ULBS
Universitatea "Lucian Blaga" din Sibiu

Interdisciplinary Doctoral School

PhD field of study: Computers and Information Technology

PhD Thesis

Categorical Mechanisms in Multi-level Modeling Methods

-REZUMAT-

PhD Student:

Daniel-Cristian Crăciunean

PhD Supervisor:

o. Univ.-Prof. Prof.h.c. Dr. Dimitris Karagiannis

SIBIU 2021

CONTENTS

1 Introducere

Principala cale de a stăpânii complexitatea și heterogenitatea sistemelor implicate în toate domeniile de activitate din zilele noastre este modelarea. Metodele conceptuale de modelare, care presupun reprezentarea entităților statice și dinamice ale sistemelor, sprijină comunicarea între utilizatori și dezvoltatori și facilitează stăpânirea eficientă a complexității și heterogenității acestor sisteme. Astfel, concepte ca internet of things, digital twins, smart factories, smart products, cyber-physical systems, devin concepte tot mai frecvente în procesele de modelare. De exemplu, specificarea unui proces flexibil de fabricație implică o varietate de sisteme cum ar fi roboți, manipulatoare, componente software, elemente de control etc. Rezultatul agregării acestor componente este un Cyber-Physical Production System(CPPS) [Muller2018].

Prin modelarea conceptuală a unui sistem înțelegem procesul de gândire și raționament asupra unui sistem în scopul construirii unui model, precum și procesul de specificare a modelului rezultat. Modelarea este o metodă de cunoaștere și de reflectare asupra unui sistem cu scopul de a-l înlocui cu un model mai accesibil studiului. Modelul rezultat trebuie să ofere mecanisme de raționament, verificare, validare, simulare și generare de cod executabil [Fill2013].

Construirea unui model presupune și specificarea lui într-un formalism pentru a putea fi utilizat în comunicarea cu alții sau pentru a efectua experimente. Construirea unui model începe de obicei cu un nivel informal, util pentru discuții și confruntarea de idei, trece apoi printr-o fază semi-formală și ajunge la un nivel formal, adică executabil. În general formalismul de specificare al unui model influențează modul în care este perceput un sistem și prin urmare modul în care este modelat.

Este validată ideea că limbajele diagramatice posedă caracteristici intuitive importante și prin urmare sunt cele mai potrivite pentru modelarea fenomenelor și a relațiilor dintre ele [Wolter2015]. Deși există o mulțime de astfel de limbaje, diversitatea sistemelor care trebuie modelate în zilele noastre face ca acestea să fie, de multe ori, prea generale pentru specificarea modelelor din anumite domenii. Prin urmare implementarea limbajelor de modelare specifice domeniului (DSML) [Fowler2010] a devenit o parte integrantă a procesului de modelare.

Un limbaj specific domeniului (DSL) este un limbaj cu sintaxă și semantică bazate pe abstracții aliniate la un anumit domeniu al cunoașterii cu scopul de a permite specificarea optimă a problemelor din domeniul respectiv.

In contextul Model-Driven Development (MDD), DSL-rile utilizate pentru specificarea modelelor se numesc limbaje de modelare specifice domeniului (DSML) și joacă un rol fundamental. Modelele specificate în felul acesta sunt artefacte primare pe baza cărora se pot genera modele executabile sau interpretabile.

MDD are ca obiectiv principal mutarea efortului de dezvoltare software de la programare la modelare. Aceasta presupune, printre altele, generarea automata de cod executabil pe baza informațiilor extrase din modele [Rumpe2017]. Dar codul executabil este o specificație precisă, completă și fără ambiguități, adică o specificație formală. Prin urmare acest deziderat impune cerințe mai mari pentru exactitatea specificațiilor modelelor, deoarece o eroare în proiectarea unui model ar poate fi

descoperită numai după ce produce deja pierderi cumulate mari. Acest lucru duce inevitabil la necesitatea construirii de modele formale care pot fi verificate teoretic.

Pentru a specifica formal un model avem nevoie de un limbaj formal, adică de un limbaj înzestrat cu sintaxă și semantică precise și lipsite de ambiguitate. Nu este suficient ca limbajul de specificare să fie formal pentru ca modelele specificate cu acel limbaj să fie la rândul lor formale. Un limbaj riguros poate fi folosit pentru a face specificații imprecise, dar pentru specificarea formală a modelelor o condiție esențială este ca limbajul să fie formal [Rumpe2016].

Conceptele fundamentale pe care se concentrează această lucrare sunt: modelarea multi-nivel, modeling method, și limbajul destinat specificării conceptului de modeling method: MM-DSL. La acestea se adaugă câteva mecanisme din teoria categoriilor care servesc la formalizarea conceptelor de modelare multi-nivel și modeling method. Această lucrare contribuie printre altele și la compatibilizarea mecanismelor din teoria categoriilor cu conceptele implicate în activitatea de modelare.

Utilizarea mai multor niveluri de modele în sensul de specificare a modelelor în termenii altor modele are o istorie îndelungată. Modelarea pe mai multe niveluri pare a fi cea mai bună metodă de abordare a reducerii complexității modelelor prin abstractizare și distribuirea acestei complexități pe mai multe nivele. Această modalitate de modelare este veche dar a început să fie foarte importantă o dată cu creșterea complexității sistemelor modelate [Atkinson2001] [Atkinson2008] [Lara2003] [Lara2010].

O abordare de succes este arhitectura introdusă de OMG care clasifică modelele în patru straturi notate cu M0, M1, M2 și M3 [OMG_MDA2014]. O altă abordare importantă a modelarii pe mai multe nivele este cea bazată pe conceptul de clabject [Atkinson1997], implementată într-un instrument numit Melanee [Atkinson2016].

În general pentru dezvoltarea unui domeniu este nevoie de definirea unor concepte adecvate care să concentreze efortul unui număr important de cercetători care să contribuie la perfecționarea respectivului concept. Un astfel de concept este cel de modeling method introdus de Karagiannis și Kühn [Karagiannis2002].

Conceptul de modeling method integrează trei componente esențiale ale unui instrument de modelare și anume limbajul de modelare, algoritmii și mecanismele, și procedura de modelare. Cu aceste componente implementate, în conformitate cu procedura de modelare, orice model va putea fi specificat utilizând limbajul de modelare și va putea fi validat, analizat, executat sau simulat utilizând algoritmii și mecanismele. Conceptul de modeling method presupune modelarea multi-nivel, în special modelarea pe 4 nivele [OMG_MDA2014] , dar nu exclude modelarea pe mai multe nivele.

O condiție esențială pentru ca un limbaj să poată fi înzestrat cu o semantică formală este ca sintaxa sa să fie precisă și fără ambiguități. De cele mai multe ori limbajul asociat unei modeling method este un limbaj grafic. Dacă în cazul limbajelor textuale există un mecanism de specificare puternic și care oferă algoritmi eficienți de analiză și translatare, și anume gramatica generativă context-free, în cazul limbajelor grafice lucrurile sunt departe de a fi la fel. La prima vedere s-ar părea că nu există diferență între limbajele textuale și cele grafice decât de notare a atomilor lexical. Mai există însă un detaliu legat de relațiile între acești atomi în construcțiile sintactice. Dacă în cazul limbajelor textuale fiecare atom lexical poate avea cel mult doi atomi vecini, în cazul limbajelor grafice fiecare atom poate

interacționa cu un număr nelimitat de atomi. De aici rezultă și un beneficiu legat de expresivitatea sporită a acestor limbaje dar și o dificultate mare legată de analiza sintactică.

O direcție importantă și naturală de specificare a limbajelor grafice este cea bazată pe generalizarea mecanismelor utilizate în cazul limbajelor textuale la limbajele grafice. Astfel s-au înlocuit producțiile din gramatica Chomsky cu transformări graf. Ceea ce a rezultat este un concept la fel de elegant ca și în cazul limbajelor textuale, cu suport matematic dar încă nu s-au găsit algoritmi suficient de eficienți de analiză pentru a putea fi utilizată în practică în mod similar cu gramatica Chomsky. Din acest motiv sintaxa unui limbaj de modelare este, de obicei, specificată printr-un alt metamodel sau utilizând o notație matematică. Noi vom folosi în această lucrare schița categorică pentru a specifica sintaxa unui limbaj grafic. Vom folosi totuși transformările graf pentru a specifica sintaxa comportamentală a unui model. În acest caz nu avem nevoie de o analiză sintactică ci doar de o potrivire a regulilor comportamentale într-un model care poate fi făcută eficient prin legarea acestor transformări graf, la nivel de metamodel, de un anumit tip de elemente. Astfel, noi specificăm sintaxa comportamentală prin asocierea transformărilor graf cu semnături de acțiuni comportamentale.

Dacă sintaxa limbajului este formală atunci o mapare semantică la un domeniu cu semantica bine cunoscută atribuie un sens lipsit de ambiguitate fiecărei construcții sintactice specificate în limbajul de modelare. Atât dimensiunea statică cât și dimensiunea comportamentală a unui model trebuie înzestrate cu semantică. Astfel, semantica unui limbaj de modelare are două dimensiuni, o dimensiune statică și o dimensiune comportamentală.

Dimensiunea statică a semanticii are la rândul său doua componente și anume o componentă care înzestrează elementele limbajului cu tipuri și o componentă care mapează structurile modelului la structuri cu o semantică bine cunoscută.

Dimensiunea comportamentală a semanticii se definește de cele mai multe ori prin maparea semanticii de execuție specifică domeniului la semantica unor limbaje cu o semantica formală, ca de exemplu Petri Nets sau la anumite specificații algebrice. În această lucrare noi am specificat semantica comportamentala prin semantica transformărilor graf, care pot modifica structura graf a modelului, asociate cu acțiuni comportamentale care sunt funcții matematice și pot modifica atributele modelului. Aceste două mecanisme pot asigura trecerea modelului dintr-o stare în alta.

Notația se referă la elementele grafice ale limbajului și care face ca limbajele grafice să fie mai intuitive decât limbajele textuale. Prin notație figurile geometrice sau icons-urile joaca rolul atomilor lexicali din limbajele textuale. Deși, în general numărul atomilor lexicali grafici este mai mic decât numărul atomilor lexicali din limbajele textuale, relațiile mai complexe dintre aceștia oferă o expresivitate mai mare a limbajului.

Notația poate fi statică sau dinamică. Notația statică este o notație fixă pentru elementele limbajului în sensul că nu se schimbă în timpul execuției modelului. În acest caz trecerea modelului de la o stare la alta este reflectată doar prin modificarea valorilor atributelor. Notația statică este specifică celor mai multe limbaje de modelare. Notația dinamică presupune că limbajul de modelare oferă facilitatea de modificare dinamică a notației, în funcție de starea modelului sau valoarea atributelor unui model într-o anumită stare.

O metodă de modelare trebuie să ofere facilitați de analiză, verificare, validare, transformare, interogare și simulare a modelelor. În spatele acestor facilitați stau mecanisme și algoritmi care au la

bază proprietăți universale ale modelelor dintr-un anumit domeniu. În abordarea noastră, teoria categoriilor oferă o serie de construcții universale care pot constitui o sursă importantă de algoritmi generici.

Așa cum am menționat deja, dezvoltarea DSML-urilor devine o parte integrant[a procesului de modelare. Aceste DSML-uri sunt, în general, limbaje grafice adaptate pentru specificarea modelelor dintr-un anumit domeniu. Prima fază a acestui proces este specificarea și implementarea metodei de modelare pe o platformă de metamodelare. Următorul pas este specificarea modelelor din domeniu cu noul instrument de modelare specific. Dacă ne aflăm într-un blocaj, ceea ce înseamnă că platforma de metamodelare aleasă nu ne oferă suficient suport pentru a putea specifica modelele specifice domeniului, va trebui să o luăm de la capăt, prin crearea instrumentului de modelare specific peste o altă platformă de metamodelare. Dacă acest instrument de modelare specific ar fi independent de platformă, am putea evita această muncă suplimentară. Acest lucru este rezolvat de limbajul MM-DSL, care este gândit ca o alternativă la construirea unei metode de modelare independente de platformă. Independența platformei este realizată de translatoare specifice fiecărei platforme de metamodelare. Limbajul MM-DSL a fost specificat la Universitatea din Viena [Karagiannis2016] [Visic2016] și a fost implementată doar partea statică a descrierii metamodelelor. Noi am implementat, în aceasta teză, partea dinamică a limbajului, adică cea care oferă facilități pentru specificarea algoritmilor de calcul.

Procesul de trecere de la scrierea de cod la scrierea de modele duce inevitabil la necesitatea construirii de modele formale care pot fi verificate teoretic. Teoria categoriilor oferă un limbaj pentru descrierea precisă a modelelor, un anumit tip de gândire a modelelor numită gândire pe săgeți și o mulțime de instrumente în acord cu modelele de design și principiile structurale din practica MDE.

Teoria categoriilor oferă atât un limbaj, cât și o mulțime de mecanisme și construcții universale pentru gestionarea eficientă a modelelor. Toate modelele matematice pot fi organizate în funcție de structura lor pe categorii specifice, iar astfel de categorii specifice pot fi la rândul lor organizate într-o categorie mai generală care duce la integrarea mai multor modele. Un aspect important al modelării este construirea de funcții complexe dintr-un set dat de funcții simple, folosind diferite operații pe funcții precum compoziția și compoziția repetată. Teoria categoriilor este exact algebra potrivită pentru astfel de construcții.

Modelele categorice sunt un suport direct pentru modelarea și clasificarea matematică precisă a structurilor implicate în modelarea sistemelor precum și a transformărilor și operațiilor asupra modelelor vehiculate în practică MDE. Operațiile de compunere, transformare, instanțiere, sincronizare, etc. sunt execuții ale specificațiilor categorice [Diskin2012].

MDE se concentrează în mod esențial pe structurarea corectă a modelelor în submodele, care, la rândul lor, sunt structurate în continuare similar. Acest tip de modelare a sistemelor moderne necesită relații și operațiuni inter-structurale care nu sunt oferite de matematica clasică. In abordarea MDE sunt mai importante operațiunile și relațiile asupra obiectelor matematice, oferite de teoria categoriilor, decât structura lor internă. Așa cum mai multe modele pot fi compuse, prin anumite interacțiuni specifice, pentru a forma modele mai complexe tot la fel o mulțime de categorii pot fi compuse, prin morfisme specifice, pentru a forma o categorie mai complexă și prin urmare ambele sunt structuri holistice.

