarea şi implementarea unei prime iteraţii a limbajului de modelare, a bazei de cunoştinţe şi a aplicaţiei front-end. De asemenea, s-a realizat diseminare continuă pe tot parcursul etapei.

2. INGREDIENTE ŞI TEHNOLOGII IMPLICATE

- **Limbajul de modelare şi implementarea agilă a acestuia într-un instrument software** se bazează pe o metodologie (AMME) aplicată cu ajutorul unei platforme de metamodelare (ADOxx[1]). Astfel de metodologii şi platforme au apărut pentru a oferi alternative viabile la utilizarea standardelor de modelare, pentru situaţiile în care există cerinţe de personalizare, cu specific de domeniu sau pentru crearea unui limbaj complet nou care nu este înrudit cu standardele existente. Alte exemple de astfel de metodologii şi platforme au fost prezentate în (Kelly et al., 2013) (Frank, 2013). În derularea proiectului am optat pentru AMME şi ADOxx, acestea fiind ghidate de conceptul de "metodă de modelare" structurată conform (Karagiannis and Kuhn, 2002) în notaţie, sintaxă, semantică, procedură şi funcţionalităţi de modelare;
- **Structura conceptuală a limbajului de modelare** proiectat are ca punct de pornire frameworkul Zachman (Zachman, 1987) precum şi studiul mai multor limbaje de modelare a întreprinderii care sugerează câteva principii cheie: cel al descompunerii metamodelului în mai multe perspective şi cel al abstractizării prin selectarea proprietăţilor relevante ce se vor interoga pe parcursul utilizării limbajului. Un exemplu de limbaj consacrat ce respectă aceste principii este Archimate (The Open Group, 2017), vezi Fig. 1 care sugerează şi o corespondenţă cu faţetele Zachman. Framework-ul respectiv a introdus o structură ontologică pentru sistemele informaționale ale întreprinderilor, fără a oferi însă și mijloacele de integrare semantică cu metodologii de dezvoltare software. Abia prin evoluția tehnologiilor semantice, mai exact a frameworkurilor de reprezentare formală a cunoștințelor (RDF, limbaje agile de modelare), acest lucru devine posibil, iar îmbinarea acestor ingrediente stă la baza proiectului de față;

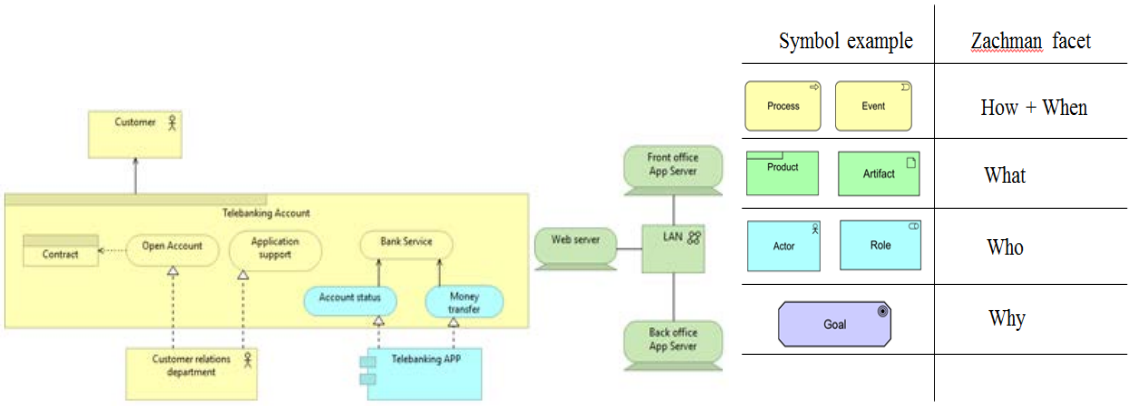

Fig. 1. Diagrame ArchiMate și fațete Zachman acoperite

- Pentru **stocarea bazei de cunoştinţe hibride** s-a apelat la serverul GraphDB[2] care foloseşte modelul de date bazat pe grafuri RDF[3] şi oferă un serviciu de interogare la distanţă prin HTTP, de tip REST. Un recent raport Bloor (Howard, 2016) arată că bazele de grafuri reprezintă segmentul

[1] https://www.adoxx.org/
[2] https://ontotext.com/products/graphdb/
[3] https://www.w3.org/RDF/

cu cea mai rapidă creștere de pe piața bazelor de date, datorită suportului lor pentru: (i) manipularea facilă a relațiilor m:n între entități; (ii) reprezentarea de semantică procesabil de către calculator, făcând posibilă raționarea pe baza unei algebre a grafurilor; (iii) aplicarea de metode analitice pe grafuri de cunoştinţe. GraphDB care a fost recent adoptat pentru platforma Springer Nature SciGraph (Springer, 2017). Proiectul de față reprofilează avantajele bazelor de grafuri în scopul gestionării informației derivate din diagrame şi integrării semantice a acestora cu alte informaţii relevante. Relațiile m:n și relațiile n-are sunt comune in astfel de reprezentări grafice. Interpretarea conectorilor vizuali ca "relații" implică faptul că limbajele de modelare sunt guvernate de o conceptualizare bine definită (metamodel). Aşadar folosirea unui program de modelare nu se limitează la documentarea grafică; poate urmări și reprezentarea diagramatică de cunoștințe în sprijinul unor procese. În evoluția modelării diagramatice, ne aflăm într-un punct în care obiectivele inițiale de a facilita comunicarea de la om la om sunt complementate atât de o utilizabilitate bogată (notație dinamică și interactivă) dar și de interoperabilitatea semantică (focusul proiectului de față). Pe lângă diagrame și axiome, un al treilea factor care poate lua parte la hibridizarea discutată – date relevante pentru execuție. Versiunile recente ale GraphDB includ plug-in-ul OntoRefine (OntoRefine, 2017), care poate importa date la nivel de instanță din surse non-graf, offline sau online (ex., Excel, CSV, JSON). Cei trei factori pot forma o bază de cunoștințe hibridă care este consolidată și uniformizată sub aspectul reprezentării de către Resource Description Framework.

