|  |
| --- |
| Szakdolgozat |
| Minta előlap |

Összefoglaló

A szoftverfejlesztők az informatika kezdete óta próbálnak olyan technikákat alkalmazni, melynek segítségével egyszerűbbé és gyorsabbá válhat egy szoftver létrehozásának a folyamata. Ezek módszerek egyik modern változata modell alapú fejlesztés.

A modell, hivatalos definíciója szerint, a valóság egy részének egyszerűsített képe, amely a rendszert helyettesíti bizonyos megfontolásokkal.[[1]](#footnote-1) A fejlesztők az elkészítendő szoftverhez modelleket készítenek tetszőleges részletességgel, és ezen modelleket felhasználják a fejlesztés minden fázisában, főként a tervezés és az implementálás fázisában, ha a vízesésmodellt nézzük.[[2]](#footnote-2)

Szakdolgozatom fő témája a modell alapú szoftverkészítés kipróbálása és a létjogosultságának eldöntése bizonyos mértékben. Ehhez egy régóta előforduló, de többnyire megoldott problémát újból előveszek, mely ismételten meg lesz oldva, de most modellek segítségével.

Ez a probléma nem más, mint a karakterkódolások és a karakterkészletek sokszínűségének bonyodalma. Rengeteg megoldás és szoftver készült, de egyik sem modell alapokon. Miután megvalósítom a saját megoldásomat a karakterkódolások problémájára, lesz egy eredményem, mennyire hasznosítható ez az újfajta fejlesztési módszer.

(Az informatikában egy szoftver fejlesztése során sok nehézségbe ütköznek a fejlesztők. Sokszor félreértik egymás döntéseit, ha valami éppen nincs jól dokumentálva; egyre növekvő projektekre pedig még inkább igaz az állítás: a felhalmozott adatok és elvégzendő feladatok mennyisége akár hatványosan is növekedhet a projekt növekedésével.)

Abstract

Ever since the beginning of Information Technology, software developers try to come up with new technics, that will help the process of the development for a software much faster and much simpler. One of these technics’ modern variant is the model-driven engineering.

A model, by its official definition, is a simpler representation of a piece from reality, which substitute the system with given considerations. Developers make models for their software to be prepared with as many details as they want. Later, they use these models in every phase of the development, especially in the designing and the implementation phase, if we are based in the development’s waterfall model.

The main article of my thesis is the try-out of the model-driven engineering and somewhat decide whether it is usable or not in real situation and real development. For this purpose, I will expose an old problem and I will re-solve it once again but with the usage of models.

This problem is nothing less, than the problem of character encodings and character sets and their diversity. A lot of solutions have been already made, but none of those did actually use models. After I have created my own solution for the problem of character encoding, I will acquire an outcome of how useful this new style of development can be.

Bevezető

Egy szoftver fejlesztése során egyáltalán nem újszerű ötlet diagramok készítése. Például, a szoftver tervezés fázisában a régóta használunk UML diagramokat, hogy az objektum orientált programozási elemekről tervezési döntéseket nagy hatásfokkal jegyezhessünk fel. Továbbá, a szoftverhez kapcsolódó dokumentációkban is több szerepet kapnak, mint egy szöveges leírás. Azonban ezek a diagramok nem használhatóak fel olyan mértékben, mint egy modell. Egy modell sokkal több funkciót képes ellátni, és sokkal több lehetőség van benne.

A modell alapú fejlesztés egyik fajtája a Domain Specific Language alapú alkalmazások, röviden DSL-ek. Egy DSL külsőre egy programozási nyelv, azonban nincs saját fordítója és nem is annyira sokoldalú, mint egy megszokott programozási nyelv. Egy DSL legtöbbször egy másik programozási nyelvre fordul át, vagy legalábbis egy adathalmazra, amely egy másik nyelv felhasználat. A DSL-ek tipikusan egy-egy fajta, konkrét feladat betöltésére lettek megalkotva.

A DSL-ek létjogosultsága abban rejlik, hogy a bennük való kódírás sokkal egyszerűbb és letisztultabb lehet egy átlagos nyelvhez képest, így laikusok is képesek benne programozni. Bevonhatja a szoftver megrendelőjét, a steakholder-eket a fejlesztés folyamatában, megkönnyíti a fejlesztőkkel való munkát, azáltal, hogy az ügyfél konkrétan tudja közölni az igényeit az alkalmazás doménjén keresztül.

DSL lehet például egy építészeti egységeket, kémiai vagy gazdasági folyamatokat leíró nyelv, egy programozást oktató vagy statisztikákat készítő nyelv. De a weboldalak HTML nyelve és a Linuxban használatos szkriptkészítés is DSL-nek tekinthetők, bár ezek jóval áltolánosabbak, mint amit DSL-nek szokás nevezni. A DSL másik használatos, informális neve a mini – language.

Az informatika nem egy mai találmány, fél évszázada használjuk és alkotunk benne eszközöket, illetve programokat. Ennek ellenére a többi tudományághoz képest, az informatika fejlődik és növekszik a leggyorsabb ütemben, aminek vannak következményei. Egy a sok közül, mellyel a szakdolgozatom is foglalkozik, a karakterkódolások sokféleségének a problémája.

