!

SZAKDOLGOZAT-FELADAT

**Stasz Balázs (BD8P7P)**

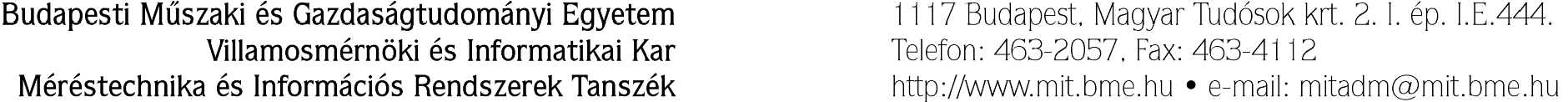
szigorló informatikus hallgató részére

# Felhasználói felületek modellvezérelt tervezése

Napjainkban a modellvezérelt módszerek fontos és hatékony segítséget nyújtanak számos szoftverfejlesztési területen, például a webes és mobil alkalmazások készítése során. A magasszintű modellezési nyelvekkel lehetőség nyílik arra, hogy az alkalmazás számos aspektusát (felhasználói űrlapok, üzleti logika bizonyos részei, kommunikáció) absztrakt mérnöki modellek segítségével tervezzük meg, melyből platformspecifikus modellek, majd a végső kód automatikus transzformációk és kódgenerálás során állnak elő. Ilyen eszközökkel a kézi módszereknél lényegesen hatékonyabban készíthetők komplex szoftverrendszerek, a hatékonyság növekedése fejlesztési költségeket takarít meg (mind a gyorsabb munkavégzés, mind pedig a hibák számának csökkentése, illetve a drága tervezési hibák korai javítása miatt).

A BlackBelt Kft. belső fejlesztési projekt keretében kifejlesztette a Judo keretrendszert, melynek fő célja webes és mobil alkalmazások modellvezérelt fejlesztésének támogatása. A folyamat során a rendszerelemzők által elkészített magas szintű UML modellekből előbb platformfüggetlen, majd platformspecifikus modellek készülnek modelltranszformációk segítségével, a lánc végén a kimenet pedig az alkalmazások prototípusának (vázának) forráskódja is előállítható kódgenerátorok segítségével.

A hallgató feladata, hogy esettanulmány jelleggel kifejlesszen egy a Judo keretrendszerbe beépülő, Eclipse Modeling Framework, valamint Epsilon és VIATRA technológiákra épülő nyílt forrású eszközt, melynek segítségével web platformon futó alkalmazások felhasználói felületének generatív fejlesztése támogatható.



A szakdolgozat kidolgozása a következő részfeladatok megoldását igényli:

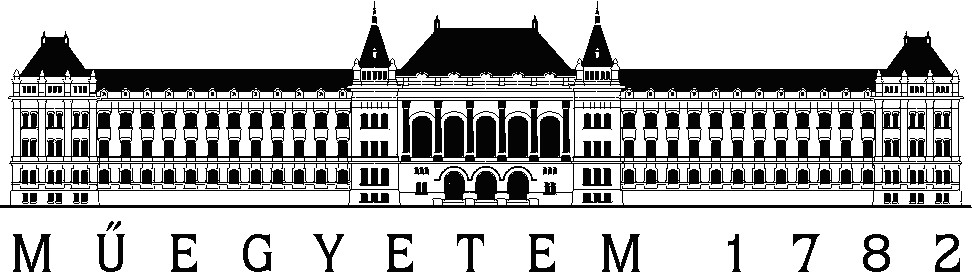
* Végezzen irodalomkutatást, és mutassa be a Judo keretrendszert, valamint a legfontosabb kapcsolódó modellvezérelt technológiákat.
* Tervezzen és valósítson meg egy modellezési nyelvet, mellyel webes alkalmazások felhasználói felülete megtervezhető.
* Tervezzen és valósítson meg automatizált modelltranszformációkat, melyekkel a magasszintű specifikációkat leíró modellekből előállíthatók az alkalmazás-specifikus modellek. Valósítsa meg a modelltranszformációkat inkrementális formában is, melyek lehetővé teszik a későbbi, magas szinten elvégzett modellezési változtatások összefésülését az alacsonyabb szintű modellek változásaival.
* Egy, a VIATRA és az Epsilon modelltranszformációs keretrendszerekben megvalósított transzformációkat összehasonlító esettanulmányon keresztül mutassa be a rendszer működését.