4. ARHITECTURA

Arhitectura de nivel înalt este prezentată în Fig. 2. Conținutul modelelor împreună cu informațiile din metamodel sunt convertite în grafuri RDF și conectate prin tehnici Linked Data (Heath și Bizer, 2011) la alte entități care nu se regăsesc în modele. Este nevoie de o bază de grafuri cu capabilități inferențiale precum GraphDB.

S-a folosit un adaptor pentru ADOxx pentru a serializa modelele (îmbogățite și cu informații din metamodel) pentru orice limbaj de modelare implementat pe această platformă în concordanță cu un meta-metamodel abstract (ușor de transpus la nevoie și în altă platformă de metamodelare). Rezultatul este o bază de grafuri RDF – câte un graf pentru fiecare diagramă precum și pentru schema RDF derivată din metamodel și metadatele asociate cu fiecare diagramă. Această bază de modele poate fi în continuare îmbunătățită prin aplicarea de reguli pentru a produce informaţii necesare componentei front-end.

Artefactele software realizate pentru front-end sunt parametrizate semantic cu informații care pot fi extrase prin intermediul interogărilor SPARQL sau prin alte mijloace bazate pe servicii Web (Cinpoeru, 2017). Cerințele pentru modificări în funcționalitatea dezvoltată se pot propaga înapoi către limbajul de modelare, declanșând noi iterații AMME și crearea unui nou prototip al instrumentului de modelare, apoi crearea unui nou set de modele, eventual extinzându-le pe cele din iterația precedente.

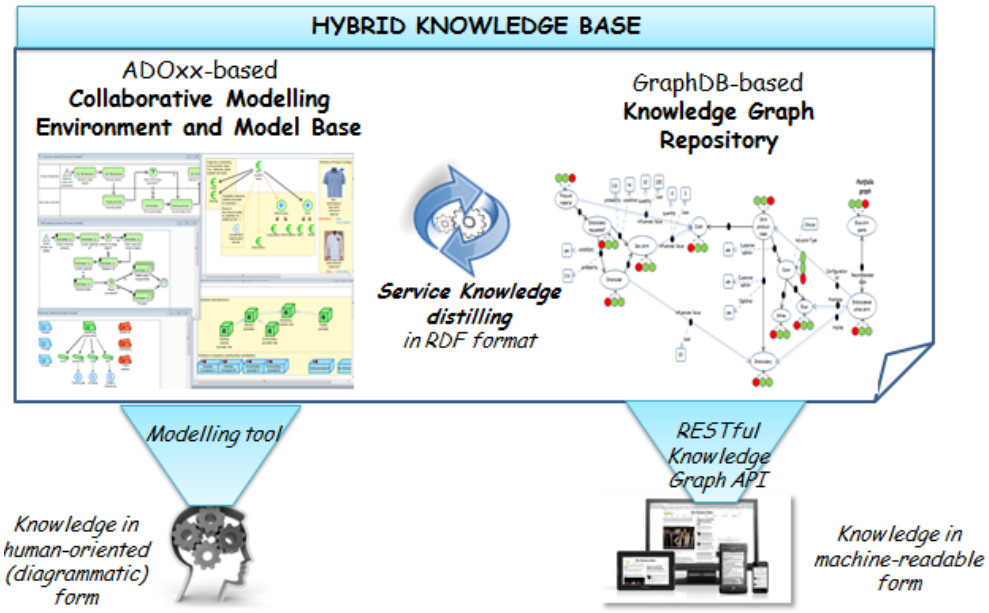

Fig. 2. Arhitectura demonstratorului (Buchmann și Ghiran, 2017)

3. METODA DE INGINERIE SOFTWARE PROPUSĂ

Lucrarea de referință (Karagiannis, 2015) prezintă pe larg principiile ce stau la baza AMME. Artefactul dezvoltat prin AMME este o *metodă de modelare* iar rezultatul pragmatic şi utilizabil este un instrument de modelare care expune un limbaj de modelare către utilizator, atât pentru activitățile de modelare de bază cât și pentru funcţionalităţi bazate pe modele (generare de cod, transformarea modelelor etc.).

Un astfel de instrument de modelare evoluează printr-o dezvoltare incrementală iterativă. La fel cum în dezvoltarea agilă de software cerințele sunt considerate instabile sau evolutive, tot așa și cerințele de modelare sunt considerate dinamice în AMME.

Cercetarea noastră demonstrează fezabilitatea unei metodei inovative de inginerie software bazată pe modele care inversează *subordonarea tradițională dintre implementare model-driven și modele* –în loc să avem implementarea subordonată unui limbaj de modelare invariant (limitat de un spațiu semantic rigid), propunerea este de a avea limbajul de modelare subordonat unor nevoi de implementare în continuă dezvoltare (iterativă). Astfel, limbajul de modelare și instrumentul de modelare corespunzător sunt adaptate în mod agil pentru a oferi semantica solicitată implementării.

