OCL kifejezések vizsgálata

A projekt célja

A modern szoftverfejlesztés megköveteli megfelelõ szoftverfejlesztési módszertanok használatát. A szoftverfejlesztési módszertanok alapvetõ lépése a modellezés. A széles körû modellezéshez ma már megfelelõ kellékek állnak rendelkezésre. A modellezõ eszközök közös jellemzõje a használt nyelv, egy grafikus jelölésrendszer az UML.

A modern modellezõ szoftverek lehetõvé teszik a megalkotott UML modell alapján az automatikus kódgenerálást. Azonban ez a kód általában csak kódváz, azaz például egy java objektumorientált kódban csak gyakorlatilag osztály-, attribútum-, függvénydefiníciókkal és kommentekkel találkozhatunk. A kódgenerátorok nem képesek az UML modellben megadott viselkedést (pl. idõbeliséget, kényszereket) kód szintjén is megjeleníteni (az UML modellek viselkedés-leírásának egyik eszköze az OCL /*Object Constraint Language*/).

A projekt alapcélja, hogy létrehozzon egy olyan kódgenerátort, amely az UML-ben megadott viselkedést kód szintjén is képes megjeleníteni. Távolabbi célok, az UML kifejezõerejének kiterjesztése az erõsebb modellek létrehozása érdekében, hogy ezekbõl automatikus generációval pontosabb forráskódot lehessen elõállítani. Ha a modellezõ eszköz kifejezõereje nagyobb, akkor a specifikáció pontosabban képezhetõ le a modellre és ha a modell nagy részébõl automatikusan elõállítható a programkód, akkor a programozók feladatát nagy mértékben megkönnyíthetjük. Egyrészt kevesebb leírt kódsorra van szükség, ennek következményeként kevesebb a szintaktikai hiba lehetõsége, másrészt, ha képes a kódgenerátor automatikus hibaellenõrzõ mechanizmusok implementálására, akkor részben kiküszöbölhetõek az alattomosabb szemantikai hibák is.

Összességében elmondhatjuk, hogy ha a cél megvalósul, akkor ennek egyenes következménye lesz a rendszer fejlesztési idejének rövidülése, amely természetesen a költségek csökkenéséhez vezet. A kész rendszer megbízhatósága is magasabb azáltal, hogy várhatóan kevesebb lesz benne a bennmaradó hibák száma, mint egy hagyományos fejlesztés során kialakuló rendszerben.

Hibaellenõrzõ mechanizmusok

Az elmúlt félév során megvizsgáltam a hibaellenõrzõ mechanizmusok kezelésének kérdését, azaz hol, milyen szinten érdemes ezeket a mechanizmusokat implementálni. Két alternatíva közül választhattam. Az elsõ megoldás minden objektumhoz létrehoz egy metaobjektumot. Minden metaobjektum elkapja a hozzá tartozó objektumnak küldött és az objektumtól kifelé menõ üzeneteket. Az üzeneteken ellenõrzést végez, és ha nem talált semmilyen szokatlant, akkor továbbküldi az üzeneteket az eredeti címzettnek. Ha az ellenõrzés hibát jelez, akkor elindít valamilyen hibakezelõ eljárást vagy kivételt dob a metaobjektum.

A második megoldásnál, minden objektum belsejében történik az ellenõrzés, tehát például egy függvény meghívásakor az átadott paraméterek helyességét a függvény maga ellenõrzi.
Az elsõ megoldás metaobjektum-szinten valósítja meg a hibakezelést. Ennek elõnye, hogy általános megoldás, megtartja az eredeti objektumot. Sajnos azonban a megvalósítás a reflektív programozás elvét követi, amely speciális programozási nyelvet igényel (Open C++). A második megoldás nem tartja meg az eredeti objektumot, de cserébe a megvalósítás egyszerûbb. A kódgenerátor az említett elõnyei miatt az utóbbi alternatíva szerinti hibakezelõ mechanizmusokat fogja megvalósítani.

OCL

Az UML-beli viselkedésleírás egyik eszköze az Object Constraint Language. Az OCL egy OO, típusos leíró nyelv, amellyel elõírások, szabályok (kényszerek) fogalmazhatóak meg UML modelleken. Az OCL az az eszköz, amely segítségével hibaellenõrzõ kód generálható. A félév során megvizsgáltam ennek szintaktikáját és szemantikáját egyaránt.

Kódgenerátor

A kódgenerátort java nyelven fogom implementálni. Bemenete egy rendszer UML modellje, kimenete pedig java forráskód. A bemenet tipikusan XML alapú, ennek következtében szükség lesz XML értelmezõre. Jelenleg kettõ elterjedtebb keretrendszer közül lehet választani, ezek a SAX (Simple API for XML) és a DOM (Document Object Modell). Az OCL kifejezések szintaktikai elemzéséhez szükséges egy OCL értelmezõ, amely az IBM-tõl és a Drezdai Mûszaki Egyetemtõl is beszerezhetõ. Az OCL kifejezések tartalmazhatnak szintaktikai édesítõszereket, ezért a további feldolgozhatóság kedvéért minden OCL kifejezést normalizálni kell. Ezután a már szabványos alakra hozott OCL kifejezéseket szintaktikai értelmezésnek kell alávetni. Az elemzett kifejezésbõl ezután töredék java kódot kell elõállítani. Az utóbbi funkciókat megvalósító program demo verziója már létezik a drezdai egyetemen, azonban ennek használhatósága a jelenlegi munkában még nem tisztázott. Az elõállított töredék java kódot be kell illeszteni egy elõre legenerált java kódvázba. Például ha egy OCL kifejezés elõír egy függvény bemeneti paramétereire valamilyen feltételt, akkor az ennek megfelelõ ellenõrzõ java kódot be kell illeszteni a programvázban a függvény elejére. A java programváz elõállításához felhasználható az XSLT (eXtensible Stylesheet Language Transformation) technológia, amely segítségével XML dokumentumok stíluslapok segítségével átalakíthatóak a kívánt alakra.