În această lucrare am specificat atât conceptul de metodă de modelare, cât și pe cel de modelare multi-nivel în limbajul oferit de teoria categoriilor. Parți importante din această teză au fost publicate în [Craciunean2018] [Craciunean2019] [Craciunean2019A] [Craciunean2019B] [Craciunean2019C] [Craciunean2019D].

1.1 Declararea problemei

Procesul de modelare presupune specificarea unui model, care imită un sistem real, dintr-un anumit punct de vedere, cu un anumit grad de fidelitate și care este capabil să interacționeze cu alte sisteme reale sau cu alte modele într-un mod similar cu sistemul real. De exemplu, modelul unui Cyber-Physical Production System (CPPS) implică interacțiunea acestuia cu o varietate de sisteme reale cum ar fi roboți, manipulatoare și modele software, cum ar fi componentele de control. De asemenea modelele software încorporate într-un autoturism trebuie să interacționeze cu componentele mecanice ale autoturismului sau cu alte componente software ale aceluiași autoturism sau ale altor autoturisme.

Prin urmare procesul de modelare are ca punct de plecare sistemele din lumea reală și ca punct final modelele executabile ale acestor sisteme. Trecerea de la sistemele din lumea reală la modele executabile este un proces complex, care presupune trecerea modelului prin cel puțin trei forme distincte: modelul conceptual, modelul formal și modelul executabil.

Modelul executabil este specificat într-un limbaj care poate fi interpretat sau executat de o mașină.

Modelul formal trebuie să fie specificat precis într-un limbaj formal, adică un limbaj cu reguli sintactice precise, cu o semantică precisă și cu reguli adecvate pentru prelucrarea informațiilor din domeniul de utilizare. În general scrierea unui model într-un limbaj formal presupune o experiență de durată în specificarea de modele formale și un efort important din partea modelatorului. Evitarea acestui neajuns se poate face doar prin utilizarea unui limbaj specific domeniului de expertiză al modelatorului.

Modelul conceptual se construiește prin contribuția tuturor parților implicate în procesul de modelare și prin urmare presupune o specificație înțeleasă de către toate aceste părți.

Pentru ca specificația unui model conceptual să fie eficientă, este nevoie ca aceasta să fie compatibilă cu instrumentul de modelare în care se face specificarea formală. Această compatibilitate se referă la existența unui morfism între modelul conceptual și modelul formal. Lipsa compatibilității între cele două modele implică mai multe iterații în ciclul de proiectare și prin urmare duce la creșterea costurilor de proiectare și dezvoltare.

Pentru ca specificarea unui model conceptual să fie eficientă iar modelul să poată fi înțeles de către toți participanții la procesul de dezvoltare, modelul formal, corespunzător, trebuie să beneficieze de un instrument de modelare adecvat domeniului specific de modelare.

Diversitatea și heterogenitatea sistemelor reale face ca dezvoltarea de instrumente de modelare și de limbaje de modelare specifice domeniului, corespunzătoare, să devină o problemă centrală a modelarii.

Limbajul de modelare trebuie să permită, cu efort minim, specificarea precisă și intuitivă a conceptelor și a modelelor din domeniul de modelare. Pentru a îndeplini acest deziderat un astfel de limbaj formal

trebuie să dispună de anumite tehnici naturale de modelare și analiză specifice domeniului și ca urmare va avea un domeniu limitat de aplicabilitate.

Este evident avantajul adus de instrumentele de modelare specifice domeniului înzestrate cu limbaje de modelare specifice domeniului (DSML), care aduc un salt de eficientă comparabil cu saltul de eficiență adus, în programare, de trecerea de la limbajele de asamblare la limbajele de nivel înalt. Acest avantaj derivă, în special, din faptul că un limbaj de modelare specific domeniului este gândit pentru modelatorii din domeniu nu pentru mașină sau pentru programatori. Dar construirea de instrumente specifice domeniului implică costuri. Prin urmare, este natural ca noi să dorim să construim un instrument de modelare specific domeniului eficient, rapid și cu costuri minime.

În urma cercetărilor efectuate în literatura de specialitate din domeniul mecanismelor și tehnologiilor de metamodelare care susțin formalizarea și implementarea metodelor de modelare am constatat că preocupările din acest domeniu sunt destul de rare.

Accentul principal, în literatura de specialitate, se pune pe studiul limbajelor existente, pe extinderea limbajelor existente pentru a acoperi noi necesități și pe construirea de limbaje noi și nu, suficient de mult, pe cunoștințele, mecanismele și instrumentele care să faciliteze dezvoltarea optimă a instrumentelor de modelare specifice domeniului. Preocupările legate de mecanismele de construire a unor limbaje de modelare formale specifice domeniului diagramatice pentru specificarea precisă, sistematică și incrementală a sistemelor complexe sunt greu de găsit.

O altă problemă majoră legată de implementarea instrumentelor de modelare este lipsa de portabilitate între diferitele platforme de metamodelare. După ce am ales un instrument de metamodelare și am implementat conceptul de modeling method pe acesta următorul pas este să începem să specificăm modelele concrete cu acest instrument de modelare specific. Dacă platforma de metamodelare aleasă nu ne oferă suficient suport pentru a putea specifica eficient modelele, va trebui să o luăm de la început prin implementarea conceptului de modeling method peste o altă platformă de metamodelare . Dacă acest instrument de modelare specific ar fi independent de platformă, am putea evita această muncă suplimentară.

Daca ne referim la formalizarea matematică, dezvoltarea instrumentelor de modelare este o activitate mai mult experimentală și intuitivă, bazată pe fundamente semantice surprinzător de slabe. Formalizarea în limbaj matematic a modelelor face ca acestea să devină obiecte matematice cu proprietăți adecvate pentru analiză și verificare. Limbajul oferit de teoria categoriilor are o sintaxă implicită și o semantică implicită și oferă algoritmi generic care pot fi implementați la nivel de metamodel. Deși este cel mai expresiv limbaj matematic pentru specificarea modelelor este puțin utilizat.

Prin urmare, importanța dezvoltării de instrumente de modelare specifice domeniului asociată cu lipsa preocupărilor satisfăcătoare în domeniu, expresivitatea remarcabilă a mecanismelor categorice în specificarea modelelor asociată cu utilizarea insuficientă a acestora în formalizarea matematică a metamodelelor, dependența de platformă în implementarea instrumentelor de modelare, compun motivația principală a cestei teze.

In această idee ne propunem să răspundem la întrebarea generală:

RQ. Care sunt mecanismele teoretice și practice care asigură construcția metodologică optimă a instrumentelor de modelare?

Pentru a sistematiza răspunsul la această întrebare generală, o vom descompune în câteva întrebări mai precise care vor sublinia reperele de bază ale acestei teze.

RQ1. Care sunt mecanismele matematice adecvate pentru o formalizare a modelelor compatibilă cu instrumentele de metamodelare?

Dacă efortul de programare presupune scrierea de clase sau module de program în cazul modelarii problema principală devine compoziția acestor componente, în conformitate cu interacțiunile complexe dintre conceptele pe care le reprezintă.

Componentele software care se compun într-un model reprezintă concepte din lumea reală și prin urmare nu sunt doar noduri într-un graf, ci sunt structuri algebrice încapsulate, iar modelarea se concentrează pe operații și relații asupra structurilor considerate entități holistice.

Matematica clasică oferă mecanisme pentru modelarea conceptelor atomice dar nu oferă mecanisme adecvate pentru specificarea relațiilor și operațiilor inter-structurale și a raționamentului asupra, acestora care sunt esențiale în specificarea modelelor. Teoria categoriilor oferă astfel de mecanisme, mai mult aceste mecanisme sunt chiar esența teoriei categoriilor. Gândirea pe săgeți specifică teoriei categoriilor este adecvată relațiilor inter-structurale, specifice modelelor. Aceste săgeți acoperă toate tipurile de interacțiuni dintre structurile modelelor, de la relațiile simple m la n specifice modelelor ER până la relații complexe functoriale care transferă proprietățile unei structuri în termenii altei structuri.

RQ2. Care sunt mecanismele matematice adecvate pentru specificarea sintaxei limbajelor diagramatice?

Proiectarea și implementarea instrumentelor de modelare specifice domeniului a devenit în zilele noastre o etapă integrantă a procesului de modelare în conformitate cu conceptul MDE. Limbajele de modelare specifice domeniului (DSML) integrate în aceste instrumente de modelare sunt în cele mai multe cazuri limbaje diagramatice, deoarece acestea sunt mai intuitive. Dacă în cazul limbajelor textuale avem gramatica context free ca mecanism standard de specificare a acestora cu algoritmi puternici pentru analiza lexicală și sintactică a acestora, în cazul limbajelor diagramatice lucrurile sunt încă în faza de căutare a mecanismelor și algoritmilor echivalenți.

Un model diagramatic are două dimensiuni; o dimensiune statică și o dimensiune comportamentală. Prin urmare specificarea sintaxei unui limbaj de modelare diagramatic presupune specificarea sintaxei celor două dimensiuni.

Noi credem că schița categorică este un mecanism adecvat pentru specificarea sintaxei dimensiunii statice a unui model. O schiță categorică este un tuplu format dintr-un graf și o mulțime de constrângeri asupra componentelor grafului.

Pentru a defini sintaxa dimensiunii comportamentale am introdus noțiunea de regulă comportamentală care compune o transformare graf cu o acțiune comportamentală. Transformările graf specifică transformări locale ale grafurilor, iar acțiunile comportamentale sunt funcții care modifică valorile atributelor asociate componentelor graf ale modelului.

RQ3. Care sunt mecanismele matematice adecvate pentru specificarea semanticii limbajelor diagramatice la nivel de metamodel?

Semantica statică a unui model va fi definită prin maparea atributelor asociate cu conceptele modelului la domenii de valori precum și prin maparea structurii modelului la structuri cunoscute.

Așa cum am văzut, o regulă comportamentală este o agregare între o transformare graf și o acțiune comportamentală. Semantica comportamentală a modelelor diagramatice am definit-o prin maparea regulilor comportamentale la algoritmi și mecanisme, implementate la nivel de metamodel, care execută transformările graf și acțiunile. Transformarea graf are, în abordarea noastră, două funcții și anume, localizarea zonei de definiție a acțiunii comportamentale și modificarea structurii graf a modelului. În teoria categoriilor transformarea graf se obține printr-un dublu pushout.

RQ4. Cum poate fi specificată execuția și simularea unui model cu mecanisme categorice?

Semantica comportamentală a modelelor diagramatice am definit-o prin reguli comportamentale definite la nivel de metamodel. Am definit stările unui model ca instanțe ale modelului static iar tranzițiile ca reguli comportamentale. Mulțimea instanțelor unui model static împreună cu mulțimea regulilor comportamentale care pot fi definite între aceste instanțe formează o categorie. Am numit această categorie, categoria instanțelor și a transformărilor comportamentale (CIBT). Această categorie definește evoluția posibilă a modelului în timpul simulării sau execuției. Fiecare instanță reprezintă o stare posibilă a procesului de execuție corespunzător modelului. Fiecare stare este caracterizată, prin urmare, de valorile atributelor asociate componentelor modelului și de structura graf a instanței în cauză. Rezultatul este un proces reprezentat de un sistem de tranziție care reprezintă comportamentul real al sistemului modelat [Milner2009] [Aalst2011] [Weske2012].

RQ5. Cum poate fi abordată metodic proiectarea instrumentelor de modelare?

Unul dintre obiectivele principale ale acestei teze este de a identifica un flux optimal pentru proiectarea și implementarea instrumentelor de modelare. Aceasta abordare presupune secvențierea fazelor de construire a instrumentului de modelare în vederea distribuirii complexității procesului de dezvoltare în pași relativ ușor de gestionat. Procesul de dezvoltare se sprijină pe conceptul de metodă de modelare pe dimensiunea metodică și pe un instrument de metamodelare pe dimensiunea tehnologică. În abordarea noastră elementul central al procesului de dezvoltare este conceptul de modeling method introdus de Karagiannis și Kühn [Karagiannis2002].

Conceptul de modeling method integrează trei componente esențiale ale unui instrument de modelare și anume limbajul de modelare, algoritmii și mecanismele și procedura de modelare. Cu aceste componente implementate, un instrument de modelare, permite specificarea modelelor, din domeniul de modelare, în limbajul de modelare în conformitate cu procedura de modelare, și va putea fi validat, analizat, executat sau simulat utilizând algoritmii și mecanismele.

RQ6. Cum pot fi specificate componentele conceptului de modeling method în contextul mecanismelor categorice?

Principalele componente ale conceptului de modeling method sunt limbajul de modelare, algoritmii și mecanismele și procedura de modelare. Scopul acestor componente este de a permite specificarea și manipularea eficientă a conceptelor din domeniul de modelare. Pentru implementarea cu succes a unei

instanțe a conceptului de modeling method nu este suficient ca aceste componente să fie specificate informal, ci este nevoie de o specificare formală precisă a acestora.

Formalizarea unui limbaj presupune specificarea formală a celor trei componente esențiale din definiția unui limbaj și anume sintaxa, domeniul semantic și maparea semantică [Rumpe2016]. Pentru specificarea formală a acestor componente avem nevoie de un limbaj formal. În această lucrare noi folosim limbajul oferit de teoria categoriilor pentru specificarea formală a unui limbaj diagramatic.

Proiectarea unui DSML trebuie să includă mecanisme de specificare a celor două dimensiuni ale unui model și anume dimensiunea statică și dimensiunea comportamentală. Ambele dimensiuni trebuie specificate prin agregarea acelorași concepte atomice care reprezintă conceptele din domeniul de modelare. Semantica unui model se definește prin maparea construcțiilor sintactice la un domeniu semantic adecvat care dă sens acestor modele.

Pentru specificarea sintaxei dimensiunii statice a modelelor vizuale noi folosim schița categorică, care oferă un limbaj formal declarativ înzestrat cu o semantică implicită. Prin urmare și mecanismul de specificare a dimensiunii statice a unui limbaj de modelare vizual este tot schița categorică. Constrângerile asociate unei schițe categorice permit construirea unor algoritmi generic la nivel de metamodel pentru verificarea corectitudinii sintactice a modelelor.

Semantica unui model static se definește prin maparea atributelor asociate componentelor atomice ale acestuia la domenii de date corespunzătoare și prin maparea structurilor graf la structuri cu semantica bine definite. Un model static reprezintă o stare pentru dimensiunea comportamentală.