Metoda propusă extinde așadar AMME pentru aplicabilitate generalizabilă în domeniul ingineriei software. Extensiile aduse sunt sintetizate într-o perspectivă metodologică de ansamblu prin Fig. 3.

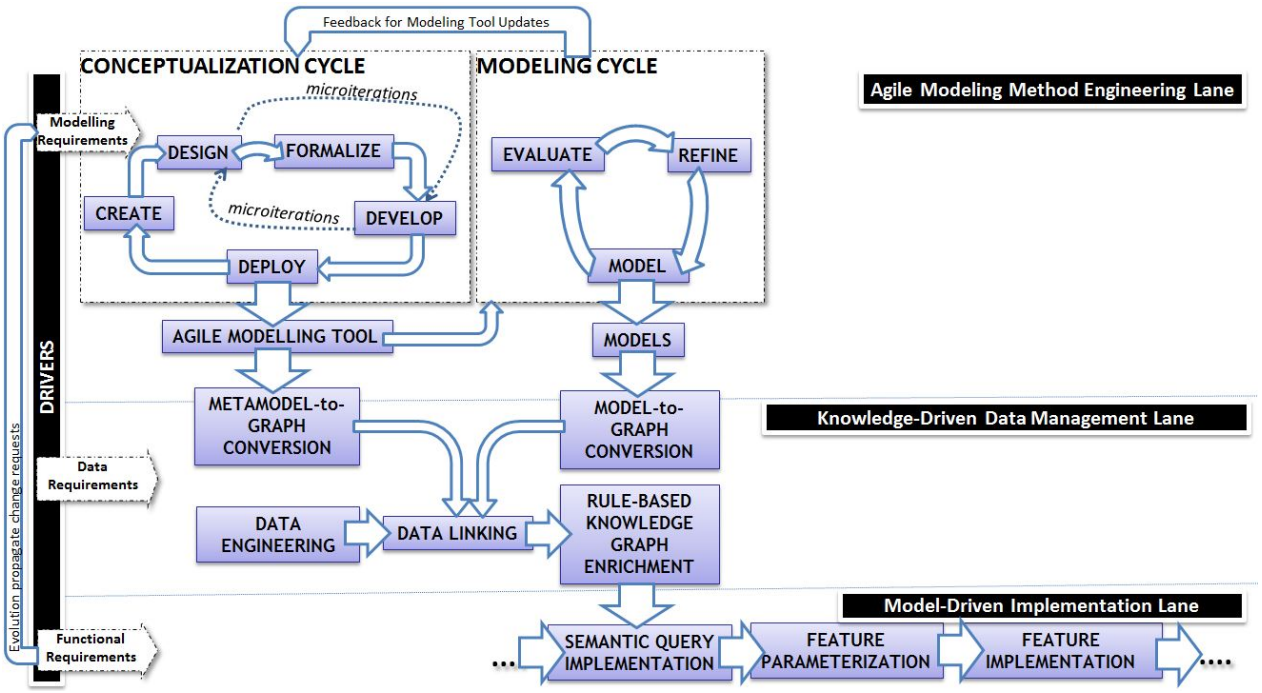

Fig. 3. Fluxul conversiei cunoștințelor în ingineria software bazată pe modele (Buchmann et al., 2018a)

Având în vedere că AMME permite crearea unui limbaj de modelare evolutiv (și în consecință a unui spațiu semantic evolutiv), nu mai este fezabilă dezvoltarea sincronizată a generatoarelor de cod – cerințele de modificare pot fi atât de drastice încât să presupună adăugarea unui tip complet nou de diagramă în limbaj, limitând refolosirea însă încurajând cuplarea semantică dintre modele și artefactul software din front-end. Pornind de la aceste premise, limbajele de modelare dezvoltate prin AMME nu servesc direct proiectării sistemelor (pentru care standardele existente sunt deja consacrate) ci mai degrabă pentru achiziția flexibilă de cunoștințe și integrarea lor într-un ciclu de conversie a cunoștințelor definit şi generalizat în (Karagiannis et al., 2017). Fiind în esență un framework pentru metamodelare, AMME permite captarea cunoștințelor pe două niveluri:

- *cunoștințele domeniului* – exprimate prin conceptele şi proprietăţile asimilate la fiecare iteraţie a limbajului, în cadrul unui ciclu de *Conceptualizare*;
- *cunoștințele specifice unui caz* – exprimate prin modele, prin folosirea unei iterații a limbajului ce a fost proiectat la nivelul anterior, în cadrului unui ciclu de *Modelare*.

Mai departe, RDF și metodele de interogare a grafurilor aferente (W3C, 2018) se folosesc pentru a facilita fluxul conversiei de cunoștințe spre fazele de dezvoltare software ghidată de modele.

5. Scenariu și exemplu ilustrativ pentru prima iterație a implementării

Un tip de sistem IT comun în întreprinderi sunt sistemele de gestiune a execuției fluxurilor de lucru (Workflow Management System) care pot fi cuplate cu descrieri diagramatice ale proceselor. Scenariul adoptat pentru exemplificare este o întreprindere virtuală ce oferă haine particularizate la a căror producție contribuie o rețea de colaborare ce include croitori, curieri și alte roluri, implicate în funcție de disponibilitate și capabilități în execuția procesului de producție la comandă și de livrare.