**Tanszéki konzulens:** Dr. Ráth István, adjunktus

**Külső konzulens:** Turi Gyula, szoftvermérnök, BlackBelt Kft.

Budapest, 2017. október 3.

…………………… Dr. Dabóczi Tamás tanszékvezető



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Méréstechnika és Információs Rendszerek Tanszék

Felhasználói felületek modellvezérelt tervezése

Szakdolgozat

|  |  |
| --- | --- |
| *Készítette* | *Konzulens* |
| Stasz Balázs | Dr. Ráth István |
|  |  |

2018. december 07.

# Tartalomjegyzék

Kivonat

1. Bevezető
   1. A feladat motivációja
   2. A feladat értelmezése
   3. A dokumentáció tartalma
2. Technikai háttérismeretek
   1. Modelvezérelt szoftverfejlesztés előnyei
   2. Eclipse Modeling Framework (EMF)
   3. Epsilon
   4. VIATRA
   5. JUDO keretrendszer
3. A probléma elemzése
   1. PSM metamodel bemutatása
   2. UI metamodel bemutatása
4. A probléma leképzése Epsilon/VIATRA feladattá
   1. Epsilon
      1. Batch transzformáció
      2. ETL fájlok bemutatása
      3. EGL fájlok bemutatása
   2. VIATRA
      1. Reaktív transzformáció
      2. Rule-ok bemutatása
      3. Patternek bemutatása
5. A megoldások kiértékelése
   1. Epsilon és VIATRA tanszformációk eredményeinek összehasonlítása
   2. Epsilon EGL kimenet értékelése
   3. A JUDO framework felhasználásával az előállított JSON-ből UI készítése
   4. Batch és Reaktív transzformáció összehasonlítása
6. Az Epsilon és VIATRA szoftverek használata
   1. Részletes leírás az Epsilon transzformáció lefuttatásához
   2. Részletes leírás a VIATRA transzformáció lefuttatásához
   3. Részletes leírás az összehasonlító teszt futtatásához
7. Összefoglalás

# Bevezetés

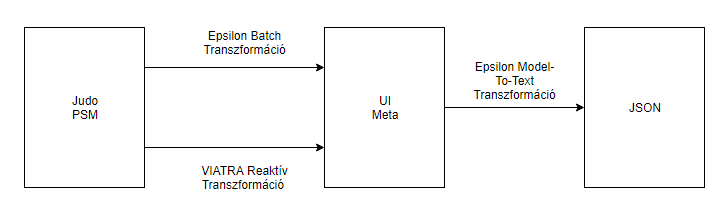
* 1. A feladatom motivációja

A Blackbelt Technology Kft belsős projektjeként indult, amelynek én is részese vagyok. Célja a Modelvezérelt szoftverfejlesztési folyamat megreformálása. Eddig a fejlesztési folyamat Modelio és Python alapokon működött, melynek kezelhetőségi, bővíthetőségi és kompatibilitási problémái voltak. Az új folyamat MagicDraw alapokon nyugszik, így felmerült az igény az egész transzformálási folyamat megújítására, amely könnyebben karbantartható és már rendelkezésre áll számtalan bővíthetőségi lehetőség. Ezen kritériumok miatt esett a választás az Eclipse alapú Epsilon és VIATRA keretrendszerekre.

A VIATRA keretrendszer elsősorban a reaktív transzformáció támogatása miatt került előtérbe. A reaktív transzformációban rejlő lehetőségek megkönnyítenék mind az üzleti elemzők, mind a fejlesztői csapat munkáját és felgyorsítaná a model változtatással járó kellemetlenségek megoldását.

* 1. A feladat értelmezése

A kitűzött cél egy olyan model transzformációs folyamat felépítése, amely végén egy, a JUDO keretrendszer által feldolgozható JSON formátumú szöveges állomány jön létre, melyből egy alap funkciókra (CRUD) képes felhasználói felület generálható.



A transzformáció kiindulópontja a Blackbelt Kft-től kapott JUDO PSM metamodel, amely egy EMF ECore fájlban van tárolva. Az ezen alapuló modellek transzformációja a következő lépés, amely kétféleképpen történhet. Az első esetben az Epsilonnal megvalósított transzformációval, amely a teljes kiinduló modelt transzformálja, a változtatások számától függetlenül. A második esetben VIATRA reaktív transzformáció alkalmazásával, amely csak az előző állapot óta történt változásokra reagálva végzi el a szükséges transzformációs lépéseket. A transzformáció az általam elkészített UI metamodellre történik, amely egy felhasználói felület általános leírását teszi lehetővé. Végezetül az Epsilonnal megvalósított Model-To-Text transzformáció eredményeképp egy JSON formátumú állomány jön létre, amely a JUDO keretrendszernek megfelelő formázással és tartalommal rendelkezik, így előállítható belőle egy működő felhasználói felület.