Pentru specificarea sintaxei dimensiunii comportamentale a unui model vizual noi folosim conceptul de regulă comportamentală care este o asociere dintre o transformare graf și o acțiune comportamentală. Semantica transformărilor graf se poate definii printr-un double pushout care poate modifica structura graf a modelului static transformându-l într-un alt model static, iar semantica acțiunilor comportamentale prin funcții care calculează valorile atributelor noului model static pornind de la valorile atributelor vechiului model static.

Mecanismele și algoritmii sunt componente funcționale atât pentru limbajul de modelare, cât și pentru procedura de modelare. Componentele generice sunt resurse funcționale pentru toate implementările metodelor de modelare, componentele specifice numai pentru anumite implementări, iar componentele hibride sunt generice, dar pot fi configurate pentru implementări multiple [Karagiannis2016]. Unul dintre marile avantaje ale modelării categorice este că teoria categoriilor este o resursă importantă de componente generice. Grafurile, homomorfismele de grafuri, categoriile, functorii, diagramele, transformările naturale, conurile, coconurile, limitele, colimitele sunt doar cateva dintre aceste construcții cu un caracter generic și care pot fi implementate ca mecanisme sau algoritmi, independent de orice model diagramatic care le folosește.

RQ7. Cum putem realiza portabilitatea metamodelelor de la o platformă de metamodelare la alta ?

Dacă după implementarea conceptului de modeling method, din anumite motive, trebuie schimbată platforma de metamodelare, din cauza lipsei de portabilitate între diverse platforme, pierdem toată munca depusa pentru implementare și trebuie să o luăm de la început prin implementarea conceptului de modeling method pe noua platformă. Dacă acest instrument de modelare specific ar fi independent de platformă, am putea evita această muncă suplimentară. Acest neajuns poate fi remediat prin utilizarea limbajului MM-DSL, care este gândit ca o alternativă pentru specificarea unei metode de

modelare, care este independent de platformă. Independența de platformă este realizată de translatoare specifice fiecărei platforme de metamodelare.

Limbajul MM-DSL a fost conceput și specificat la Universitatea din Viena și include cele mai relevante concepte de metamodelare [Visic2016]. Limbajul conține un subset de instrucțiuni descriptive care permit definirea componentelor unei metode de modelare și un subset de instrucțiuni de calcul și control al fluxului care permit specificarea algoritmilor. Partea descriptivă a limbajului a fost implementată în lucrarea [Visic2016], iar în această lucrare am finalizat implementarea limbajului cu partea de calcul și control, adică construcțiile de limbaj care permit specificarea algoritmilor.

RQ8. Cum pot fi asamblate mecanismele prezentate în această lucrare pentru implementarea unui instrument de modelare concret?

Modelarea anumitor aspect ale comportamentului unui sistem prin procese este o abordare frecventă în practica de modelare [Aalst2011]. Pentru a demonstra modul în care mecanismele prezentate în această lucrare pot fi asamblate în procesul de implementare a unui instrument de modelare am prezentat două posibile implementări pentru două instrumente de modelare, ambele adecvate pentru specificarea proceselor, dar cu grad de specificitate diferite.

Am ales, pentru exemplificare, o metodă de modelare cu un limbaj bine cunoscut: Petri Nets și un limbaj specific pentru modelarea proceselor de fabricație. În ambele cazuri am folosit limbajul oferit de teoria categoriilor pentru specificarea matematică a componentelor conceptului de modeling method iar apoi am implementat aceste exemple folosind ca suport limbajul MM-DSL cu translatare pe platforma ADOxx.

1.2 Conținut și contribuții

Secțiunea 2 a lucrării prezintă fundamentele teoretice și conceptuale specifice domeniului de modelare precum și o scurtă analiză a diferitelor tehnologii de metamodelare existente. Această secțiune este rezultatul studiului sistematic al mecanismelor și cunoștințelor din domeniul de cercetare și urmărește introducerea unei înțelegeri comune a conceptelor de metamodelare, modelare conceptuală și dezvoltare bazată pe model. Conținutul acestei secțiuni reprezintă suportul teoretic și instrumental pentru identificarea și motivarea temei noastre de cercetare precum și pentru integrarea rezultatelor cercetării în acest domeniu de cercetare.

Secțiunea 3 introduce noțiunile de model și metamodel prin combinarea conceptului de model static, care reprezintă o stare a modelului, cu noțiunea de regulă comportamentală și reunește aceste concepte într-un model pe mai multe nivele.

Contribuția acestei secțiuni constă în specificarea, în contextul mecanismelor categorice, a conceptelor de modelare pe mai multe niveluri, ierarhiei de modelare pe mai multe niveluri, simulare și co-simulare. Am definit un model static ca imaginea unui functor definit pe o schiță categorică, numită metamodel, care respectă constrângerile impuse și care reprezintă o stare a modelului. În abordarea noastră (secțiunea 3.3.), o stare a unui model este reprezentată de o structură statică și un set de valori ale atributelor.

Pentru a defini comportamentul unui model (secțiunea 3.4.), am introdus noțiunea de regulă comportamentală care transformă structura modelului și atributele în limita permisă de constrângerile

atașate. Aceste reguli de transformare acționează local, ceea ce înseamnă că modifică mai multe substări locale în funcție de anumite condiții prealabile. O regula comportamentală este o descriere generală a transformărilor locale care pot apărea într-un model și determină comportamentul acestuia în funcție de evoluția sistemului modelat. În principiu, o regulă comportamentală constă în înlocuirea unei părți a modelului cu o altă parte, precum și calcularea sau recalcularea valorilor unor atribute, în conformitate cu constrângerile impuse de metamodel.

Pentru a defini generic noțiunea de regulă comportamentală, am introdus noțiunile de semnătură diagramatică a unei reguli comportamentale și de semnătura diagramatică a unei acțiuni care specifică modul de transformare a elementelor graf la un nivel generic. De asemenea, în același scop, am introdus noțiunea de trei-diagrama, care este un mecanism pentru asocierea componentelor formale ale grafului, reprezentate prin forme graf, la componentele reale din graful schiței categorice.

Am asamblat apoi toate aceste elemente în conceptul de schiță categorică comportamentală, care definește comportamentul modelelor la nivel de metamodel. Pe baza schiței comportamentale am definit apoi conceptul de model al unei schițe comportamentale pe care l-am numit model comportamental (secțiunea 3.5.).

Evident, un model de sistem implică combinarea modelului static cu modelul comportamental în același format diagramatic pentru a permite modelatorului să testeze și să analizeze sistemul real. În acest fel, am definit un metamodel prin cuplarea metamodelului static cu cel comportamental. De aici rezultă că un model, la rândul său, se obține prin cuplarea celor două dimensiuni specificate de metamodel, și anume dimensiunea statică și dimensiunea comportamentală, prima dimensiune specifică stările modelului și a doua tranzitia de la o stare la alta.

În secțiunea 3.6. am introdus noțiunea de metamodel și noțiunea de ierarhie de modelare pe mai multe niveluri. O ierarhie de modelare pe mai multe niveluri poate fi reprezentată de un arbore, care are rădăcina reprezentată de cel mai abstract metamodel, nodurile interioare sunt metamodele, iar frunzele sunt modele. Prin urmare, fiecare model este caracterizat direct sau indirect de toate metamodelele care se află pe un lanț de tipuri. O ierarhie de modelare pe mai multe niveluri specifică o ierarhie care conține o familie de modele structurată pe diferite niveluri de abstractizare, care au o rădăcină comună și două dimensiuni în care pot fi localizate, o dimensiune statică și o dimensiune comportamentală. Dimensiunea statică este reprezentată la fiecare nivel i al unui lanț de tipizare. Deoarece tipizarea este unică în fiecare ierarhie, fiecare nod, cu excepția rădăcinii, are exact un nod părinte în ierarhie. Pentru ca fiecare element să aibă un tip și procesul de modelare să fie finit, nodul rădăcină conține o colecție de tipuri individuale auto-definite.

Tot în secțiunea 3.6. am introdus noțiunea de trei-diagramă indirectă, care este o nouă trei-diagramă obținută prin compunerea unui trei-diagrame cu o potrivire p. Modelarea pe mai multe niveluri creează astfel oportunitatea implementării comportamentului la un nivel ridicat de abstractizare, prin reguli comportamentale generice definite la nivelurile superioare și utilizarea acestora la nivelul modelelor concrete.

Putem observa că, în abordarea noastră, semnăturile comportamentale pot fi introduse la diferite niveluri de abstractizare împreună cu trei-diagramele corespunzătoare și sunt apoi mapate la trei-diagrame indirecte la reguli comportamentale care se propagă până la cel mai puțin abstract nivel,

adică la nivelul de modelul concret. Prin urmare, regulile comportamentale pot fi implementate generic, la un nivel ridicat de abstractizare și apoi aplicate modelului concret.

Transformările comportamentale definite în această lucrare reprezintă o bază solidă pentru implementarea unui motor generic de execuție a modelului care permite simularea modelelor. În abordarea noastră, tranziția de la o stare la alta se face prin transformări comportamentale. Desigur, anumite transformări comportamentale pot fi realizate în paralel în procesul de evoluție al modelului. Plecând de la un rezultat teoretic care spune că orice două transformări comportamentale pot fi compuse într-o singură transformare, numită transformare E-concurentă care cumulează aplicarea a două transformări secvențiale într-o singură transformare [Ehrig2006] [Ehrig2015], am organizat procesul de execuție a unui model într-o categorie pe care am numit-o categoria instanțelor și transformărilor comportamentale care are ca obiecte instanțe ale modelului static și ca săgeți transformări comportamentale. Transformările comportamentale dintre două instanțe ale modelului sunt astfel reprezentate de macro-tranziții declanșate de starea reprezentată de instanța inițială.

Deoarece potrivirea într-un model este o problemă complexă care implică algoritmi NP, ne propunem să rezolvăm această problemă prin integrarea transformărilor comportamentale în unele dintre obiectele și arcele modelului. Mai exact, propunem integrarea unor apeluri la procedurile asociate transformărilor comportamentale generice. Apelurile de procedură vor fi efectuate cu parametrii corespunzători obținuți prin indirectarea trei-diagramelor. Aceste proceduri pot fi implementate ca proceduri reactive, adică se vor activa de fiecare dată când starea locală a modelului suferă o transformare.

Plecând de la ideea că simularea începe de la o stare inițială, am definit categoria de simulare a proceselor (PSC) ca o subcategorie a categoriei instanțelor și regulilor comportamentale (CIBR). Categoria PSC permite studiul comportamentului unui sistem pe baza urmelor modelului în procesul de simulare, care se identifică cu căile din categoria PSC care au ca obiect de pornire instanța inițială.

Tot in această secțiune se face și o primă evaluare a conceptelor introduse prin scurte exemple relevante și un studiu de caz.

Secțiunea 4 prezintă utilizarea mecanismelor categorice pentru a specifica conceptul de metodă de modelare, prezentare însoțită de specificarea și implementarea pe platforma de metamodelare ADOxx a unui exemplu de metodă de modelare. Contribuția esențială originală a secțiunii 4 este legată de specificația categorică a conceptului de metodă de modelare.

Structura unui model și comportamentul său sunt dimensiuni esențiale ale unui model și fiecare dintre ele are propria sintaxă și semantică. Deși există o legătură strânsă între semantica structurală a unui model și semantica comportamentală a acestuia, acestea sunt totuși diferite.

Condiția esențială pentru ca un limbaj să permită definirea unei semantici precise este să aibă o sintaxă riguroasă, care specifică în mod clar și fără ambiguitate construcțiile sintactice permise. Schița categorică este un concept ideal pentru a specifica riguros sintaxa unui limbaj. Schița categorică se asociază foarte bine cu transformările graf care formează, în opinia noastră, un mecanism riguros, expresiv și eficient pentru specificarea comportamentului unui model.

Semantica unei structuri de model implică maparea atributelor la setul lor de valori, și interpretarea structurii grafice a modelului, adică maparea construcțiilor sintactice la structuri bine definite precum

secvențiale, repetitive, recursive, join, fork etc. și definirea de operații pe atribute în contextul acestor structuri.

Semantica comportamentală a unui model implică maparea modelului la o mulțime de predicate, care verifică starea modelului, la o mulțime de acțiuni (funcții) care recalculează valorile atributelor și la o mulțime de transformări graf care transformă structura grafică a modelului prin repoziționarea componentelor, adăugarea sau eliminarea de componente.

Această abordare oferă o resursă importantă de componente generice, cum ar fi: grafuri, morfisme de grafuri, categorii, functori, diagrame, transformări naturale, conuri, coconuri, limite, colimite etc., care devin concepte fundamentale în modelarea diagramatică. Toate aceste construcții au un caracter generic și pot fi implementate ca tipuri de date, mecanisme sau algoritmi generice, independent de orice model diagramatic care le folosește.

Prezentarea este însoțită de specificarea și implementarea unui exemplu de metodă de modelare.

Secțiunea 5 prezintă mecanismele pentru definirea și implementarea limbajului MM-DSL.

Contribuția noastră esențială în secțiunea 5 este implementarea parțială a limbajului MM-DSL. Limbajul MM-DSL a fost conceput și specificat la Universitatea din Viena de Niksa în teza sa de doctorat [Visic2016] sub îndrumarea profesorului Karagiannis. Limbajul conține un subset de instrucțiuni descriptive care permit definirea componentelor unei metode de modelare și un subset de instrucțiuni de calcul și control al fluxului care permit specificarea algoritmilor. Partea descriptivă a limbajului a fost implementată în lucrare [Visic2016] iar noi în această lucrare am finalizat implementarea limbajului cu partea de calcul și control, adică construcțiile de limbaj care permit specificarea algoritmilor.

Conceptele de bază ale limbajului MM-DSL și relațiile dintre acestea, precum și conceptele specifice unui model de aplicație sunt specificate de gramatica instrucțiunilor de metamodelare. Reprezentarea grafică a acestor concepte este specificată prin gramatica instrucțiunilor de reprezentare grafică. Specificarea mecanismelor și algoritmilor este o caracteristică esențială oferită de gramatica instrucțiunilor de specificare a algoritmilor.

La această contribuție esențială putem adăuga implementarea metodei de modelare SMM în limbajul MM-DSL, care demonstrează facilitățile esențiale oferite de limbajul MM-DSL pentru implementarea metodelor de modelare.