Fig. 4 exemplifică un proces de producție simplificat creare-la-comandă împreună cu o captură de ecran a componentei front-end de gestiune și atribuire de sarcini de lucru, arătând sarcinile atribuite unui user autentificat – active, îndeplinite și în așteptare (în ultimul caz, arată și datele de contact pentru persoanele responsabile).

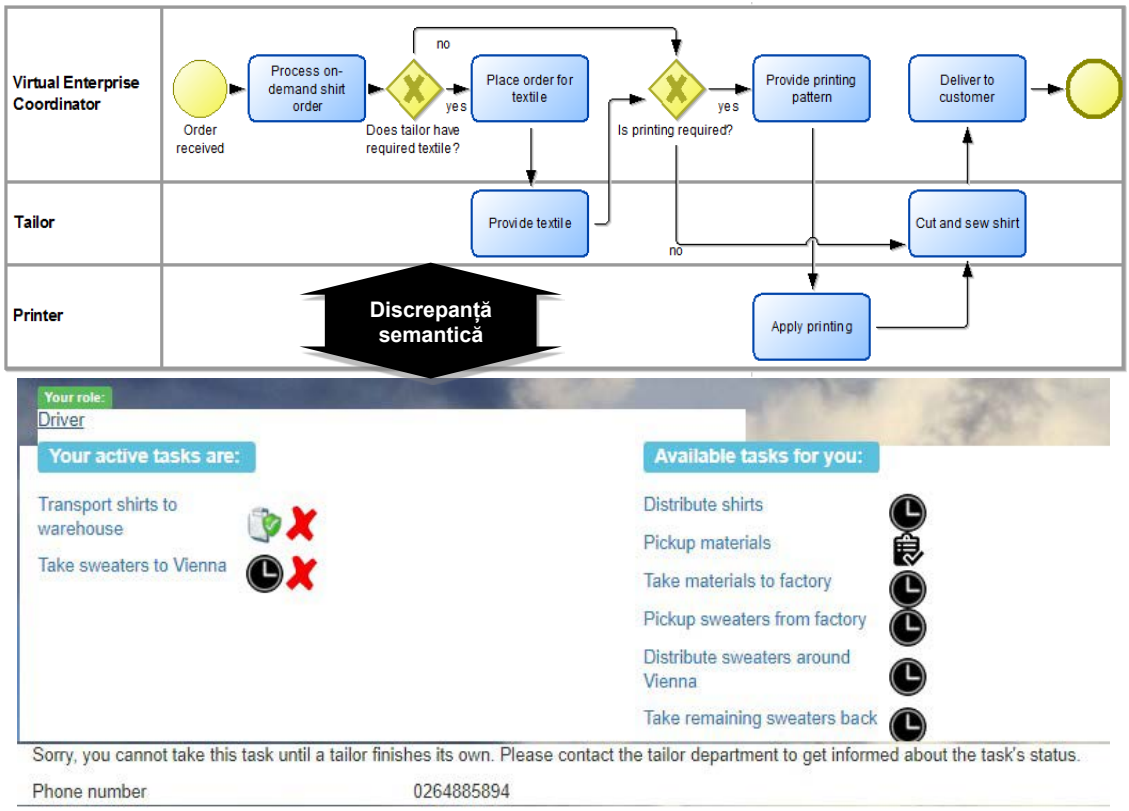

Fig. 4. Exemplu de proces modelat prin limbaje standard (BPMN) și componentă front-end

Între funcționalitatea front-end și informațiile din model există o discrepanță semantică evidentă care în general este rezolvată în sistemele de gestiune a execuției fluxurilor de lucru prin modelul de date întrebuințat la momentul execuției, folosind descrierea modelului ca și ghid pentru executarea sarcinilor, extins după necesități la nivelul implementării front-end.

Metoda propusă de proiectul de față recomandă o particularizare agilă a limbajului de modelare pentru a asimila cunoștințe mai bogate direct în mediul de modelare (chiar și funcționalitate de tipul atribuirii de sarcini spre responsabilii acestora). Nu este neobișnuit să întâlnim anumite redundanțe conceptuale între modelul de date folosit în timpul execuției și metamodelul folosit în instrumente de modelare – de aceea proiectul de față recomandă minimizarea acestei redundanţe prin transferul unor fragmente conceptuale dinspre modelul de date necesar execuţiei către mediul de modelare.

Față de modelul exemplificat în Fig. 4, următoarele cerințe de modelare sunt asumate pentru demonstratorul dezvoltat în proiect:

