Modellező eszközök és kódgenerálási módszerek

Hibatűrő Rendszerek Kutatócsoport

2017

Tartalomjegyzék

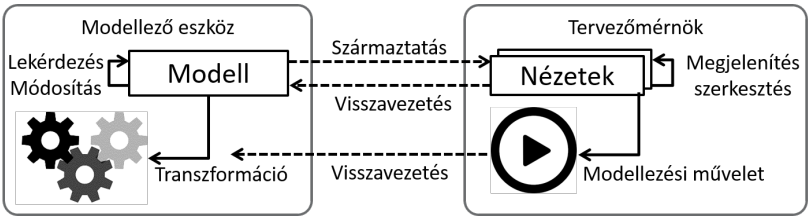
Bevezetés

A fejezet célja, hogy megismertessük a korszerű modellező eszközökkel és kódgenerálási technikákkal kapcsolatos fogalmakat, alapvető felépítésüket és működési elvüket. A fejezet gyakorlatias megközelítésben tárgyalja a témát, minden pontban bemutatjuk, hogyan és munkával készíthetünk saját, vagy egészíthetünk ki meglévő modellező eszközt.

1. Modellező eszközök felépítése

A modellező eszközök célja, hogy különböző modellezési nyelvekhez szerkesztőfelületet nyújtson, és a modellekre épülő automatizált műveletekkel támogassa a fejlesztési folyamatot.

A *modelező eszköz* és a felhasználó *tervező mérnök* közötti interakciókat az az 1. ábra szemlélteti. A modellező eszköz alapvető feladata, hogy karbantartson egy *modellt*, és *lekérdezés (olvasási, vagy keresési)* és *módosítási (beszúrás, törlés)* műveletekkel láthatóvá és szerkeszthetővé tegye azt a *tervezőmérnöknek*. Ezért a tervezőeszköz különböző *nézeteket származtat* a modellekből, amit a mérnök *megjeleníthet* és meghatározott *szerkesztési műveletekkel* változtathat. A modellező eszköz feladata, hogy ezeket a módosításokat visszavezesse a modellbe. Egy modellhez több nézet is tartozhat, ami különböző részleteit emeli ki a modellnek.



1. ábra. Modellező eszközökök alapvető felépítése

Az elészült vagy akár félkész modelleken a tervezőmérnök különböző *automatizált műveleteket* kezdeményezhet (tervezési szabályok ellenőrzése, kódgenerálás, modell refactorálása) ami hatására *automatikus transzformációk* hajtódnak végre a modellen. A transzformáció eredménye lehet jelentés (például hibajelentés a tervezési szabályok alapján), újonnan létrehozott dokumentum (például forráskód), vagy egy módosított modell. A modellezési műveleteket egy fejlett keretrendszernszer magától is elindíthatja fejlesztés közben (például minden mentéskor lefut a tervezési szabályok ellenőrzése), de érdemes ezeket külön meghívhatónak is meghagyni.

Példa. Vegyünk egy a Yakindu tervezőeszközből: a tervezőeszközben megnyithatjuk a `.sct` kiterjesztésű fájlokat, melyeket beolvasva elkészíti a program a modell belső adatreprezentációját. Ezután az eszköz készít és megjelenít egy olyan nézetet, amelyben az állapotokat négyzetekkel, az állapotátmeneteket nyilakkal, a triggereket, őrfeltételeket és akciókat pedig szövegesen ábrázolja. A grafikus nézethez szerkesztőfelületet is tartozik, amin keresztül a tervezőmérnök módosíthatja a modellt. A Yakindu tervezőeszköz fontos tulajdonsága, hogy szigorúan ellenőrzi a modelleket és szabályozza modellmódosítások körét. Emiatt a fejlesztés során tipikusan helyes modelleket finomíthatunk egy újfent helyes modellé (ellentétben a forráskóddal, ahol nem ritka hogy órák után fordul újra a program).
Egy yakindu modellből (egy `.sgen` kiterjesztésű generátor modell segítségével) kódot is generálhatunk, azaz a modellezésért cserébe komoly fejlesztési feladatokat automatikusan elvégezhetünk. Yakindu esetén egy 10 állapotból álló rendszer is 3000 sor java kódot eredményezhet.