În secțiunea 4 am formalizat conceptele de bază ale unei metode de modelare în limbajul teoriei categoriilor. Putem observa că limbajul MM-DSL ne oferă toate facilitățile necesare pentru a putea specifica o metodă de modelare categorică. Formalizarea bazată pe teoria categoriilor introdusă în secțiunile 4 indică faptul că limbajul MM-DSL oferă toate facilitățile pentru specificarea conceptelor de bază ale unei metode de modelare: reprezentarea obiectelor formale care pot fi clase care conțin atribute, instanțe ale claselor care conțin atribute, elemente individuale sau seturi de elemente, funcții sau seturi de funcții și evenimente; reprezentarea de funcții formale care pot fi funcții matematice, relații sau arce în sensul teoriei grafurilor.

Prin urmare, limbajul MM-DSL acceptă implementarea completă a unui concept de metodă de modelare într-un instrument de modelare specific domeniului. Acest instrument include pe lângă un limbaj de modelare și funcționalități de bază, specifice unei clase de modele, precum și îndrumare și

constrângeri pentru scenarii de modelare în funcție de diferite scopuri. În acest fel, un model nu este doar o specificație vizuală a unui model, ci moștenește și un comportament specific din metamodel și permite, de asemenea, procesarea structurală și semantică, inclusiv raționamentul asupra structurii sau proprietăților specifice ale elementelor modelului.

Secțiunea 6 prezintă formalizarea și implementarea a două exemple relevante de metode de modelare, utilizând limbajul MM-DSL.

Contribuția esențială din secțiunea 6 este specificarea și implementarea a două exemple relevante de metode de modelare, și anume unul referitor la construirea unui instrument de modelare bazat pe limbajul Petri Nets și celălalt bazat pe un DSML pentru specificarea unui anumit tip de proces de fabricație. Prin urmare, am exemplificat facilitățile oferite de teoria categoriilor împreună cu limbajul MM-DSL pentru formalizarea și implementarea conceptului de metodă de modelare. Această secțiune are rolul de a evalua și valida mecanismele teoretice și conceptuale introduse sau identificate în această teză.

Secțiunea 7 încheie lucrarea prin câteva concluzii și un scurt rezumat al contribuțiilor personale.

2 Fundamente teoretice si conceptuale

Această secțiune prezintă fundamentele teoretice și conceptuale specifice domeniului de modelare precum și o scurtă analiză a diferitelor tehnologii de metamodelare existente. Conținutul acestei secțiuni este rezultatul studiului sistematic al mecanismelor și cunoștințelor din domeniul de cercetare și urmărește introducerea unei înțelegeri comune a conceptelor de metamodelare, modelare conceptuală și dezvoltare bazată pe model. Conținutul acestei secțiuni reprezintă suportul teoretic și instrumental pentru identificarea și motivarea temei noastre de cercetare precum și pentru integrarea rezultatelor cercetării în acest domeniu de cercetare.

3 Modelarea categorică multi-nivel

Specificațiile diagramatice sunt răspândite în modelare. La început, notațiile diagramatice au fost utilizate în principal ca specificații de comunicare între experții în modelare, proiectanții de software și programatori. Această abordare a condus la o modelare diagramatică, în care sensul semantic al unei construcții era aproximativ și confuz.

În prezent, modelarea diagramatică este o parte integrantă a unor concepte precum Model-Driven Engineering (MDE), Model-Driven Development (MDD) sau Model-Driven Architecture (MDA), care sunt nume diferite ale unei abordări comune care are ca scop mutarea efortului de la scrierea codului la scrierea de modele, ca artefacte principale în dezvoltarea de software și apoi generarea de cod direct din aceste modele.

În contextul MDE, modelarea se bazează pe bidimensionalitatea conceptuală a sistemelor din lumea reală și, prin urmare, un model este reprezentat în mod natural de diagrame. Obiectivele actuale ale

MDE impun o sintaxă formală și o semantică precisă și concisă a notațiilor diagramatice utilizate pentru a specifica modelele. Reprezentarea modelelor prin șiruri de simboluri le aplatizează structura și ascunde legăturile dintre ele, ducând astfel la specificații voluminoase, rigide și dificil de înțeles, validat și analizat.

Teoria categoriilor oferă o abordare bazată pe diagrame, care se concentrează în special pe relațiile dintre componentele unui model. Această abordare are ca mecanism principal schița categorică care oferă un formalism universal pentru definirea sintaxei și semanticii limbajelor diagramatice echivalent cu formalismul EBNF utilizat pentru definirea sintaxei limbajelor de programare. Abordarea noastră cu privire la utilizarea schiței categorice ca mecanism de specificare a dimensiunii statice a modelelor este prezentată în secțiunea 3.2.

În abordarea noastră, un model diagramatic este specificat în prima fază, static, și apoi evoluția sa este specificată prin transformări ale modelului. Prin urmare, modelele statice sunt imagini ale unor functori definiți pe schițe categorice care sunt metamodele. Detaliile acestei abordări sunt prezentate în secțiunea 3.3.

Grafurile sunt o modalitate intuitivă de a reprezenta stările unui sistem în limbajele vizuale. În acest context, comportamentul sistemelor, astfel reprezentat, poate fi modelat prin transformări graf care exprimă modificări locale ale grafurilor și, prin urmare, sunt foarte potrivite pentru a descrie transformările locale ale stărilor modelului. Secțiunea 3.4. abordează problema modelării comportamentului unui model prin transformări graf.

Secțiunea 3.5. introduce noțiunile de model și metamodel prin combinarea conceptului de model static, care reprezintă o stare a modelului, cu noțiunea de transformare comportamentală și conceptul de model comportamental. Evident, un model de sistem implică combinarea celor două modele în același format diagramatic pentru a permite modelatorului să testeze și să analizeze un sistem real.

Secțiunea 3.6. prezintă abordarea noastră pentru modelarea pe mai multe niveluri ca o metodă de abordare a reducerii complexității modelului prin abstractizarea și distribuirea acestei complexități pe mai multe niveluri.

Secțiunea 3.7 tratează transformările comportamentale ca o bază solidă pentru implementarea unui motor generic de execuție a modelului care permite simularea modelului. Principalele avantaje ale transformărilor comportamentale sunt date de natura lor generică, o proprietate care permite implementarea lor la un nivel ridicat de abstractizare și natura lor vizuală.

4 Metoda de modelare categorică

Deși există o mulțime de instrumente de modelare, diversitatea și eterogenitatea sistemelor modelate în zilele noastre face ca acestea să fie prea generale în multe cazuri și prin urmare nu oferă mecanisme pentru a putea specifica eficient modelele din anumite domenii specifice. Soluția la această problemă este dezvoltarea de noi instrumente de modelare care să ofere toate mecanismele necesare specificării eficiente a modelelor din aceste domenii.

Conceptul de modeling method [Karagiannis2011][Karagiannis2016] abstractizează noțiunea de instrument de modelare, scoate în evidență componentele fundamentale ale unui instrument de modelare și concentrează eforturile unui grup mare de cercetători pentru găsirea celor mai potrivite metode pentru implementarea unor astfel de instrumente cu eforturi minime. În această secțiune ne concentrăm pe utilizarea mecanismelor categorice în procesul de formalizare și implementare a conceptului de modeling method într-un instrument de modelare.

Principalele componente ale conceptului de modeling method sunt limbajul de modelare, algoritmii și mecanismele, și procedura de modelare. Scopul acestor componente este de a permite specificarea și manipularea eficientă a conceptelor din domeniul de modelare. Am văzut în secțiunea 3 că modelele diagramatice sunt caracterizate de două dimensiuni, o dimensiune statică și o dimensiune comportamentală. Prin urmare aceste componente ale unui modeling method trebuie să permită specificarea și manipularea ambelor dimensiuni ale modelelor la nivel de metamodel. Teoria categoriilor oferă mecanismele adecvate pentru specificarea generică a acestor componente. Dimensiunea statică a unui model am specificat-o, așa cum am văzut în secțiunea 3, prin schița categorică iar dimensiunea comportamentală prin reguli comportamentale, concepte matematice care au o sintaxă și o semantică formală.

În modelul bazat pe teoria categoriilor, mecanismele și algoritmii sunt construcții universale. Pentru a specifica sintaxa unui metamodel diagramatic am folosit schița categorică, care la rândul său este reprezentată într-un limbaj grafic. Schița se bazează pe un graf și constrângeri: diagrame comutative, limite și colimite în categorii care sunt constructe universale și, prin urmare, sunt independente de metamodel. Conceptul de transformare naturală are și elemente de universalitate, cum ar fi proprietatea de naturalitate, care poate fi implementată generic. Săgețile graficului sunt operatori ai schiței care pot fi implementați la nivelul meta-metamodelului, eventual cu mici ajustări la nivelul metamodelului. Diagramele comutative, conurile si coconurile, pot fi implementate complet la nivelul meta-metamodelului, asigurând astfel corectitudinea sintactică a oricărui metamodel specificat de o schiță.

Mecanismele contribuie la creșterea eficienței modelării prin reutilizarea și standardizarea software-ului. Un mecanism este o entitate software reutilizabilă independentă, autonomă, care implementează o varietate de interfețe pentru relația cu modelul specificat. Astfel, un mecanism este un cod executabil care poate fi cuplat la codul altor componente ale unui model. Algoritmii care vin cu o metodă de modelare, în general, interpretează construcțiile modelului pe baza principiilor universal valabile, pentru implementarea funcționalității acestora.

Din mai multe motive, este importantă specificarea semanticii modelelor, în cea mai mare parte, la nivel de metamodel. Dacă luăm în considerare modelul bazat pe teoria categoriilor, atunci construcțiile universale din teoria categoriilor pot fi implementate la nivel de metamodel ca un pachet de mecanisme și algoritmi care vor funcționa coerent în toate modelele specificate.

O parte din conținutul acestei secțiuni a fost inclusă de autorul tezei în [wp32020].

5 MM-DSL, specificare și implementare

Atunci când procesul de modelare include și implementarea unui DSML, primul pas este, alegerea unui instrument adecvat de metamodelare și apoi specificarea și implementarea metodei de modelare pe platforma selectată. În pasul următor vom începe specificarea modelelor concrete cu noul instrument de modelare specific domeniului de modelare. Dacă ne aflăm într-un blocaj, în sensul că platforma de metamodelare aleasă nu ne oferă suficient suport pentru a putea specifica procesele specifice din domeniu, va trebui să o luăm de la început prin crearea unui instrument de modelare specific peste o altă platformă de metamodelare. Dacă acest instrument de modelare specific ar fi independent de platformă, am putea evita această muncă suplimentară. Acest lucru este rezolvat de limbajul MM-DSL, care este gândit ca o specificație alternativă, independentă de platformă, pentru implementarea unei metode de modelare. Independența de platformă este realizată de translatori specifici fiecărei platforme de metamodelare.

Limbajul MM-DSL a fost conceput și specificat la Universitatea din Viena și include cele mai relevante concepte de metamodelare [Visic2015] [Visic2016]. Limbajul conține o submulțime de instrucțiuni descriptive care permit definirea componentelor unei metode de modelare și o submulțime de instrucțiuni de calcul și control al fluxului care permit specificarea algoritmilor. Partea descriptivă a limbajului a fost implementată în lucrarea [Visic2016] iar în această lucrare am finalizat implementarea limbajului cu partea de calcul și control, adică construcțiile de limbaj care permit specificarea algoritmilor.

Conceptualizarea unei metode de modelare depinde în mare măsură de platforma de metamodelare pe care urmează să fie implementată [Fill2013] [Visic2014]. Specificarea artefactelor specifice metamodelului, cum ar fi sintaxa abstractă, semantica abstractă, algoritmii generici specifici, depinde de platforma selectată pentru implementarea metodei de modelare. Această dependență este principalul motiv care a determinat dezvoltarea unui DSL, care ar trebui să standardizeze procesul de conceptualizare, implementare și evaluare a unei metode de modelare. MM-DSL oferă facilitatea de a conceptualiza și implementa un concept de metodă de modelare independent de platforma pe care va opera. Desigur, această caracteristică este condiționată de implementarea unui translator specific acestei platforme. În acest fel, codul MM-DSL poate fi translatat și apoi compilat și executat pe diferite platforme de metamodelare. În această secțiune prezentăm primul translator al limbajului MM-DSL care translatează o metodă de modelare specificată în limbajul MM-DSL pe platforma de metamodelare ADOxx.

Există multe tipuri de mecanisme utilizate pentru modelarea sintaxei unui limbaj, inclusiv: teoria matematică a limbajelor formale și mecanismele algebrice. Sintaxa limbajelor independente de context este adesea definită de variantele mai răspândite ale gramaticilor generative BNF și EBNF. În această secțiune prezentăm, pe scurt, specificațiile EBNF ale limbajului MM-DSL, precum si instrumentele și mecanismele pe care le-am folosit în implementarea acestui DSL.

Sintaxa unui limbaj este, de fapt, un mecanism de reprezentare simbolică a semanticii. Prin urmare implementarea semanticii DSML-urilor se poate face prin mecanisme operaționale, adică prin translatarea sintaxei obiectelor specificate în DSML în sintaxa unui limbaj cu o semantică

implementată deja pe o mașină [Voelter2010]. Noi vom translata, prin urmare, modelele specificate în limbajul MM-DSL în obiecte specificate în limbajul ADOScript.

În această lucrare noi am folosit framework-ul Xtext care conține toate facilitățile necesare pentru a specifica sintaxa limbajului MM-DSL în limbajul EBNF, pentru analiza lexicală, analiza sintactică și construirea arborelui de sintaxă abstractă (AST) în formatul modelelor Eclipse Modeling Framework (EMF). Pentru a oferi aceste facilități, Xtext include instrumental ANTLR (ANother Tool for Language Recognition), care implementează algoritmul de analiză sintactică LL(*). Pentru a specifica translatorul de la sintaxa abstractă a limbajului MM-DSL la sintaxa abstractă a limbajului ADOScript, folosim limbajul Xtend care facilitează procesul de dezvoltare a DSL-urile. Platforma Eclipse Modeling Framework (EMF) [Dave2009] [Eric2009] este gazda tuturor acestor instrumente și de asemenea oferă formatul de reprezentare a modelelor AST.

Dezvoltarea Eclipse este supravegheată de Fundația Eclipse, o organizație independentă, nonprofit, compusă din peste 100 de companii care susțin Eclipse. Eclipse Modeling Project este elementul de bază pentru tehnologiile de dezvoltare a modelelor bazate pe Eclipse. Se bazează pe EMF, care oferă framework-ul de bază pentru modelare. Alte sub-proiecte de modelare sunt construite pe EMF, oferind diverse capabilități.