- *Cerințe semantice*: Ecosistemul întreprinderii virtuale ar trebui să fie descris cu mai multe detalii decât se permite în BPMN prin încadrarea în culoare de activități (*pools, swimlanes*). Anumite concepte și relații dedicate ar trebui să fie formulate pentru a captura structurile organizaționale, rolurile, angajații precum și partenerii de afaceri grupați după capabilitățile pe care le oferă întreprinderii virtuale. În mod similar, un model pentru acoperirea geografică ar trebui să ofere cel puțin o grupare a locațiilor vizate. Pentru elementele ce reprezintă instanțe sunt necesare proprietăți interogabile (de ex. adresa, coordonatele locației, date de contact pentru partenerii de afaceri sau pentru angajați). De asemenea, trebuie să se asigure o distincție între tipuri de sarcini specifice domeniului (sarcini de producție, sarcini de livrare, sarcini de transport);
- *Cerințe sintactice*: conceptele noi trebuie să fie separate de descrierea procesului (sub forma unor tipuri distincte de modele care să poată mai departe evolua independent). E necesar un model al locațiilor și un model al participanților la afacere, cu posibilitatea de a le pune în corespondență cu sarcinile proceselor prin intermediul hiperlegăturilor dintre modele;
- *Cerințe ale notației*: În locul conectorilor vizuali se preferă gruparea simplă sub forma containerelor grafice. Distincția dintre diverse tipuri de sarcini trebuie să fie indicată în mod vizual. În locul notației standard, aceasta ar trebui să fie îmbunătățită cu indicații vizuale specifice domeniului, indicii vizuale care să poată servi și ca hiperlegături dintre modele.

Anumite cerințe ar putea fi satisfăcute și de limbaje standard – de ex. împărțirea rolului Croitor în două (unul pentru croitor în sensul propriu și unul pentru furnizorul textil) ori distincții (limitate) între tipuri diferite de sarcini cu specific de domeniu. Totuși anumite descrieri și proprietăți implicate de cerințele menționate anterior depășesc aria de acoperire semantică a limbajelor de modelare standard pentru descrierea proceselor de afaceri – de ex. structuri organizaționale, corespondențele între roluri și instanțe (necesare în cadrul configurărilor agile dintr-o întreprindere virtuală), acoperirea geografică a execuției proceselor sau atribuirile de sarcini (în general implementate în funcționalitatea front-end). Se poate argumenta că o parte din aceste informații ar aparține de domeniul instanțelor și datelor necesare la momentul execuției – însă trebuie să se ia în considerare următoarele aspecte: (i) unele date sunt suficient de stabile pentru a fi păstrate în modele; (ii) anumite reprezentări ale instanțelor sunt de interes pentru modelare în general (a se vedea paradigma modelării multi-nivel (Clark et al., 2014)) sau pentru anumite obiective ale analizei modelelor (de ex. simulări ale volumului de lucru pentru anumite

instanţe). Exemple tipice de modele ce includ descrieri de instanțe sunt hărțile de companie/procese, modelele de cerințe, descrieri ale mediului de lucru As-Is, modele de lanțuri de distribuție care iau în considerare acoperirea geografică sau parteneri de afaceri concreți, arhitecturi software As-Is. Proprietățile acestora pot fi la fel de stabile precum modelele de procese și de aceea se pot cupla semantic cu acestea pentru a îmbogăți contextul modelelor – o cerință uzuală în managementul arhitecturilor de întreprindere (Moser et al., 2017).

Cu alte cuvinte, acele părți ale unei baze de date relaționale care sunt prin natura lor mai degrabă invariante și ar putea servi unor scopuri de modelare (de ex. interogarea modelelor) pot fi transferate către o bază de modele. Legăturile dintre reprezentările acestora și proprietățile dinamice ale acestora (de ex. disponibilitatea angajaților în timp real) vor fi gestionate cu ajutorul mecanismelor de relaționare inerente oferite de bazele de grafuri.

Fig. 5 arată modul în care procesul BPMN de la care am pornit (Fig. 4) a fost elevat la nivelul unui limbaj de modelare particularizat, cu ajutorul a trei tipuri de diagrame – de proces, a locațiilor și a participanților. Hiperlegăturile dintre modele stabilesc relații semantice între acestea (de ex. locațiile participanților atribuiri de sarcini la nivel de rol sau la nivel al instanțelor). Relația conceptuală *hasResponsible – areResponsabil* (între sarcini și cei care sunt responsabili pentru realizarea acestora) la nivel de modelare, este specializată în două hiperlegături – *assignedInstance InstanțaAtribuită* (dacă responsabilul este o entitate instanță) și *requiredCapability - CapabilitateSolicitată* (dacă responsabilitatea este alocată unui rol). Agilitatea se manifestă în toate elementele constitutive ale metodei de modelare – *notație* (de ex. înlocuirea simbolurilor BPMN cu indicații vizuale specifice domeniului) *sintaxă* (de ex. împărțirea metamodelului în mai multe tipuri de modele conectate prin hiperlegături), *semantică* (de ex. adăugarea unor concepte noi, concepte specializate, adăugarea unor proprietăți specifice domeniului).

În continuarea procesului de dezvoltare, modelele astfel particularizate sunt exportate într-o bază de cunoștințe RDF, deschizându-le spre extensii relevante (relaționarea cu alte date sau inferențe). Fig. 6 izolează un fragment care conține elemente ale modelelor din toate cele trei tipuri de modele reprezentate în Fig. 5, inclusiv hiperlegăturile dintre acestea.

Graful derivat conține toate relațiile exprimate vizual în modele (atât prin conectori cât și containere), extinse cu:

- Legături către date dinamice (de ex. disponibilitatea resurselor umane) care nu ar avea sens să fie incluse în modele (datorită naturii schimbătoare a acestora). În acest exemplu este vorba de disponibilitatea în timp real a resurselor (precum resurse umane, locuri de parcare). Astfel de legături sunt stabilite pe baza identificatorilor URI.