A szakdolgozat terjedelme a JUDO PSM modellből való transzformációtól indul és a JSON állomány előállításáig tart, beleértve a UI metamodell előzetes elkészítését is.

1.3 A dokumentáció tartalma

A második fejezetben bemutatásra kerülnek a modellvezérelt szoftverfejlesztés előnyei, illetve a felhasznált technológiák rövid ismertetése. Ezek ismerete elengedhetetlen az elkövetkezendő fejezetek megértéséhez.

A harmadik fejezetben a két, transzformációban használt metamodell részletes leírása olvasható.

A negyedik fejezet két fő részből áll. Először az Epsilon transzformációhoz létrehozott ETL fájlok részletes bemutatása történik, ezt követi a kód generáláshoz létrehozott EGL fájlok bemutatása. A második részben a VIATRA transzformációhoz felhasznált Rule és Pattern fájlok leírása olvasható. A fejezet igyekszik választ adni a transzformációs szabályok mögött megbúvó logikára.

Az ötödik fejezetben a transzformációk és generátorok eredményeinek kiértékelése és összehasonlítása olvasható. Ezen felül egy példa UI, amely az Epsilon által előállított JSON-ből keletkezett a Judo keretrendszer felhasználásával.

A hatodik fejezetben három leírás található, amelyek tartalmazzák az Epsilon és VIATRA transzformációk futtatásának módját, illetve ezek eredményének összehasonlítására szolgáló teszt futtatását.

Végezetül, a hetedik fejezetben egy rövid összefoglalás olvasható a feladat elvégzése közben szerzett tapasztalatokról.

# Technikai háttérismeretek

* 1. Modelvezérelt szoftverfejlesztés előnyei

A modellvezérelt szoftverfejlesztés előtérbe kerülése több tényezőnek köszönhető. Ezen tényezők mind a fejlesztési folyamatra, mind az üzleti kiadásokra pozitív hatással vannak.

Az első ilyen tényező az automatizált kódgenerálás, amely használatával lehetőség van gyakran használt kódrészletek automatikus generálására. Például egy osztály esetén a kötelező létrehozás/törlés/frissítés (CRUD) operációk. Ezzel jelentős idő és pénz takarítható meg. A fejlesztői csapatnak elegendő az általános funkcionalitáson túlmutató logikát implementálnia.

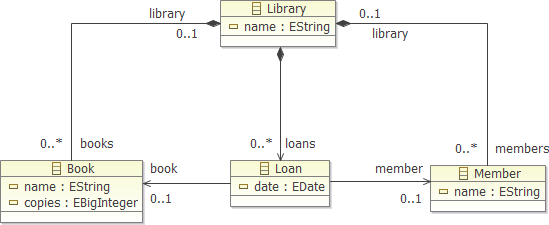
A transzformáció megkezdése előtt lehetőség van validálni a modelt. Így még a modell transzformáció megkezdése előtt kiderül, ha tervezési időben hiba csúszott a kivitelezésbe.

Segíti a szoftver dokumentálását. Egy jól megtervezett modell már önmagában megkönnyíti az összefüggések megértését, kiemeli az összetartozásokat.

Mivel a platform-független (PIM) modell nem vesz tudomást arról, hogy milyen architektúrán fogják a készülő szoftvert implementálni, így elegendő a szemtantikai működést megtervezni és a késöbbiekben, ha a szükség úgy hozza, architekturális változások a modell változtatása nélkül végrehajthatóak.

2.2 Eclipse Modeling Framework (EMF)

Az Eclipse Modeling Framework (EMF) egy modellező és kódgeneráló keretrendszer. Az ECore metamodell az EMF alapeleme, melynek segítségével kifejezhető bármely más model, akár önmaga is. Az ECore metamodelleken alapuló modelleket hozhatunk létre, melyek az ECore metamodelljükben megadott eszközkészlettel rendelkeznek.