Eclipse Modeling Framework (EMF) este un framework de modelare care exploatează caracteristicile oferite de Eclipse (vezi secțiunea 2.4.3) [Dave2009] [Bettini2016] [Eric2009]. EMF utilizează date structurate pentru a reprezenta modele, peste care alte instrumente pot construi modele specifice. Meta-metamodelul EMF se integrează foarte bine în Eclipse, dar poate fi utilizat și fără Eclipse. EMF [OMG2011] [OMG2015A] [OMG2002] se bazează pe metamodelul MOF [OMG_MOF] și a devenit în zilele noastre o tehnologie standard pentru construirea modelelor și limbajelor de modelare.

O parte din conținutul acestei secțiuni a fost publicat de către autorul acestei teze în [Craciunean2019A].

6 Două exemple relevante

Modelele, în general, se construiesc cu scopul de a modela diferite aspect ale comportamentului unui sistem și prin urmare instrumentele de modelare trebuie să fie adecvate pentru a surprinde, eficient, aceste aspecte. Pentru ca procesul de modelare să fie eficient este necesar ca același limbaj de modelare să poată fi utilizat, atât în faza informală cât și în cea formală, adică executabilă. Dacă instrumentul de modelare nu este adecvat domeniului de modelare, de cele mai multe ori, modelul informal nu poate fi exprimat, pe înțelesul specialiștilor din domeniul de modelare, în limbajul de modelare. De aici necesitatea de a construi noi instrumente de modelare adecvate domeniului de modelare în cauză.

O abordare frecvent întâlnită, în practica de modelare, este aceea de a modela anumite aspect ale comportamentului unui sistem prin procese. Un proces conține sarcinile care trebuie executate și ordinea în care acestea urmează să fie realizate. Astfel, putem considera un proces ca o procedură specifică unui anumit tip de cazuri. În esență, un proces este construit din sarcini, subprocese și condiții. Fiecare subproces este din nou compus din sarcini, condiții și, eventual, subprocese. O

sarcină este o unitate logică a muncii indivizibile și, prin urmare, este întotdeauna realizată în totalitate sau deloc. Dacă o operațiune din corpul unei sarcini nu se execută corect în timpul executării unei sarcini, atunci starea de la începutul sarcinii va fi restaurată. În această secțiune noi prezentăm o posibilă implementare a două instrumente de modelare, ambele adecvate pentru specificarea proceselor, dar cu grade de specificitate diferite.

Am folosit limbajul oferit de teoria categoriilor pentru a formaliza două exemple de metodă de modelare și apoi am implementat aceste exemple folosind ca suport limbajul MM-DSL cu translatare pe platforma ADOxx așa cum am văzut în secțiunea 5. Am ales pentru a implementa o metodă de modelare cu un limbaj bine cunoscut: Petri Nets și un limbaj specific pentru modelarea proceselor de fabricație. Cele două exemple de metode de modelare implementate sunt prezentate în subsecțiunile 6.2. și, respectiv, 6.3.

Un model Petri Nets este caracterizat de o dimensiune statică și o dimensiune comportamentală. Din punct de vedete sintactic dimensiunea statică este un graf cu două tipuri de noduri, unele care reprezintă locuri și altele care reprezintă tranziții. Arcele dintre aceste noduri direcționează cazurile.

Evident că nu orice graf care are nodurile de cele două tipuri, locuri și tranziții, este un model Petri Nets. Va trebui, prin urmare, să adăugam o serie de constrângeri acestui tip de graf cum ar fi faptul că graful este bipartit, conex și ponderat. După cum am văzut în secțiunea 5, sintaxa dimensiunii statice a unui model poate fi specificată printr-o schiță categorică în care constrângerile sunt impuse prin mecanisme categorice.

Semantica dimensiunii statice a unui model Petri Nets este dată de structura graf a rețelei, de poziția jetoanelor pe locurile rețelei care în implementarea noastră vor fi reprezentate prin valori ale unor atribute atașate acestui tip de noduri și prin ponderile arcelor care în cazul implementării noastre vor fi reprezentate prin valorile unor atribute asociate arcelor.

Dimensiunea dinamică a unui model Petri Nets este dată de tranziții care modifică distribuția locală a jetoanelor și de condițiile necesare pentru declanșarea tranzacțiilor. Declanșarea unei tranziții se face dacă sunt îndeplinite precondițiile și postcondițiile și afectează numai locurile din vecinătatea tranziției. În abordarea noastră, am modelat dimensiunea comportamentală a unui model, la nivel de metamodel, prin reguli comportamentale compuse din transformări graf și acțiuni comportamentale.

Digitalizarea sistemelor de fabricație actuale, care sunt printre cele mai complexe și heterogene sisteme, reprezintă o provocare majoră pentru MDD. Condițiile de concurență din zilele noastre impun, imperative, creșterea eficienței, care poate fi realizată doar prin reconfigurare și adaptare rapidă la cerințele externe. Modelarea este principalul factor în realizarea acestor obiective și, prin urmare, modelarea este principalul motor de creștere a eficienței. Dar, pentru ca modelarea să fie, cu adevărat, un factor de creștere a eficienței, este necesar ca și procesul de modelare să fie eficient la rândul său. Eficiența procesului de modelare este dependentă, în mare parte, de facilitățile oferite de instrumentul de modelare utilizat. Datorită faptului că în activitatea de modelare a proceselor de fabricație sunt implicate mai multe specializări, este nevoie ca limbajul de modelare utilizat să ofere suport în toate fazele de modelare, începând cu faza de specificare a modelului informal și până în faza de generare a modelului executabil.

Datorită diversității mari a proceselor de fabricație, de multe ori, limbajele de modelare existente sunt prea generale și nu oferă suport corespunzător pentru specificarea tuturor conceptelor din domeniul de modelare de la faza informală până la faza executabilă.

Pentru a satisface cerințele de modelare de la faza formală până la faza de execuție, limbajul de modelare trebuie să ofere o lizibilitate accesibilă tuturor participanților la procesul de modelare și în același timp mecanisme de specificare precisă a modelelor. Pentru îndeplinirea acestor două cerințe, limbajul de modelare trebuie să aibă un grad ridicat de specificitate. De exemplu Petri Nets, permit o specificare precisă a modelelor deoarece se bazează pe un formalism matematic puternic, dar sunt prea abstracte și prin urmare au o lizibilitate limitată. Deși Perti Nets, oferă mecanisme de specificare exactă a conceptelor și a relațiilor dintre acestea, aplicabilitatea acestora este limitată la o clasă de modele legate de dinamica proceselor [Karagiannis2016].

În cazul proceselor de fabricație, semantica statică reprezentată de structura graf a modelelor și de valorile atributelor este la fel de importantă ca și semantica comportamentală reprezentată de regulile comportamentale. În aceste condiții, soluția este implementarea unui nou instrument de modelare, înzestrat cu un limbaj de modelare specific domeniului, care să ofere astfel de facilități.

Prima fază a construirii unui limbaj de modelare specific unui domeniu de modelare este modelarea conceptuala. În această fază se stabilesc conceptele atomice ale domeniului de modelare și care sunt caracteristicile implicate în modelele pe care dorim să le specificăm. Tot în aceasta fază se stabilește cum trebuie să arate modelele pe care dorim să le reprezentăm și care sunt operațiile la care urmează a fi supuse aceste modele. Prin urmare, această fază, presupune structurarea tuturor informațiilor din domeniul de modelare în termeni de concept.

În faza următoare va trebui stabilit, în funcție de cerințele utilizatorilor finali, cum trebuie să arate instrumentul de modelare pe care vrem să-l construim. În această fază avem un concept reper, pentru specificarea componentelor instrumentului de modelare și anume conceptul de modeling method. Va trebui, prin urmare să specificăm componentele conceptului de modeling method, adică limbajul de modelare, procedura de modelare și mecanismele și algoritmii. Desigur că instrumentul de metamodelare ales are și el un rol esențial în această fază. Limbajul de metamodelare MM-DSL are rolul de a oferii flexibilitate în alegerea instrumentului de metamodelare.

În general, se acceptă faptul că limbajele diagramatice sunt cele mai adecvate pentru specificarea modelelor dintr-un anumit domeniu deoarece sunt intuitive și suficient de expresive pentru a oferii suport în toate fazele de modelare. Simbolurile sugestive atribuite conceptelor atomice oferă o imagine intuitiva a modelelor, iar modul de conectare multiplă a acestor componente atomice în modele oferă un grad de expresivitate suficient pentru reprezentarea informală și formală a modelelor.

În abordarea noastră, specificarea unui model presupune specificarea celor două dimensiuni și anume dimensiunea statică și dimensiunea comportamentală. Ambele dimensiuni trebuie specificate printr-o sintaxă, iar această sintaxă trebuie să primească înțelesul dorit prin maparea la un domeniu semantic. Evident că este de preferat ca dimensiunea comportamentală să fie specificată și implementată, în totalitate dacă este posibil, la nivel de metamodel. De asemenea semantica dimensiunii statice este de preferat să fie specificată formal, la nivel de metamodel, iar limbajul să fie oferit utilizatorilor finali doar cu sintaxa necesară pentru specificarea dimensiunii statice a modelelor și cu o semantică specificată textual în limbajul natural. Abordarea noastră urmărește atingerea acestor obiective.

Algoritmii implementați la nivel de metamodel vor fi integrați în componenta algoritmi și mecanisme a conceptului de modeling method.

În această subsecțiune am prezentat pe scurt procesul de conceptualizare a unei metode de modelare pe care am numit-o Modeling method for a digital manufacturing planner (MM-DiMaP), și pe care am implementat-o într-un instrument de modelare pe care l-am numit Digital manufacturing planning tool (DiMaP). Instrumentul de modelare DiMaP este înzestrat cu un limbaj de modelare diagramatic pe care l-am numit DiMaP-DSML și care este un limbaj formal.

Pentru implementarea instrumentului de modelare am utilizat limbajul MM-DSL. Exemplul din aceasta subsecțiune este inspirat de procesele de fabricație ale unei companii care produce lasere medicale, pentru care eu am dezvoltat software peste 10 de ani. Conținutul acestei subsecțiuni a fost parțial inclus de către autorul acestei teze în lucrările [Craciunean2019C] [wp32020].

7 Concluzii și contribuții

Esența demersului nostru în această teză este legată de identificarea și definirea mecanismelor teoretice și practice, adecvate procesului de abstractizare a unui domeniu specific și de specificare optimă a soluțiilor din acest domeniu. Acest demers este motivat de necesitatea imperativă a dezvoltării de instrumente de modelare specifice domeniului în contrast cu preocupările timide din acest domeniu și are ca obiectiv identificarea mecanismelor implicate în procesul de dezvoltare metodologică a acestor instrumente.

Dezvoltarea limbajelor vizuale adecvate pentru a specifica modele dintr-un anumit domeniu, numite DSML, este o parte integrantă a procesului de modelare software. Proiectarea unui nou DSML implică deseori interpretarea nodurilor și muchiilor unui graf ca obiecte specifice modelului și apoi impunerea constrângerilor specifice domeniului și atribuirea simbolurilor vizuale sugestive pentru a reprezenta conceptele modelate. Preocuparea noastră este, în primul rând, să găsim mecanismele care permit implementarea generică, la nivel de metamodel, a constrângerilor sintactice și a comportamentului dinamic al modelelor.

Această lucrare evidențiază compatibilitatea dintre mecanismele categorice și mecanismele de modelare software și contribuie la reducerea decalajului dintre natura abstractă a teoriei categoriilor și modelare. Schița categorică reprezintă un metamodel schematic al unei clase de modele grafice și oferă un cadru matematic puternic pentru formalizarea sintaxei acestor modele la nivel de metamodel. Constrângerile în definirea schiței categorice sunt impuse de construcțiile universale din teoria categoriilor sau de semnături diagramatice ale predicatelor. Construcțiile universale din teoria categoriilor ne oferă un pachet de rezultate universal valabile care pot fi implementate independent de orice model concret la cel mai înalt grad de abstractizare. Semnătura diagramatică a predicatelor poate fi, de asemenea, implementată cu un grad ridicat de abstractizare.

Pentru a defini comportamentul unui model (secțiunea 3.4.), am introdus noțiunea de transformare comportamentală care transformă structura modelului și atributele, în limita permisă de constrângerile asociate. Aceste reguli de transformare acționează local, ceea ce înseamnă că modifică mai multe substări locale în funcție de anumite condiții prealabile. O regulă comportamentală este o descriere

generală a schimbărilor locale care pot apărea într-un model și determină comportamentul acestuia în funcție de evoluția sistemului modelat. În principiu, aplicarea unei reguli comportamentale constă în înlocuirea unei părți a modelului cu o altă parte compatibilă cu definiția modelului, precum și calcularea sau recalcularea valorilor unor atribute, în conformitate cu constrângerile impuse de metamodel.

Pentru a defini generic noțiunea de regulă comportamentală, am introdus noțiunile de semnătură diagramatică a unei reguli comportamentale și de semnătură diagramatică a unei acțiuni care specifică modul de transformare a elementelor graf la un nivel generic. De asemenea, în același scop, am introdus noțiunea de trei-diagramă, care este un mecanism pentru asocierea componentelor grafice formale, reprezentate prin forme graf, la componentele reale din graful schiței categorice.

Am asamblat apoi toate aceste elemente în conceptul de schiță comportamentală categorică, care definește comportamentul modelelor la nivel de metamodel. Pe baza schiței comportamentale am definit apoi conceptul de model al unei schițe comportamentale pe care l-am numit model comportamental (secțiunea 3.5.).

Evident, un model de sistem implică combinarea modelului static cu modelul comportamental, în același format diagramatic, pentru a permite modelatorului să testeze și să analizeze un sistem real. În acest fel, am ajuns să definim un metamodel prin cuplarea metamodelului static cu cel comportamental. De aici rezultă că un model, la rândul său, se obține prin cuplarea celor două dimensiuni specificate de metamodel, și anume dimensiunea statică și dimensiunea comportamentală, prima dimensiune specifică stările modelului iar a doua tranziția de la o stare la alta.

În secțiunea 3.6. am introdus noțiunea de metamodel și noțiunea de ierarhie de modelare pe mai multe niveluri. O ierarhie de modelare pe mai multe niveluri poate fi reprezentată de un arbore, care are rădăcina reprezentată de cel mai abstract metamodel, nodurile interioare sunt metamodele, iar frunzele sunt modele. Prin urmare, fiecare model este caracterizat direct sau indirect de toate metamodelele care se află pe un lanț de tipizare. O ierarhie de modelare pe mai multe niveluri specifică o ierarhie care conține o familie de modele legate de diferite niveluri de abstractizare, care au o rădăcină comună și două dimensiuni în care pot fi localizate, o dimensiune statică și o dimensiune comportamentală. Dimensiunea statică este reprezentată la fiecare nivel i al unui lanț de tipizare. Deoarece tipizarea este unică în fiecare ierarhie, fiecare nod, cu excepția rădăcinii, are exact un nod părinte în ierarhie. Pentru ca fiecare element să aibă un tip și procesul de modelare să fie finit, nodul rădăcină este o colecție de elemente cu tipuri auto-definite.