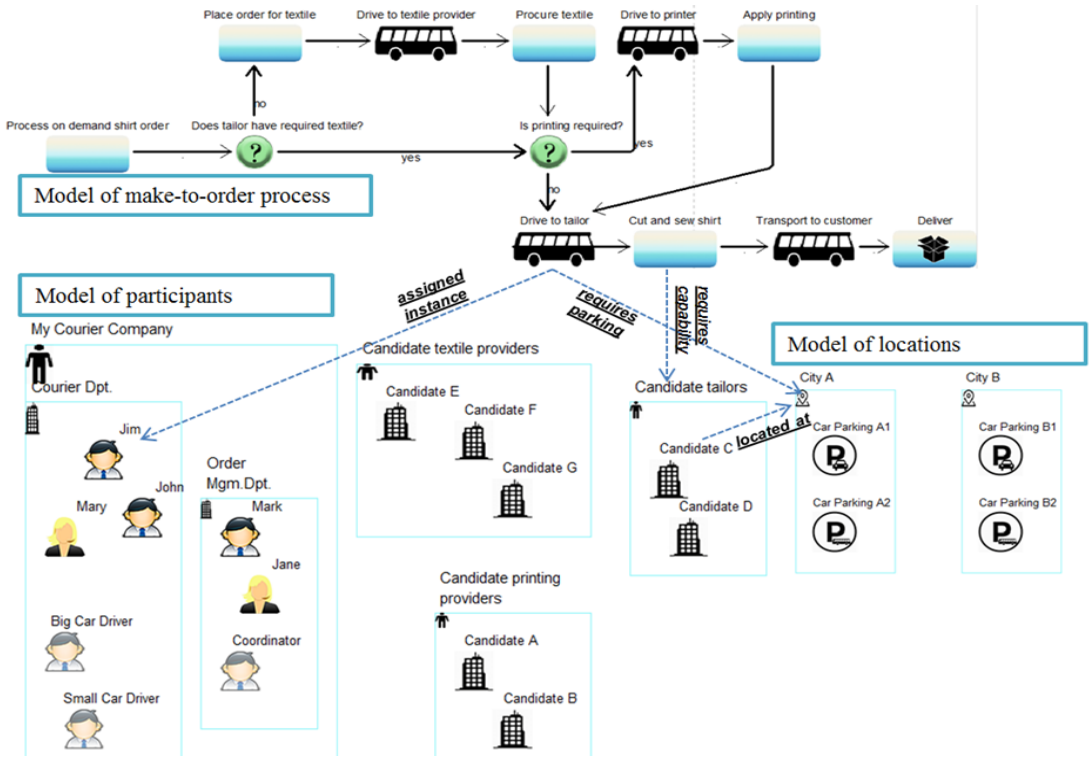

Fig. 5. Iterația curentă a instrumentului de modelare agilă: exemple de modele și hyperlinkuri între acestea

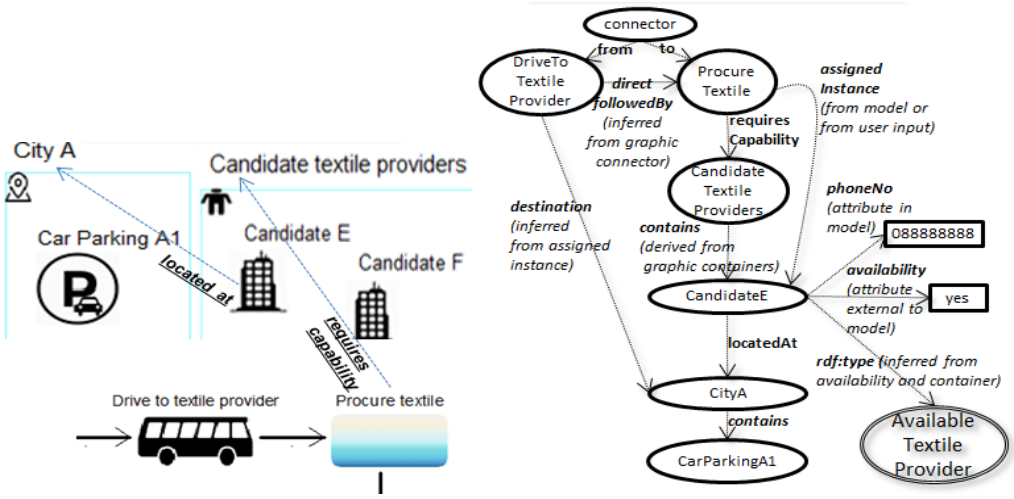

Fig. 6. Fragment izolat de modele și linkuri (stânga); structura de graf derivată din acestea (inclusiv relații deduse (dreapta)

- Relații directe deduse (de ex. *directFollowedBy – urmatDirect*) corespunzătoare conectorilor grafici disponibili în modele. Acestea sunt necesare deoarece conectorii vizuali nu pot deveni arcuri directe în graful exportat, datorită posibilității de a avea și alte proprietăți care sunt atribuite acestora (din acest motiv conectorii devin relații de ordin n). Axiomele OWL pot fi folosite în acest sens:

```
:directFollowedBy owl:propertyChainAxiom([owl:inverseOf :from] :to).
```

- Tipuri deduse din restricții ale proprietăților (de ex. *AvailableTextileProvider FurnizorTextilDisponibil* pentru acele elemente la care proprietatea *availability – disponibiltate* este adevărată și care sunt conținute în caseta *CandidateTextileProdiver - PosibilFurnizorTextil* pentru *required capability.* Exemple de axiome OWL serializată în RDF:

```
:containedBy owl:inverseOf :contains.

:TextileProvider owl:onProperty :containedBy; owl:hasValue :CandidateTextileProvider.

:AvailableInstance owl:onProperty :availability; owl:hasValue true .

:AvailableTextileProvider owl:intersectionOf (:AvailableInstance :TextileProvider).
```