2. Modellek ellenőrzése és ábrázolása

Ahogy korábban említettük, a helyes modelleket szigorú tervezési szabályok határozzák meg: egy megszabott elemkészlet használva tehetjük össze a modelleket, miközben több szabály be kell tartaniuk: egy fejlett modellezési környezetben a véletlenszerűen összetett diagramok és szövegek szinte mindig hibássak lesznek. Ezeket a szabályokat *szintaxisnak* nevezzük.

Definíció. *Szintaxisnak* nevezzük a modellekkel szemben támasztott szabályokat. Ha egy modell teljesíti ezeket a szabályokat, akkor azt mondjuk rá hogy szintaktikailag helyes. Amikor modellekről beszélünk, általában szintaktikailag helyes modellekre gondolunk.

A modellek elemzése és szerkesztése során külön kezeljük a modellek megjelenítésével és nézeteivel kapcsolatos részleteket a mögöttes logikai felépítéstől. A konkrét ábrázolási módot érintő szabályokat (például állapotgép diagramokban az állapotokat négyzettel ábrázoljuk, a `var` kulcsszóval vezetünk be változókat, a szorzásnak magasabb a precedenciája mint az összeadásnak…) *konkrét szintaxisnak* fogjuk nevezni, míg a lényegi mögöttes logikai szerkezetet vizsgáló szabályokat (tranzíciók csak állapotok között mehetnek, tranzíciókon keresztül minden állapotnak elérhetőnek kell lennie) *absztrakt szintaxisnak* mondjuk.

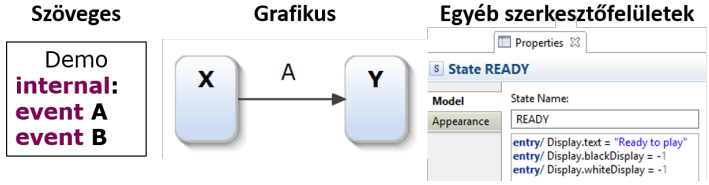
Definíció. *Konkrét szintaxisnak* nevezzük a modell ábrázolásával kapcsolatos szabályokat (színek, alakzatok, kulcsszavak, precedencia, kommentelési szabály). Megkülönböztetünk szöveges és és grafikus szintaxist.

Amennyiben egy modell megfelel a konkrét szintaxisnak, a tervezőeszköz felépíti annak logikai strukturáját. A folyamat hasonlít arra, mint ha strukturálisan modelleznénk magukat a modelleket, elhagyván azokból a konkrét megjelenítésből fakadó részleteket (például a koordinátája a modellelemeknek). Ennek a folyamatnak a kimenetele egy olyan struktúramodell lesz, amiben a lényegi elemek szerepelnek.

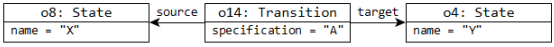
Definíció. *Absztrakt szintaxisnak* nevezzük a modellek logikai felépítésére vonatkozó szabályokat, melyek függetlenek minden megjelenítéssel kapcsolatos részlettől. Ezen kívül absztrakt szintaxis modellnek nevezzük magát a logikai vagy számítógépes reprezentációt is.

A definíciók szerint absztrakt szintaxis alatt egyaránt értjük a szabályokat és modelleket is. Ez azért van, mert a konkrét és absztrakt szintaxis ellenőrzésével együtt szokott előállni a mögöttes modell is.

Példa. Yakinduban



2. ábra. Állapotgép konkrét szintaxisok Yakindu tervezőeszközben



3. ábra. Kétállapotú állapotgép absztrakt szintaxisa

Hivatkozások

Tárgymutató