Tot în secțiunea 3.6. am introdus noțiunea de trei-diagramă indirectă, care este o nouă trei-diagramă obținută prin compunerea unui trei-diagrame cu o potrivire p. Modelarea pe mai multe niveluri creează astfel oportunitatea implementării comportamentului la un nivel ridicat de abstractizare, prin reguli comportamentale generice definite la nivelurile superioare și utilizarea acestora la nivelul modelelor concrete.

Putem observa că, în abordarea noastră, semnăturile comportamentale pot fi introduse la diferite niveluri de abstractizare împreună cu trei-diagramele corespunzătoare și sunt apoi mapate la trei-diagrame indirecte la reguli comportamentale care se propagă până la cel mai puțin abstract nivel, adică la nivelul de modelul concret. Prin urmare, regulile comportamentale pot fi implementate generic, la un nivel ridicat de abstractizare și apoi aplicate modelului concret.

Transformările comportamentale definite în această lucrare reprezintă o bază solidă pentru implementarea unui motor generic de execuție a modelului care permite simularea modelelor. În abordarea noastră, tranziția de la o stare la alta se face prin transformări comportamentale. Desigur, anumite transformări comportamentale pot fi realizate în paralel în procesul de evoluție al modelului. Plecând de la un rezultat teoretic care spune că orice două transformări comportamentale pot fi compuse într-o singură transformare, numită transformare E-concurentă care cumulează aplicarea a două transformări secvențiale într-o singură transformare [Ehrig2006] [Ehrig2015], am organizat procesul de execuție a unui model într-o categorie pe care am numit-o categoria instanțelor și transformărilor comportamentale care are ca obiecte instanțe ale modelului static și ca săgeți transformări comportamentale. Transformările comportamentale dintre două instanțe ale modelului sunt astfel reprezentate de macro-tranziții declanșate de starea reprezentată de instanța inițială.

Plecând de la ideea că simularea începe de la o stare inițială, am definit categoria de simulare a proceselor (PSC) ca o subcategorie a categoriei instanțelor și regulilor comportamentale (CIBR). Categoria PSC permite studiul comportamentului unui sistem pe baza urmelor modelului în procesul de simulare, care se identifică cu căile din categoria PSC care au ca obiect de pornire instanța inițială.

Tot în această secțiune se face și o primă evaluare a conceptelor introduse prin scurte exemple relevante și un studiu de caz.

Procesul natural de formalizare a modelelor din secțiunea 3 demonstrează adecvarea mecanismelor categorice la specificarea conceptelor implicate în procesul de metamodelare precum și a relațiilor inter-structurale dintre acestea. De asemenea limbajul categoric permite și specificarea fidelă a spațiului de simulare și execuție al unui model. Categoria CIBR are ca obiecte stările unui model iar ca săgeți tranzițiile modelului de la o stare la alta și prin urmare reprezintă, drumurile admisibile de evoluție ale modelului. Componentele categoriei CIBR sunt obiecte care pot fi înzestrate cu proprietăți care pot colecționa dinamic informații referitoare la evoluția sistemului precum, durata, frecvența, costul și probabilitatea de execuție a unei tranziții. Aceste informații, dacă sunt stocate într-o bază de date, pot fi utilizate pentru autoreglarea sau optimizarea sistemului cu sprijinul unor instrumente inteligente de analiză și de învățare adaptate.

Contribuția esențială originală a secțiunii 4 este legată de specificarea categorică a conceptului de metodă de modelare. Structura unui model și comportamentul acestuia sunt dimensiuni esențiale ale unui model și fiecare dintre ele are propria sintaxă și semantică. Deși există o legătură strânsă între semantica structurală a unui model și semantica comportamentală a acestuia, acestea sunt totuși diferite.

Condiția esențială pentru ca un limbaj să permită definirea unei semantici precise este aceea ca acesta să aibă o sintaxă riguroasă, care specifică în mod clar și fără echivoc construcțiile sintactice permise. Schița categorică este un concept ideal pentru a specifica riguros sintaxa unui limbaj diagramatic. Schița categorică se asociază foarte bine cu transformările graf și formează, în opinia noastră, un mecanism riguros, expresiv și eficient pentru specificarea comportamentului unui model.

Semantica unei structuri de model implică maparea atributelor la setul lor de valori, și interpretarea structurii grafice a modelului, adică maparea construcțiilor sintactice la structuri bine definite precum secvențiale, repetitive, recursive, join, fork etc. și definirea de operații pe atribute în contextul acestor structuri.

Semantica comportamentală a unui model implică maparea modelului la o mulțime de predicate, care verifică starea modelului, la o mulțime de acțiuni (funcții) care recalculează valorile atributelor și la o mulțime de transformări graf care transformă structura grafică a modelului prin repoziționarea componentelor, adăugarea sau eliminarea de componente.

Această abordare oferă o resursă importantă de componente generice, cum ar fi: grafuri, morfisme de grafuri, categorii, functori, diagrame, transformări naturale, conuri, coconuri, limite, colimite etc., care devin concepte fundamentale în modelarea diagramatică. Toate aceste construcții au un caracter generic și pot fi implementate ca tipuri de date, mecanisme sau algoritmi generice, independent de orice model diagramatic care le folosește.

Prezentarea este însoțită de specificarea și implementarea unui exemplu de concept de metodă de modelare.

Contribuția noastră esențială în secțiunea 5 este implementarea parțială a limbajului MM-DSL. Limbajul conține un subset de instrucțiuni descriptive care permit definirea componentelor unei metode de modelare și un subset de instrucțiuni de calcul și control al fluxului care permit specificarea algoritmilor. Partea descriptivă a limbajului a fost implementată în lucrarea [Visic2016], iar în această lucrare am finalizat implementarea limbajului cu partea de calcul și control, adică construcțiile de limbaj care permit specificarea algoritmilor.

La această contribuție esențială putem adăuga specificarea metodei de modelare SMM în limbajul MM-DSL pentru a demonstra facilitățile esențiale oferite de limbajul MM-DSL pentru implementarea metodelor de modelare. Limbajul MM-DSL permite specificarea, independentă de platforma de metamodelare, a unei metode de modelare. Această implementare demonstrează o foarte bună compatibilitate a mecanismelor categorice cu sintaxa și semantica limbajului MM-DSL.

Formalizarea conceptului de metodă de modelare, bazată pe teoria categoriilor, introdusă în secțiunea 4 indică faptul că limbajul MM-DSL oferă toate facilitățile pentru specificarea conceptelor de bază ale unei metode de modelare.

Contribuția noastră principală din secțiunea 6 este legată de specificarea și implementarea a două exemple relevante de metode de modelare, și anume unul referitor la construirea unui instrument de modelare bazat pe limbajul Petri Nets și celălalt bazat pe un DSML pentru specificarea unui anumit tip de proces de fabricație. Prin urmare, am exemplificat facilitățile oferite de teoria categoriilor împreună cu limbajul MM-DSL pentru formalizarea și implementarea conceptului de metodă de modelare.

Procesul de conceptualizare, formalizare, specificare și implementare a metodei de modelare MM-DiMaP în contextul mecanismelor categorice demonstrează fezabilitatea și eficiența abordării noastre metodologice. La sfârșitul acestui proces am obținut instrumentul de modelare DiMaP, care a fost specificat in limbajul MM-DSL și apoi translatat și testat pe platforma de metamodelare ADOxx. Cu toate că acest instrument este un prototip înzestrat doar cu funcționalități minimale el demonstrează, totuși, principalele avantajele oferite de compatibilitatea unui instrument de modelare adecvat cu un domeniu restrâns de modelare.

Limbajul DiMaP-DSML, pe baza unui număr redus de atomi lexicali, oferă suficiente facilități pentru specificarea formală a celulelor de fabricație precum și facilități de analiză și optimizare a celulelor de fabricație. Încărcătura semantică bogată a limbajului derivă, pe de o parte, din complexitatea

nelimitată a conexiunilor între atomii lexicali, iar pe de altă parte din includerea semanticii comportamentale la nivel de metamodel. Prin urmare, deși limbajul DiMaP-DSML este formal, acesta oferă facilități de specificare a unui model din domeniul de modelare similare cu facilitățile oferite de limbajele informale.

8 Referințe

1. [Aalst2011] Wil M.P. van der Aalst: Process Mining Discovery, Conformance and Enhancement of Business Processes, Springer-Verlag Berlin Heidelberg 2011.
2. [Aalst1998] Wil M.P. van der Aalst, Formalization and verification of event-driven process chains (Computing science reports; Vol. 9801). Eindhoven: Technische Universiteit Eindhoven 1998.
3. [Aalst2004] Wil M.P. van der Aalst and K.M. van Hee: Workflow Management: Models, Methods, and Systems. MIT press, Cambridge, MA, 2004.
4. [Aho2006] Aho, A.V., Lam, M.S., Sethi, R., Ullman J.D.: Compilers: Principles, Techniques, and Tools, 2nd edn. Pearson Education, Inc, India (2006)
5. [Aldini2010] Alessandro Aldini, Marco Bernardo, Flavio Corradini, A Process Algebraic Approach to Software Architecture Design, Springer-Verlag London Limited 2010
6. [Asperti1991] Andrea Asperti, Giuseppe Longo, Categories Types And Structures, An Introduction to Category Theory for the working computer scientist, Foundations Of Computing Series M.I.T. Press, 1991.
7. [Andy2005] Andy Ju An Wang, Kai Qian, Component-Oriented Programming ISBN: 978-0-471-64446-0, 2005. Copyright 2005 by John Wiley & Sons, Inc.
8. [ANTLR2018] ANTLR.[Online]. Available: http://www.antlr.org/. December 18, 2018. As of 4.7.2.
9. [Atkinson1997] Colin Atkinson. Meta-modelling for distributed object environments. In Enterprise Distributed Object Computing Workshop [1997]. EDOC'97. Proceedings. First International, pages 90{101. IEEE, 1997.
10. [Atkinson2000] Colin Atkinson and Thomas Khuhne. Meta-level independent modelling. In International Workshop on Model Engineering at 14th European Conference on Object-Oriented Programming, pages 12{16. Citeseer, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/ download?doi=10.1.1.26.3964&rep=rep1&type=pdf. 45
11. [Atkinson2001] Colin Atkinson and Thomas Khuhne. The essence of multi-level metamodeling. UMLThe Unified Modeling Language. Modeling Languages,Concepts, and Tools, pages 19{33, 2001. URL http://www.springerlink.com/index/ccbgph1thqmx9myn.pdf. 43, 44, 45, 118,220, 236
12. [Atkinson2002]Colin Atkinson and Thomas Khuhne. Rearchitecting the UML infrastructure.ACM Transactions on Modeling and Computer Simulation,12(4):290{321, October 2002. ISSN 10493301. doi: 10.1145/643120.643123. URL http://portal.acm.org/citation.cfm?id=643120. 643123. 34, 41, 45, 236, 242
13. [Atkinson2003]Colin Atkinson and Thomas Khuhne. Model-driven development:A metamodeling foundation. IEEE Software, 20(5):36{41, September 2003. ISSN 0740-7459. doi: 10.1109/MS.2003.1231149. URL http://www.computer.org/portal/web/csdl/doi/10.1109/MS.2003.1231149. 34, 45, 220, 236

14. [Atkinson2008] Colin Atkinson and Thomas Khuhne. Reducing accidental complexity in domain models. Software and Systems Modeling, 7(3):345{359, 2008. ISSN 1619-1366. URL http://dx.doi.org/10.1007/s10270-007-0061-0. 190
15. [Atkinson2014] Colin Atkinson, Ralph Gerbig and Thomas Kuhne. Comparing Multi-Level Modeling Approaches, Proceedings of the 1st Workshop on Multi-Level Modelling co-located with the 17th ACM/IEEE International Conference MODELS 2014, CEUR, Vol-1286, 2014, At Valencia, Spain, Volume: 1286
16. [Atkinson2014A] Colin Atkinson, Georg Grossmann, Thomas Kühne, Juan de Lara (Eds.) ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems September 28 – October 3, 2014 , Valencia (Spain), MULTI 2014 – Multi-nivel Modelling Workshop Proceedings
17. [Atkinson2016] Atkinson C., Gerbig R. Flexible Deep Modeling with Melanee. In: Betz S., Reimer U. (eds.) Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband. Modellierung 2016 Vol. 255. Gesellschaft für Informatik, Bonn, pp. 117–122
18. [Atkinson2017] Colin Atkinson and Thomas Kuhne. On Evaluating Multi-Level Modeling, Published in MODELS 2017
19. [AToM3] http://moncs.cs.mcgill.ca/MSDL/research/projects/AToM3.
20. [ADOxx] ADOxx, https://www.adoxx.org
21. [Bettini2016] Bettini Lorenzo, Implementing Domain-Specific Languages with Xtext and Xtend, Published by Packt Publishing Ltd, Second Edition:August 2016.
22. [Bergstra1985] Bergstra & Klop's Algebra of communicating processes with abstraction, Theoretical Computer Science Volume 37, Pages 77-121, 1985
23. [Barr2012] Michael Barr And Charles Wells, Category Theory For Computing Science-Reprints in Theory and Applications of Categories, No. 22, 2012.
24. [Barr2002] Michael Barr Charles Wells. Toposes, Triples and Theories November 2002.
25. [Bak2012] C. Bak and D. Plump, 'Rooted graph programs', Proceedings of the 7th International Workshop on Graph Based Tools (GraBaTs 2012), Electronic Communications of the EASST, vol. 54, 2012. DOI: 10.14279/tuj.eceasst.54.780.
26. [Bork2014] D. Bork, H.G. Fill, Formal Aspects of Enterprise Modeling Methods: A Comparison Framework, In: System Sciences (HICSS), 47th Hawaii International Conference, pp. 3400-3409 (2014)
27. [Bork2018] Dominik Bork and Robert Andrei Buchmann and Dimitris Karagiannis and Moonkun Lee and Elena-Teodora Miron, An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem, Communications of the Association for Information Systems, Vol. 44, pp. 673-697,2018, Copyright by AIS. DOI: https://doi.org/10.17705/1CAIS.04432
28. [Bork2019] D. Bork, R.A. Buchman, D. Karagiannis, M. Lee, E.T. Miron, An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem, Communications of the Association for Information Systems, forthcoming, http://eprints.cs.univie.ac.at/5462/1/CAIS-OMiLAB-final-withFront.pdf (2019)
29. [Bork2020] Dominik Bork, Dimitris Karagiannis, Benedikt Pittl, A survey of modeling language specification techniques, Information Systems 87 (2020) 101425, journal homepage: www.elsevier.com/locate/is
30. [Borger2012] E. Borger: Approaches to Modeling Business Processes. A Critical Analysis of BPMN, Workow Patterns and YAWL. in: J. SOFTWARE AND SYSTEMS MODELING, Volume 11, Issue 3 (2012), page 305-318, DOI: 10.1007/s10270-011-0214-z. ISSN: 1619-1366 (print version), ISSN: 1619-1374 (electronic version)
31. [Calude2001] Calude Ch. S., Păun Gh., Computing with Cells and Atoms an introduction to quantum, DNA and membrane computing, Taylor & Francis, (2001).