- Anumite relații (de ex. *instanță atribuită*) pot fi stabilite fie în instrumentul de modelare (de ex. coordonatorul va atribui direct instanțe către anumite sarcini prin intermediul mijloacelor de modelare) sau în interfața front-end (de ex. un utilizator va prelua responsabilitatea pentru anumite sarcini puse în containerul său de capabilități);
- Relații deduse bazate pe înlănțuirea proprietăților relevante (de ex. *destination – destinația* este identificată prin combinarea separării vizuale cu disponibilitatea instanței). Acest lucru poate fi obținut printr-o succesiune de axiome sau reguli Horst (suportate de motorul de raționare GraphDB). Un exemplu de axiomă OWL serializată în RDF este:

```
:destination owl:propertyChainAxiom (:directFollowedBy :assignedInstance :locatedAt).
```

Graful rezultat este expus interfeței front-end din Fig. 7 – o interfață de utilizare pentru actori implicați în procesele descrise unde aceștia pot vedea și accepta sarcinile alocate lor, conform informațiilor interogate din acest graf. Graful este interogat prin interfața REST oferită de serverul GraphDB și prin intermediul unui serviciu server-side care pregătește informația pentru afișare. În iterația curentă s-a apelat la tehnica dereferențierii (Cinpoeru, 2017), prin care un identificator (URI) al unui element din model e tratat ca URL ce oferă un serviciu Web prin care proprietățile acelui element de model pot fi obținute și aduse în front-end într-o manieră procesabilă de către mașini, apoi interogată local și pregătită pentru afișare.

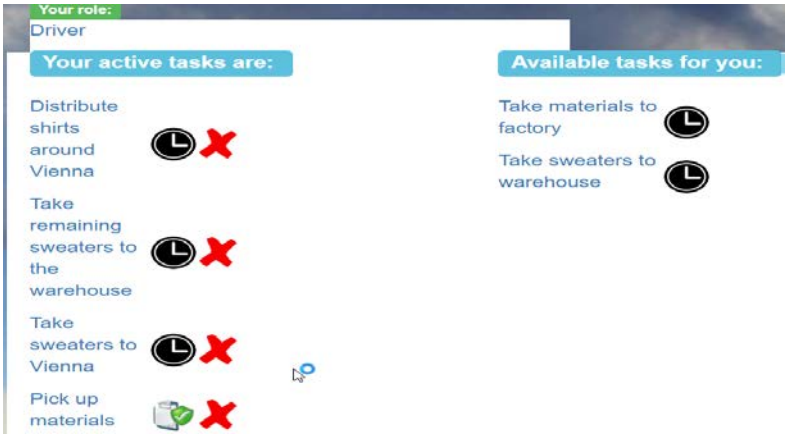

Fig. 7. Mostră din interfața front-end cu sarcinile unui angajat autentificat cu rolul de șofer

6. DISEMINARE

Lista lucrărilor afiliate proiectului publicate în Etapa 2017:

Articole în conferinţe clasa A şi workshopuri afiliate conferinţelor clasa A[4]:

1. Karagiannis D., Buchmann RA., Walch M. (2017) "How Can Diagramatic Conceptual Modelling Support Knowledge Management?". In Proceedings of the 25th European Conference on Information Systems (ECIS), Guimarães, Portugal, June 5-10, 2017, (pp. 1568-1583). ISBN 978-989-20-7655-3, Association for Information Systems.
2. Moser C., Buchmann RA., Utz W., Karagiannis D., (2017) "CE-SIB: A Modelling Method Plug-in for Managing Standards in Enterprise Architectures", In Proceedings of the 36th Int. Conf. on Conceptual Modelling (ER 2017), November 6-9, 2017, Valencia, Spain, LNCS 10650, (pp. 21-35), ISBN 978-3-319-69904-2, Springer.

Articole în conferinţe clasa B şi workshopuri afiliate conferinţelor clasa B:

3. Cinpoeru M., (2017) "Dereferencing Service for Navigating Enterprise Knowledge Structures from Diagrammatic Representations", In Proceedings of the BIS 2017 workshops, 2017, Poznan, Poland, June 28-30, 2017, LNBIP 303, (pp. 85-96), ISBN 978-3-319-69023-0, Springer.
4. Ghiran AM., Osman CC., Buchmann RA. (2017) "A Semantic Approach to Knowledge-Driven Geographical Information Systems" In Proceedings of the 18th European Conference on Knowledge Management (ECKM), Barcelona, Spain, September 7-8, 2017, (pp. 353-362).ISBN 978-1-911218-49-4, ACPI
5. Ghiran AM.(2017) "Meme Lifecycle Semantics and Organizational Knowledge Creation" In Proceedings of the 18th European Conference on Knowledge Management (ECKM), Barcelona, Spain, September 7-8, 2017, (pp. 363-372). ISBN 978-1-911218-49-4, ACPI
6. Mocean L., Buchmann RA.(2017) "Open Knowledge-Aware Academic Management Systems" In Proceedings of the 18th European Conference on Knowledge Management (ECKM), Barcelona, Spain, September 7-8, 2017, (pp. 714-722). ISBN 978-1-911218-49-4, ACPI