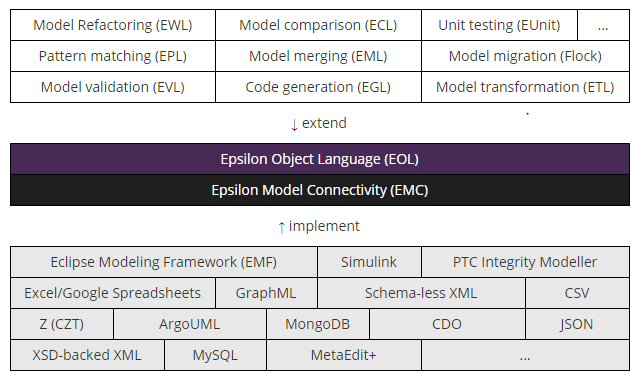


Az alábbi példán egy könyvtár egyszerű metamodellje látható. Egy könyvtár (Library) objektumnak lehetnek könyvei (Book), tagjai (Member), kikölcsönzései (Loan), illetve neve (name), amely String típusú.

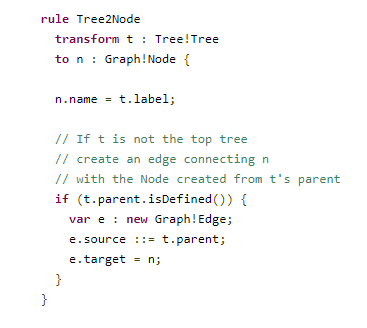
2.3 Epsilon

Az Epsilon nyelvek és eszközök gyűteménye, amelyeket kód generálásra, Model-To-Model transzformációra, model validálásra, model összehasonlításra és refaktorálásra terveztek.

Az Epsilon magja az Epsilon Object Language (EOL), egy imperatív model orientált nyelv, amely kombinálja a proceduális Javascriptet az OCL lekérdező képességeivel.



Epsilon Transformation Language (ETL) : Szabály alapú Model-To-Model transzformációs nyelv, amely támogatja a több modelből több modelbe való transzformálást, szabályok öröklését, a lazy és greedy szabályokat, és mind a bemeneti, mind a kimeneti modelek módosítását és lekérdezését.



Az alábbi példában a Tree model Tree objektumát transzformálja a Graph model Node objektumává.

Epsilon Generation Language (EGL) : Sablon alapú Model-To-Text nyelv kód és dokumentáció generáláshoz.

2.4 Eclipse VIATRA

Az Eclipse VIATRA egy model és adat transzformációs keretrendszer, amely esemény vezérelt és reaktív transzformációkra fókuszál.

The Eclipse **VIATRA** framework is a model and data transformation tool that helps move information back and forth in the most efficient way. The platform focuses specifically on event-driven and reactive transformations, which happen on-the-fly as the models change.

VIATRA is a long-running project that incorporates the results of over 15 years of active research and development. VIATRA offers

* a domain-specific language for graph queries and reactive transformations
* a Java runtime library which is scalable and easy to embed.

VIATRA primarily targets model-driven engineering tools based on the industry-standard Eclipse Modeling Framework, though it can be used in other contexts as well. VIATRA is an enabling technology and its biggest benefit is the power it gives you to close abstraction gaps. These exist in every data-driven application where structured data needs to be moved across various languages or representation formats.

VIATRA helps you create model transformations that convert objects between languages. Trivial transformations (i.e. copying pieces of data from one document to another) are easy with any general purpose programming language such as Java. In contrast, VIATRA gives you easy-to-use and reusable programming patterns to address non-trivial transformations, such as

* Changing of structure, i.e. converting a UML model to a source code.
* Derived data and abstractions: manipulating data during conversion, e.g. filtering information for creating ‘abstract views’
* Combining multiple sources of information in a significant way to produce a new result, e.g. when creating data streams for dashboards
* Changing structure: projection of data containing only relevant information during the conversion phrase
  1. JUDO keretrendszer

A JUDO keretrendszer a Blackbelt Technology Kft. saját fejlesztésű Model-To-Model transzformációs keretrendszere. A JUDO keretrendszer működése többszörös M2M és M2T transzformációkon alapszik. Az üzleti elemző által megtervezett UML modellből először egy JUDO specifikus PSM modell készül. Ebből a PSM modellből transzformálódik például a UI, Java és RDBMS modellek. Végezetül pedig ezen modellek Model-To-Text transzformációjával keletkeznek azon állományok, melyeket a JUDO keretrendszer feldolgoz és önmagában működő szoftvert állít elő belőle. Az alap funkciók a CRUD műveleteket foglalják magukba.

# A metamodelek elemzése

# Hivatkozások

A modelvezérelt szoftverfejlesztés előnyei:

<http://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/>

Epsilon:

<https://www.eclipse.org/epsilon/doc/>