Objektum perzisztencia megvalósítását támogató kódgenerátor

Önálló labor 2002/2003 őszi félév
Kovács Péter Tamás (kovacsp@sch.bme.hu)
Konzulens: Pintér Gergely

A feladat egy olyan kódgenerátor elkészítése volt, amely képes UML modellező eszközök által exportált XML modell alapján olyan forráskódot generálni, amely a modellben szereplő osztályok vázát képezi, továbbá támogatja az objektumok perszisztens megvalósítását (attribútumok, kapcsolatok elmentése, betöltése, lehetőleg minél átlátszóbb módon). Cél volt, hogy a megvalósítás ne csak egy célnyelvet támogasson, továbbá az, hogy a megvalósítás tükrözze az UML metamodelljének szerkezetét (ez egyszerűsíti is a megvalósítást, mint később látni fogjuk). Egy ilyen kódgenerátor azért nyújthat hatalmas segítséget a szoftverfejlesztés során, mert nem csupán osztályvázakat gyárt (erre a modellezőeszközök is képesek), hanem az osztályhoz illeszkedő, perszisztenciát megvalósító kódot is automatikusan generálja, így nem kell ezt kézzel elkészíteni. Ezzel nem csak a hosszú, monoton munkától, de számos hibalehetőségtől is megkíméli a fejlesztőt.

A modellező eszköz a modellt képes XML formátumban (az XMI szabványnak megfelelően) a rendelkezésünkre bocsátani, a program ezt olvassa be és generálja a kódot. A kódnak két lényeges részét különböztethetjük meg: az osztály vázat és a perszisztencia rutinokat. Az osztály váz tartalmazza az adattagokat, az asszociációkat és a függvényeket, továbbá alapértékkel rendelkező attribútumok esetén konstruktort is, amelyben megkapják értéküket. A perszisztencia rutinok valósítják meg az attribútumok és asszociációk mentését és betöltését XML formátumban.
Az osztálymodell egy kis bővítésen megy keresztül a kódgenerálás során, ugyanis a végleges kódban minden perszisztens osztály a Persistent nevű absztrakt alaposztályból származik, ami biztosítja azt, hogy rendelkezni fognak xmlLoad() és xmlSave() nevű függvényekkel, vagyis képesek lesznek önmagukat betölteni ill. elmenteni. Ezen kívül egy Manager objektum is megjelenik a modellben, amelynek feladatai a tárolt kapcsolatok XML azonosítóinak feloldása (pointerekkel való összerendelése), a mentés koordinálása, valamint egyedi objektum azonosítók kiosztása.
Mint korábban említettem, a belső megvalósítás illeszkedik az UML metamodelljéhez. A kódgenerátorban megtalálható a ModelElement osztály, amely olyan tulajdonságokkal rendelkezik, amelyekkel minden, a modellben szereplő elemnek rendelkeznie kell (név, láthatóság, XMI azonosító, dokumentáció), valamint rendelkezik egy virtuális kiir() függvénnyel is, amely biztosítja, hogy minden belőle származó elem képes lesz kiírni magát forráskódként. Belőle származnak a Paraméter, az Attribútum, a Függvény, és az Osztály osztályok. Ezek mind tovább bővítik a tulajdonságok körét, például egy attribútum típussal, számossággal és esetleg kezdőértékkel is rendelkezik. Ezen osztályok között tartalmazási viszonyok is vannak, mert pl. egy függvény objektum tartalmazza a paramétereinek megfelelő osztályokat, egy Osztály objektum pedig az attribútumainak és függvényeinek megfelelő osztályokat. Minden osztály megfelelően felüldefiniálja a kiir() függvényt, amelyben az esetleges tartalmazott osztályok kiir() függvényeit is meghívja. Az XMI-ból való betöltés is hasonlóan működik, a betöltést csak a Modell objektumra kell kiadni (ez tartalmazza az összes a modellben szereplő elemet), majd ha ez a betöltés során talál egy osztályt, akkor erre meghívja az osztály betöltőfüggvényét, az később az attribútumét és a függvényét, és így tovább.
A kódgenerálás során az UML adattípusokat le kell képezni a célnyelv adattípusaira. Az elemi adattípusoknak van megfelelőjük a legtöbb nyelvben, így ezeket egyszerűen meg lehet feleltetni. Az összetett adattípusok esetén (dátum, pénz, stb..) problémát jelent az, hogy nincs természetes megfeleltetés mint az előbbi esetben, továbbá nem tudhatjuk, hogy a kódgenerátort használó fejlesztő hogyan akarja ezeket használni, tehát nyitott megoldásra van szükség. A megoldást az jelenti, hogy az összetett adattípusokat osztályokra képezzük le, amelyeknek a Persistent alaposztályból kell származniuk és meg kell valósítaniuk a szükséges rutinokat. A kódgenerátor pedig olyan kódot generál, ami a megfelelő helyen meg is hívja ezeket a függvényeket, így ezek is elmentésre kerülnek.
A kapcsolatok megvalósítása során egy komolyabb beavatkozásra volt szükség a megszokott pointeres ábrázoláshoz képest. A pointerek helyett bevezettük a pointer objektumot, amely tartalmazza a hivatkozott objektum azonosítóját, és hivatkozás esetén be is tölti azt, majd visszaadja a rá mutató pointert. Ha a célnyelv C++, akkor szerencsére láthatatlanná tehetjük ezt a mechanizmust, mivel a -> operátor átdefiniálásával megoldhatjuk azt, hogy a programozó (szintaktikailag) ugyanúgy használhassa a pointerobjektumot, mintha egy pointert használna.

Jelenleg a kódgenerátor működőképes, egy XML modellből képes a fent leírt tulajdonságokkal rendelkező forráskód generálására. Ennek ellenére még tovább fejleszthető. Céljaim között szerepel az UML teljes eszközkészletének megvalósítása, egy lehetőleg még átlátszóbb felület létrehozása, DTD generálás, és MOF metamodellek feldolgozására való felkészítés. Ezek lesznek a következő félév feladatai.