Articole în alte conferinţe şi workshopuri internaţionale:

7. Buchmann RA., Ghiran AM. (2017) "Serviceology-as-a-Service: a Knowledge-Centric Interpretation", In: Proceedings of the 5th International Conference on Serviceology (ICServ 2017), June 28-30, 2017, Vienna, Austria, LNCS 10371 (pp. 190-201), ISBN 978-3-319-61240-9, Springer.
8. Ghiran AM., Buchmann RA., Osman CC., Karagiannis D. (2017) "Streamlining Structured Data Markup and Agile Modelling Methods", In Proceedings of the 10th IFIP WG 8.1 working conference on the Practice of Enterprise Modelling (POEM 2017), November 22-24, 2017, Leuven, Belgium,LNBIP 305 (pp. 331-340), ISBN 978-3-319-70241-4, Springer.
9. Buchmann RA., Ghiran AM., (2017) "Engineering the Cooking Recipe Modelling Method: a Teaching Experience Report", In Proceedings of the 1st International Workshop on Practicing Open Enterprise Modeling within OMiLAB (PrOse2017), November 22-24, 2017, Leuven, Belgium, CEUR-WS 1999, paper 1.
10. Harkai A.(2017) "A Comparative Survey of Prominent Enterprise Modeling Methods". In Proceedings of the 16th International Conference on Informatics in Economy (IE), Bucharest, Romania, May 4-7, 2017, (pp. 194-199). ISSN 2284-7472, ASE.

[4] conform CORE ranking 2018 (http://portal.core.edu.au/conf-ranks/), clasificare de referinţă pentru conferinţele de informatică din întreaga lume

11. Cinpoeru M.(2017) "Design and Implementation of a Dereferencing Service for Enterprise Resource Identifiers". In Proceedings of the 16th International Conference on Informatics in Economy (IE), Bucharest, Romania, May 4-7, 2017, (pp. 501-506). ISSN 2284-7472, ASE.

BIBLIOGRAFIE

Buchmann R A, Ghiran A M, 2017. *Serviceology-as-a-Service: a knowledge centric interpretation*. Proceedings of ICServ 2017, LNCS 10371, pp. 190-201, Springer

Buchmann RA., Cinpoeru M., Harkai A., Karagiannis, D. (2018a). Model-Aware Software Engineering - A Knowledge-based Approach to Model-Driven Software Engineering, In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018*), Funchal, Madeira, Portugal, (pp. 233-240), SciTe Press.

Cinpoeru M, 2017, Dereferencing service for navigating enterprise knowledge structures from diagrammatic representations. Proceedings of AKTB 2017, LNBIP 303, pp. 85-96, Springer

Clark T, Gonzalez-Perez C, Henderson-Sellers B, 2014. *A foundation for multi-level modelling*. Proceedings of Multi-level Modelling Workshop at MoDELS 2014, CEUR-WS 1286, pp. 43-52.

Frank U, 2013. *Domain-specific modeling languages: requirements analysis and design guidelines*. In *Domain Engineering*, pp. 133–157, Springer.

Heath T, Bizer C, 2011. *Linked Data: Evolving the Web into a Global Data Space* (1st edition). Morgan & Claypool

Howard P, 2016. *Graph Databases*, http://www.bloorresearch.com/technology/graph-databases, accessed Nov. 2017

Karagiannis D, 2015. *Agile modeling method engineering*. Proceedings of the 19th Panhellenic Conf. on Informatics, pp. 5-10, ACM.

Karagiannis D, Buchmann RA, Walch M, 2017. *How can diagrammatic conceptual modelling support Knowledge Management?* Proceedings of ECIS 2017, paper 101, Association for Information Systems

Karagiannis D, Kühn H, 2002. *Metamodelling Platforms*. Proceedings of the Third International Conference EC-Web 2002 – DEXA 2002, LNCS 2455, pp. 182, Springer.

Karagiannis, D, Mayr H C, Mylopoulos J, 2016. *Domain-Specific Conceptual Modeling*, Springer.

Kelly S, Lyytinen K, Rossi M, 2013. *MetaEdit+ A fully configurable multi-user and multi-tool CASE and CAME environment*. In *Seminal Contributions to Information Systems Engineering*, pp. 109-129, Springer.

Maciaszek L A, 2002. *Process Model for Round-trip Engineering with Relational Database.* In *Successful Software Reengineering*, pp.76-91, IGI Global

Moser C, Buchmann R A, Utz W, Karagiannis D, 2017. *CE-SIB: a modelling method plug-in for managing standards in enterprise architectures*, Proceedings of ER 2017, LNCS 10650, pp. 21-35, Springer

OpenRefine Community, 2017. *OpenRefine* – the official project page, http://openrefine.org, accessed Nov. 2017

Springer, 2017. *Springer Nature SciGraph* – official website, http://www.springernature.com/gp/researchers/scigraph, accessed Nov. 2017

The Open Group, 2017. *Archimate* – official page, http://pubs.opengroup.org/architecture/archimate3-doc/toc.html, accessed Nov. 2017

W3C, 2018. *RDF 1.1 Concepts and Abstract Syntax*, http://www.w3.org/TR/rdf11-concepts, accessed Nov. 2017

Zachman J A, 1987, *A framework for information systems architecture*. IBM Systems Journal 26 (3): 276-292.