Minden egyes szöveg vagy adat, amit az informatika világában létrehozunk, számok formájában kerül tárolásra, és egy karakterkódolás határozza meg, hogy melyik betűhöz vagy jelöléshez, melyik szám tartozik. Az informatika kialakulásának legelejétől a mai napig rengeteg féle kódolási szabványok jöttek és jönnek most is létre. Kezdetben igények szerint, földrajzi elhelyezkedés, kulturális háttér alapján jöttek létre, például az ASCII, amely kizárólag az angol billentyűzeten található karaktereket foglalja magába, vagy a JIS, mely a Japánban használt írásjeleket kódolja.

Napjainkban már inkább az egységes, szabványos és mindenhol használandó kódolások kezdenek elterjedni. Az Unicode kódolás áll e tekintetben a legjobb helyen, ez a legismertebb és a legrészletesebb kódolás, mivel a leg kézen fekvőbb karakterekkel kezdve, az ázsiai írásjeleken át, az egyiptomi hieroglifákon keresztül, az emojikkal bezárólag minden megtalálható már benne.

A karakterkódolással kapcsolatos egyik problémát az elavult karakterkészletek jelentik. A régebbi kódolások nem tudnak egyik napról a másikra eltűnni. Nem használt állományok még mindig ilyen kódolásokban vannak tárolva, melyek használhatatlanná válhatnak. Régi programokat nem lehet lefordítani vagy újra felhasználni, egész adatbázisok lehetnek elérhetetlenek, más kontinensen lévő nemzetek régi fájljaihoz nem lehet hozzáférni, könyvtárban tárolt szöveges állományokat nem tudunk elolvasni, stb… Pusztán azért, mert olyan régi kódolást használnak, amelyeket már fel se lehet ismerni.

A másik problémát hasonló alapon az Internet jelenti. A különböző kódolások miatt nem egyszerű e-maileket írni, weblapokat letölteni, alkalmazásokat futtatni. Mongol barátomnak angolt, vagy angol karakterekre interpretált mongolt kell használni, hogy a családjának üzeneteket küldhessen, pusztán azért, mert a webes szolgáltatásokon és alkalmazásokon nincsenek támogatva a cirill betűk.

Összefoglalva, a karakterkódolások közti átvitel problémát jelent, és nincs rá megfelelő eszköz, amely ezt legalább megkönnyítené.

A karakterátkódoláshoz könnyen készíthető valamilyen DSL. Egy átkódolás lényegi része, hogy milyen bitsorozatból milyen másik bitsorozat lesz. Ezek leírása egyszerűen és zajmentesen leírhatóak lehetnek egy DSL-ben. Szakdolgozatom is egy ilyet szeretne megvalósítani. Az én DSL nyelvemben megadható lehetne a kiinduló kódolás, a célkódolás és a konvertálandó bitsorozatok szimpla szöveges formátumban, bármilyen zavaró szintaxis nélkül.

Szakdolgozatom témája egy kódgenerátor készítése. A fent említett DSL-t már lehetne arra használni, hogy a megírt átkódolásból valamilyen közismert programozási nyelven megírt API-t (application programming interface) generáljak, hozzáadva azon „zajokat”, amely a DSL eltávolított. A programot nem kézzel kellene megírni, hanem generálnám, így a hosszan megírt karakterátkódolási szabályokat könnyen lehetne kezelni.

A feladatot egy DSL szerkesztőben kell elkészíteni, mely az Eclipse-ben platformon elérhető. A szerkesztőnek, vagyis Eclipse modulnak a neve Xtext, melyben gyorsan, egyszerűen és mégis eredményesen lehet DSL-t készíteni. Az Xtext nagymértékben támaszkodik az Eclipse Modelling Frameworkre, mely az Eclipse modellezéshez kapcsolódó legelterjedtebb modulja. Az Xtext segítségével Java nyelvben lehet megírni a DSL részeit és rengeteg segédosztályt és segédfüggvényt biztosít, hogy a DSL miden funkcióját az igényekhez igazítsuk.

A konkrét igénnyel Kövesdán Gábor konzulensem állt elő. Felhívta a figyelmemet, hogy a beágyazott rendszerekben használt FreeBSD-ben nem elérhető olyan karakterkódolásokhoz használt konvertáló, mely elég kompakt ahhoz, hogy elférjen egy beágyazott rendszer szűkös memóriájába. Egy C nyelvben megírt függvénycsoportra lenne szükség, melyben lekérhetjük az egyes átkódolásokban használatos bitsorozatokat és a konvertáláshoz szükséges egyéb információkat. Ezt elég érdekesnek találtam ahhoz, hogy szakdolgozatom témájának válasszam, és remélhetőleg egy open source projekthez is hozzájárulhatok.

Hivatkozások

<https://en.wikisource.org/w/index.php?title=Cause,_Effect,_Efficiency_%26_Soft_Systems_Models>

<https://vik.wiki/images/6/6c/Szofttech_objektumorientaltszoftverfejlesztes_konyv.pdf>

<https://www.omg.org/mda/mda_files/Model-Driven_Architecture.pdf>

<https://inf.mit.bme.hu/sites/default/files/materials/category/kateg%C3%B3ria/oktat%C3%A1s/bsc-t%C3%A1rgyak/informatikai-rendszertervez%C3%A9s/17/MDSD-BriefPreview.pdf>

<https://www.sciencedirect.com/science/article/pii/S1477842415000408?via%3Dihub>

1. TODO hivatkozás <https://en.wikisource.org/w/index.php?title=Cause,_Effect,_Efficiency_%26_Soft_Systems_Models> [↑](#footnote-ref-1)
2. TODO hivatkozás <https://vik.wiki/images/6/6c/Szofttech_objektumorientaltszoftverfejlesztes_konyv.pdf> [↑](#footnote-ref-2)