32. [Campbell2018] G. Campbell, ‘Algebraic graph transformation: A crash course’, Department of Computer Science, University of York, UK, Tech. Rep., 2018. [Online]. Available: https://cdn.gjcampbell.co.uk/2018/Graph-Transformation.pdf.
33. [Campbell2019] G. Campbell, B. Courtehoute and D. Plump, ‘Linear-time graph algorithms in GP2’, Department of Computer Science, University of York, UK, Submitted for publication, 2019. [Online]. Available: https://cdn.gjcampbell.co.uk/2019/Linear-Time-GP2-Preprint.pdf.
34. [Cesar2006] Cesar Gonzalez-Perez and Brian Henderson-Sellers. A powertypebased metamodelling framework. Software and Systems Modeling, 5 (1):72{90, 2006. ISSN 1619-1366. URL http://dx.doi.org/10.1007/ s10270-005-0099-9. 33, 45
35. [Christos2008] Christos G. Cassandras, Stéphane Lafortune- Introduction to Discrete Event Systems Second Edition- 2008 Springer Science+Business Media, LLC, ISBN-13: 978-0-387-33332-8
36. [Chen2014] Qingfeng Chen, Baoshan Chen, Chengqi Zhang-Intelligent Strategies for Pathway Mining Model and Pattern Identification, Springer International Publishing Switzerland 2014.
37. [Clark2008] Alexander Clark Francois Coste, Laurent Miclet (Eds.)-Grammatical Inference: Algorithms and Applications 9th International Colloquium, ICGI 2008 Saint-Malo, France, September 22-24, 2008 Proceedings- Springer-Verlag Berlin Heidelberg 2008.
38. [Clark2014] Tony Clark, Cesar Gonzalez-Perez2, Brian Henderson-Sellers. A Foundation for Multi-Level Modelling, ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems September 28 – October 3, 2014 , Valencia (Spain)
39. [Craciunean2018] Daniel-Cristian Crăciunean, Dimitris Karagiannis, Categorical modeling method of intelligent WorkFlow, Lecture Notes in Computer Science, 6th International Conference on Mining Intelligence and Knowledge Exploration, MIKE 2018, VOL 11308, ISSN 3029743, DOI: 10.1007/978-3-030-05918-7_11, 2018.
40. [Craciunean2019A] Daniel-Cristian Crăciunean, Daniel Volovici, MM-DSL, support for implementing modelling tools for manufacturing processes, MATEC Web of Conferences, VOL. 290, ISSN 2261-236X, 2019.
41. [Craciunean2019B] Daniel-Cristian Crăciunean, Dimitris Karagiannis, A categorical model of process co-simulation, INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, VOL 10, NR 1, ISSN 2158-107X, DOI: 10.14569/IJACSA.2019.0100355, 2019.
42. [Craciunean2019C] Daniel-Cristian Crăciunean, Categorical Grammars for Processes Modeling, INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, VOL 10, NR 3, ISSN 2158-107X, DOI: 10.14569/IJACSA.2019.0100105, 2019.
43. [Craciunean2019D] Daniel-Cristian Crăciunean, Categorical Modeling Method, proof of concept for the Petri Net language, MODELSWARD 2019 - Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, ISSN 978-989758358-2, 2019.
44. [Craciunean2020] Daniel-Cristian Crăciunean, “Implementing the Behavioral Semantics of Diagrammatic Languages by Co-simulation” International Journal of Advanced Computer Science and Applications(IJACSA), 11(9), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110964
45. [Dave2009] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, Ed Merks, EMF: Eclipse Modeling Framework, Addison-Wesley, 2009.
46. [Diskin2008] Zinovy Diskin, Uwe Wolter, A Diagrammatic Logic for Object-Oriented Visual Modeling, Electronic Notes in Theoretical Computer Science Volume 203, Issue 6, 21 November 2008, Pages 19-41

47. [Diskin2012] Zinovy Diskin, Tom Maibaum- Category Theory and Model-Driven Engineering: From Formal Semantics to Design Patterns and Beyond, ACCAT 2012
48. [Diskin2018] Diskin Z., König H., Lawford M., 2018. Multiple Model Synchronization with Multiary Delta Lenses. In: Russo A., Schürr A. (eds) Fundamental Approaches to Software Engineering. FASE 2018. Lecture Notes in Computer Science, vol 10802. Springer, Cham
49. [Dodds2008] M. Dodds, 'Graph transformation and pointer structures', PhD thesis, Department of Computer Science, University of York, UK, 2008. [Online]. Available: https://www.cs.york.ac.uk/plasma/publications/pdf/DoddsThesis.08.pdf.
50. [Efendioglu2017] Nesat Efendioglu, Robert Woitsch, Wilfrid Utz and Damiano Falcioni, A Product-Service System Proposal for Agile Modelling Method Engineering on Demand: ADOxx.org, Alexander Rossmann, Alfred Zimmermann (eds.): Digital Enterprise Computing 2017, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 199
51. [Egon2012] Egon Börger,Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL, Software & Systems Modeling ISSN 1619-1366 Volume 11 Number 3 Softw Syst Model (2012) 11:305-318 DOI 10.1007/s10270-011-0214-z, Springer.
52. [Eriksson2013] Eriksson, O., Henderson-Sellers, B., A° gerfalk, P.J.: Ontological and linguistic metamodelling revisited: A language use approach. Information & Software Technology 55(12),2099–2124 (2013)
53. [Eric2009] Eric Clayberg Dan Rubel, Eclipse Plug-ins, Third Edition, Addison-Wesley, 2009.
54. [Ehrig2006] H. Ehrig, K. Ehrig, U. Prange and G. Taentzer, Fundamentals of algebraic graph transformation, ser. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006. DOI: 10.1007/3-540-31188-2.
55. [Ehrig2015] Hartmut Ehrig, Claudia Ermel, Ulrike Golas, Frank Hermann, Graph and Model Transformation General Framework and Applications, Springer-Verlag Berlin Heidelberg 2015
56. [Fill2012] Fill, H-G., Redmond, T. and Karagiannis, D.: FDMM: A Formalism for Describing ADOxx Meta Models and Models. ICEIS, 2012.
57. [Fill2013] Fill, H., Karagiannis, D.: On the conceptualization of modelling methods using the ADOxx meta modelling platform. Enterp. Modell. Inf. Syst. Arch.—Int. J. (2013)
58. [Fowler2010] M. Fowler and R. Parsons, Domain Specific Languages, 1st ed. Addison-Wesley Longman, Amsterdam, 2010.
59. [Glabbeek2001] RJ. van Glabbeek - The Linear Time - Branching Time Spectrum I., The Semantics of Concrete, Sequential Processes- Handbook of Process Algebra-Elsevier Science (2001)
60. [Götzinger2016] David Götzinger, Elena-Teodora Miron and Franz Staffel, OMiLAB: An Open Collaborative Environment for Modeling Method Engineering, in vol. Dimitris Karagiannis, Heinrich C. Mayr John Mylopoulos Editors, Domain-Specific Conceptual Modeling Concepts, Methods and Tools, Springer International Publishing Switzerland 2016
61. [github] https://github.com/metaborg/spoofax
62. [Gomes2016] C. Gomes, C. Thule, D. Broman, P.G. Larsen, H. Vangheluwe - Co-simulation: State of the art, - ACM Computing Surveys, Vol. 1, No. 1, Article 1. Publication date: January (2016).
63. [Hofstede2010] H. M. Hofstede, A., van der Aalst, W., Adams, M., Russell, N.(eds.): Modern Business Process Automation. Springer, Berlin (2010)
64. [Hoare1985] C.A.R. Hoare - Communicating Sequential Processes, Prentice Hall 1985
65. [Henderson2013] Henderson-Sellers, B., Clark, T., Gonzalez-Perez, C.: On the search for a level-agnostic modeling language. In: Proceedings of the 25th International Conference on

Advanced Information Systems Engineering. pp. 240–255. CAiSE'13, Springer-Verlag, Berlin, Heidelberg (2013)
66. [Henderson2013] Henderson-Sellers, B., Eriksson, O., Gonzalez-Perez, C., A° gerfalk, P.J.: Ptolemaic Metamodelling?: The Need for a Paradigm Shift, chap. 4, pp. 90–146. IGI Global (2013)
67. [Henderson2011] Henderson-Sellers B., Bridging metamodels and ontologies in software engineering. Journal of Systems and Software, 84(2):301{313, February 2011. ISSN 01641212. doi: 10.1016/j.jss.2010.10.025. URL http://portal.acm.org/citation.cfm?id=1922690.1922987. 33
68. [Henderson2007] Henderson-Sellers B., and Cesar Gonzalez-Perez The Rationale of Powertype-based Metamodelling to Underpin Software Development Methodologies, 2007
69. [Heckel2018] Reiko Heckel, Gabriele Taentzer (Eds.), Graph Transformation, Specifications, and Nets, Springer International Publishing AG, part of Springer Nature 2018
70. [Hrgovcic2013] Hrgovcic, Vedran; Karagiannis, Dimitris;Woitsch, Robert. Conceptual Modeling of the Organisational Aspects for Distributed Applications: The Semantic Lifting Approach, IEEE 37th Annual Computer Software and Applications Conference Workshops-2013.
71. [Hristakiev2018] I. Hristakiev, 'Confluence analysis for a graph programming language', PhD thesis, Department of Computer Science, University of York, UK, 2018. [Online]. Available: https://etheses.whiterose.ac.uk/20255/.
72. [Jaime2014] Jaime Gómez-Ramirez, A New Foundation for Representation in Cognitive and Brain Science Category Theory and the Hippocampus, Springer Science+Business Media Dordrecht 2014.
73. [Johnson1978] S. C. Johnson, "Yacc: Yet Another Compiler-Compiler," 1978.
74. [Junginger2000] Junginger S., Kuhn H., Strobl R., Karagiannis D. (2000) Ein Geschaftsprozessmanagement-Werkzeug der nachsten Generation – ADONIS: Konzeption und Anwendungen (German). In: Wirtschaftsinformatik 42(5),pp. 392–401
75. [jetbrains] jetbrains.com/mps
76. [Kivunja2017] Charles Kivunja, Ahmed Bawa Kuyini.(2017) Understanding and Applying Research Paradigms in Educational Contexts. International Journal of Higher Education Vol. 6, No. 5; 2017.
77. [Kindler2004] Kindler, E.: On the semantics of BPMNs: A Framework for Resolving the Vicious Circle. In: J. Desel and B. Pernici and M. Weske (Hrsg.), Business Process Management, 2nd International Conference, BPM 2004. volume 3080 of Lecture Notes in Computer Science. S. 82–97. Springer Verlag. 2004.
78. [Kindler2006] E. Kindler, V. Rubin, and R. Wagner. Component Tools: Integrating Petri nets with other formal methods. In S. Donatelli and P. S. Thiagarajan, editors, Application and Theory of Petri Nets 2006, 27th International Conference, volume 4024 of LNCS, pages 37-56. Springer, June 2006.
79. [Karagiannis1996] Dimitris Karagiannis, Junginger S., Strobl R.: Introduction to Business Process Management Systems Concepts. In: Scholz-Reiter B., Stickel E. (eds) Business Process Modelling. Springer, Berlin, Heidelberg, 1996
80. [Karagiannis2002] Dimitris Karagiannis; H. Kühn: Metamodelling Platforms. Invited paper in: Bauknecht, K.; Tjoa, A Min.; Quirchmayer, G. (eds.): Proceedings of the Third International Conference EC- Web 2002 - Dexa 2002, Aix-en-Provence, France, September 2-6, 2002, LNCS 2455, Springer-Verlag, Berlin, Heidelberg.
81. [Karagiannis2011] Dimitris Karagiannis N. Visic, Next Generation of Modelling Platforms, Perspectives in Business Informatics Research 10th International Conference, BIR 2011 Riga, Latvia, October 6-8, 2011 Proceedings

82. [Karagiannis2014] Dimitris Karagiannis, Evolution of Knowledge Management: From Expert Systems to Innovation 2.0, IAEA International Conference on Human Resource Development for Nuclear, Power Programs: Building and Sustaining Capacity, 12-16 May 2014
83. [Karagiannis2015] Dimitris Karagiannis, "Agile Modeling Method Engineering," in Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, ACM, 2015, pp.5-10.
84. [Karagiannis2016] Dimitris Karagiannis, Heinrich C. Mayr John Mylopoulos Editors, Domain-Specific Conceptual Modeling Concepts, Methods and Tools, Springer International Publishing Switzerland 2016.
85. [Karagiannis2016A] Karagiannis, D., Buchmann, A.: Linked open models: extending linked open data with conceptual model information. Inf. Syst. 56, 174–197 (2016)
86. [Karagiannis2016C] Dimitris Karagiannis, Robert Andrei Buchmann, Patrik Burzynski, Ulrich Reimer and Michael Walch, Fundamental Conceptual Modeling Languages in OMiLAB in vol. Dimitris Karagiannis, Heinrich C. Mayr John Mylopoulos Editors, Domain-Specific Conceptual Modeling Concepts, Methods and Tools, Springer International Publishing Switzerland 2016.
87. [Karagiannis2018] Dimitris Karagiannis, R.A. Buchmann, A Proposal for Deploying Hybrid Knowledge Bases: the ADOxx-to-GraphDB Interoperability Case, In: System Sciences (HICSS), 51st Hawaii International Conference, pp. 4055-4064 (2018)
88. [Karagiannis2019] Dimitris Karagiannis, Patrik Burzynski, Wilfrid Utz, Robert Andrei Buchmann, A Metamodeling Approach to Support the Engineering of Modeling Method Requirements, IEEE 27th International Requirements Engineering Conference (RE) 2019.
89. [Karagiannis2019A] Dimitris Karagiannis, Dominik Bork, Wilfrid Utz, Metamodels as a conceptual structure: some semantical and syntactical operations ,The Art of Structuring, 2019 Springer, Cham
90. [Kühn2003] H. Kühn; F. Bayer; S. Junginger; D. Karagiannis: Enterprise Model Integration. In: Bauk- necht, K.; Tjoa, A M.; Quirchmayr, G. (Hrsg.): Proceedings of the 4th International Con- ference EC-Web 2003 - Dexa 2003, Prague, Czech Republic, September 2003, LNCS 2738, Springer-Verlag, pp. 379-392.
91. [Lara2002] de Lara, Juan, and HansVangheluwe. 2002.AToM3:Atool for multiformalism and meta-modelling. Lecture Notes in Computer Science 2306:174-88.
92. [Lara2003] de Lara Jaramillo, Juan, Hans Vangheluwe, and Manuel Alfonseca Moreno. 2003. Using meta-modelling and graph grammars to create modelling environments. In Electronic notes in theoretical computer science, vol. 72, edited by Paolo Bottoni and Mark Minas. NewYork: Elsevier.
93. [Lara2010] Juan De Lara and Esther Guerra. Deep meta-modelling with metadepth. In Objects, Models, Components, Patterns, pages 1–20. Springer, 2010
94. [Lara2012] Juan de Lara, Esther Guerra, Ruth Cobos, and Jaime Moreno-Llorena. Extending deep meta-modelling for practical model-driven engineering. The Computer Journal, page bxs144, 2012.
95. [Lauer2019] Fabien Lauer, Gérard Bloch-Hybrid System Identification Theory and Algorithms for Learning Switching Models, Springer Nature Switzerland AG 2019
96. [Lesk1975] M. E. Lesk and E. Schmidt, Lex: A lexical analyzer generator. Bell Laboratories Murray Hill, NJ, 1975.
97. [Manifesto] Manifesto for Agile Software Development, http://agilemanifesto.org/
98. [Macias2018] Fernando Macías, Adrian Rutle, Volker Stolz, Roberto Rodriguez-Echeverria, Uwe Wolter, An Approach to Flexible Multi-level Modelling, Enterprise Modelling and Information Systems Architectures Vol. 13, No. 10 (2018). DOI:10.18417/emisa.13.10

99. [Manes1986] Ernest G. Manes, Michael A. Arbib. Algebraic Approaches to program semantics – Springer – Verlag New York Berlin Heidelberg London Paris Tokyo – 1986
100. [Milner1989] R. Milner- Communication and Concurrency, Prentice-Hall, Inc. Upper Saddle River, NJ, USA 1989.
101. [Milner2009] Robin Milner, The Space and Motion of Communicating Agents, Cambridge University Press, 2009. ISBN 978-0-521-73833-0
102. [Mironescu2020] Mironescu I., Crăciunean D.C., Florea A., Bondrea I. (2020) Improving the Training Methods for Designers of Flexible Production Cells in Factories of the Future. In: Camarinha-Matos L.M., Afsarmanesh H., Ortiz A. (eds) Boosting Collaborative Networks 4.0. PRO-VE 2020. IFIP Advances in Information and Communication Technology, vol 598. Springer, Cham. https://doi.org/10.1007/978-3-030-62412-5_24
103. [MetaEditA] "MetaEdit+ Workbench User's Guide." [Online]. Available: http://www.metacase.com/support/50/manuals/mwb/Mw.html.
104. [MetaEditB] MetaEdit+ User's Manuals, MetaCase, http://www.metacase.com/support/45/manuals/index.html.
105. [metaborg] http://www.metaborg.org/en/latest/
106. [Mosterman2004] P. J. Mosterman and H. Vangheluwe. Computer Automated Multi-Paradigm Modeling: An Introduction. SIMULATION: Transactions of the Society for Modeling and Simulation International, 80(9):433–450, 2004. Special Issue: Grand Challenges for Modeling and Simulation.
107. [Muller2018] Daniel Muller, Christin Schumacher, and Felix Zeidler, Intelligent Adaption Process in Cyber-Physical Production Systems, Leveraging Applications of Formal Methods, Verification and Validation Distributed Systems, 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5–9, 2018, Proceedings, Part III, © Springer Nature Switzerland AG 2018.
108. [OASIS2007] OASIS.Web Services Business Process Execution Language Version 2.0 May 2007, http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.doc
109. [OMG2003] OMG, MDA Guide V1.0, http://doc.omg.org/formal/03-05-01 , May 2003.
110. [OMG2011] OMG. OMG MOF 2 XMI Mapping Specification. (August), 2011.
111. [OMG2010] OMG. Object Constraint Language, v2.2. Management, 03(February),2010.
112. [OMG2015] OMG. Unified Modeling Language: http://www.omg.org/spec/UML/2.5, 2015. formal/2015-03-01
113. [OMG2010A] OMG. Object Management Group. Unified Modeling Language, Infrastructure Version 2.3, 2010. URL http://www.omg.org/spec/UML/2.3/Infrastructure/PDF/. 30, 38
114. [OMG2015A] OMG. Meta Object Facility (MOF) Core Specification: http://www.omg.org/spec/MOF/2.5, 2015. formal/2015-06-05
115. [OMG_BPMN2011] OMG. Business Process Model and Notation (BPMN): http://www.omg.org/spec/BPMN/2.0, 2011. formal/2011-01-03
116. [OMG2001] OMG. CWM Partners; "Common Warehouse Metamodel (CWM)Specification", Feb. 2001 http://www.cwmforum.org/spec.htm.
117. [OMG2002] OMG TC; "MOF 2.0 Query/Views/Transformations RFP", 2002. http://cgi.omg.org/cgi-bin/doc?ad/02-04-10.
118. [OMG2003] OMG. Object Management Group: MDA Guide, Version 1.0.1, 12. June 2003. http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf, access 30 November 2004.
119. [OMG2012] OMG. CORBA, 2012, www.corba.org.
120. [OMG_MDA2014] Object Management Group Model Driven Architecture (MDA) MDA Guide rev. 2.0 OMG Document ormsc/2014-06-01
121. [OMG_MDA] http://www.omg.org/mda/
122. [OMG_MOF] http://www.omg.org/mof/

123. [Openmodels] http://www.openmodels.at
124. [Plotkin1981] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.
125. [Paun1998] Gh. Paun, Computing with membranes, J. Computer System Sciences,2000), in press, and TUCS Research Report No. 208, November, (http://www.tucs.fi),(1998).
126. [Padua2011] David Padua (Ed.)- Encyclopedia of Parallel Computing, Springer Science+Business Media, LLC 2011.
127. [Parr2007] T. Parr, The Definitive ANTLR Reference: Building Domain-Specific Languages. Pragmatic Bookshelf, 2007.
128. [Peffers2007] Peffers, Ken; Tuunanen, Tuure; Rothenberger, Marcus ; Chatterjee, Samir: A Design Science Research Methodology for Information Systems Research. In: Journal of Management Information Systems 24 (2007), No. 3, pp. 45–77. – ISSN 0742–1222
129. [Plump2010] D. Plump, 'Checking graph-transformation systems for confluence', ECEASST, vol. 26, 2010. DOI: 10.14279/tuj.eceasst.26.367.
130. [Plump2011] D. Plump, 'The design of GP2', in Proceedings 10th International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2011), ser. Electronic Proceedings in Theoretical Computer Science, vol. 82, 2011, pp. 1–16. DOI: 10.4204/EPTCS.82.1.
131. [Plump2019] D. Plump, 'Computing by graph transformation: 2018/19', Department of Computer Science, University of York, UK, Lecture Slides, 2019.
132. [SIRIUS] https://www.eclipse.org/sirius
133. [Visic2014] N. Višić and D. Karagiannis, "Developing Conceptual Modeling Tools Using a DSL," in Knowledge Science, Engineering and Management, R. Buchmann, C. V. Kifor, and J. Yu, Eds. Springer International Publishing, 2014, pp. 162–173.
134. [Visic2015] Višić, N., Fill, H.-G., Buchmann, R., Karagiannis, D. 2015. A domain-specific language for modelling method definition: from requirements to grammar. In: Rolland, C., Anagnostopoulos, D., Loucopoulos, P., Gonzalez-Perez, C. (eds.), Proceedings of RCIS 2015, IEEE, 286-297, DOI= 10.1109/RCIS.2015.7128889
135. [Visic2016] N. Višić, Language-Oriented Modeling Method Engineering, Doctoral Thesis , University of Vienna, Scientific Advisor: o. Univ.-Prof. Prof.h.c. Dr. Dimitris Karagiannis (2016)
136. [Voelter2010] Markus Voelter, D S L E n g i n e e r i n g Designing, Implementing and Using Domain-Specific Languages, http://dslbook.org, 2010 – 2013
137. [Walters2006] R. F. C. Walters, Categories and ComputerScience, Cambridge Texts in Computer Science ,Edited by D. J. Cooke, Loughborough University, 2006.
138. [Wainer2011] Gabriel A. Wainer, Pieter J. Mosterman- Discrete-Event Modeling And Simulation Theory and Applications , 2011 by Taylor and Francis Group, LLC
139. [Rus2015] Rus, Teodor. Computer-based problem solving process, World Scientific Publishing Co. Pte. Ltd. 5 Toh Tuck Link, Singapore 596224 ISBN 978-9814663731,2015
140. [Rutle2012] Adrian Rutle, Alessandro Rossini, Yngve Lamoa, UweWolter, A formal approach to the specification and transformation of constraints in MDE, The Journal of Logic and Algebraic Programming Volume 81, Issue 4, May 2012, Pages 422-457
141. [Rumpe2016] Bernhard Rumpe, Modeling with UML Language, Concepts, Methods, Springer International Publishing Switzerland 2016.
142. [Rumpe2017] Bernhard Rumpe, Agile Modeling with UML Code Generation, Testing, Refactoring, Springer International Publishing AG 2017.
143. [Rozsnberg1997] G. Rozsnberg (ed), HANDBOOK OF GRAPH GRAMMARS AND COMPUTING BY GRAPH TRANSFORMATION, Copyright 1997 by World Scientific Publishing Co. Re. Ltd.

144. [Srdjan] Srdjan Živković, Marion Murzek, Harald Kühn, Bringing Ontology Awareness into Model Driven Engineering Platforms BOC Information Systems GmbH, Wipplingerstrasse 1,1010 Vienna, Austria
145. [Spivak2014] David I. Spivak, Category Theory for the Sciences The MIT Press Cambridge, Massachusetts London, England ,2014 Massachusetts Institute of Technology
146. [Sprinkle2010] Sprinkle J., Rumpe B., Vangheluwe H., Karsai G. (2010) Metamodelling – State of the Art and Research Challenges In: MBEERTS Giese, H. et al. (ed.) Vol. LNCS 6100 Springer, pp. 57–76
147. [Spoofax2019] Spoofax Documentation Release 2.5.7 MetaBorg Jul 02, 2019
148. [W3C2004] W3C. Recommendation. RDF Vocabulary Description Language1.0: RDF Schema, 2004. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
149. [W3C2012] W3C OWL 2 Web Ontology Language Document Overview (Second Edition), 11 December 2012, http://www.w3.org/TR/2012/REC-owl2-overview-20121211/
150. [W3C2002] W3C. Web Services Conversation Language (WSCL) 1.0 14 March 2002 http://www.w3.org/TR/2002/NOTE-wscl10-20020314/
151. [W3C2002A] W3C. Web Service Choreography Interface (WSCI) 1.0 W3C 8 August 2002 http://www.w3.org/TR/2002/NOTE-wsci-20020808
152. [Fokkink2007] Wan Fokkink, Introduction to Process Algebra, Computer Science-Monograph (English), Springer-Verlag, 2nd edition April 10, 2007.
153. [FMI2019] Functional Mock-up Interface for Model Exchange and Co-Simulation,Document version: 2.0.1 October 2nd 2019, https://fmi-standard.org/.
154. [INTO-CPS2017] INTO-CPS Tool Chain User Manual, Deliverable Number: D4.3a Version: 1.0 Date: December, 2017 Public Document, http://into-cps.au.dk.
155. [Weske2012] Weske, Mathias. Business Process Management - Concepts, Languages, Architectures, 2nd Edition. Springer 2012, ISBN 978-3-642-28615-5, pp. I-XV, 1-403.
156. [Winskel1993] Winskel Glynn, Nielsen Mogens, Models for Concurrency, November 1993
157. [Winskel2009] Winskel Glynn, Topics in Concurrency Lecture Notes, April 2009.
158. [Wieczorek2017] Wojciech Wieczorek-Grammatical Inference Algorithms, Routines and Applications, Springer International Publishing AG 2017.
159. [Wolter2007] U. Wolter and Z. Diskin. The Next Hundred Diagrammatic Specification Techniques – An Introduction to Generalized Sketches. Technical Report Report No 358, Department of Informatics, University of Bergen, July 2007.
160. [Wolter2015] Uwe Wolter, Zinovy Diskin, The Next Hundred Diagrammatic Specification Techniques, A Gentle Introduction to Generalized Sketches, 02 September 2015 : https://www.researchgate.net/publication/253963677.
161. [wp32020] https://digifof.eu/deliverables/wp3-fof-designer-innovative-teaching-methods-and-tools
162. [XtextA] Xtext Documentation, https://eclipse.org/Xtext/documentation/
163. [Xtext2014] Xtext – Community Online]. Available: http://www.eclipse.org/Xtext/community.html. [Accessed: 27-Sep-2014].
164. [Xtend] Xtend Documentation , https://eclipse.org/xtend/documentation/
165. [Zivkovic2007] Zivkovic, S.; Kühn, H.; Karagiannis, D.: Facilitate Modelling Using Method Integration: An Approach Using Mappings and Integration Rules. In: Österle, H.; Schelp, J.; Winter, R.; (Eds.): Proceedings of the 15th European Conference on Information Systems (ECIS2007) - "Relevant rigour - Rigorous relevance", St.Gallen, Switzerland. June 2007, pp. 2038-2050.
166. [Zeigler2019] B.P. Zeigler, A. Muzy, E. Kofman, Theory of Modeling and Simulation Discrete Event and Iterative System Computational Foundations, Academic Press (2019)