SDL–UML társulás: UML2.0

MEDVE ANNA

Pannon Egyetem, Műszaki Informatikai Kar, Információs Rendszerek Tanszék, Veszprém
medve@almos.vein.hu

Kulcsszavak: SDL, UML, modellezés

Az egyre összetettebb informatikai rendszerek szoftverfolyamatában elfogadottá vált a tervezés és a modellezés fázisok fontossága. Mindez meghatározta a szoftverfolyamat kezdeti szakaszait támogató nyelvek fejlődését. Az eredetileg dokumentálásra szánt specifikáló és leíró nyelvek alkalmazását napjainkban a validáló, verifikáló és szimulációs eszközök tárháza segíti. Az automatikus tesztgenerálás és szimulációs módszerek lehetővé teszik a hibák felfedezését már a fejlesztés korai stádiumában. Ugyanakkor, a támogató eszközök fejlesztéséhez szükséges, hogy a támogatott nyelv kellően kifejező és egyértelműen definiált legyen. Az SDL (Specification and Description Language) és az UML (Unified Modelling Language) a legelterjedtebben használt modellező nyelvek az iparban. Az SDL kezdetektől, az UML a 2. verziójától formális nyelv. Cikkünkben röviden bemutatjuk mindkét nyelvet, komplemens jellegüket ismertetve alkalmazásuk variálhatóságát, továbbá a távközlés fejlődéséből eredő azon jellemzőiket, amelyek a két nyelv konvergenciájával új modellezési irányzatok magvalósításához vezettek.

1. Bevezetés

„Navigare necesse est…"

Az egyre bonyolultabb rendszerek megvalósításához a szoftverrendszerek fejlesztésben bevált gyakorlat a dolgok összetettségére a komponensek fejlesztése, a történések összetettségére pedig a komponensek kommunikációs alapokra helyezése. A szoftverrendszerek fejlesztésében manapság vitathatatlanul bekövetkezett a modern fizika történetében tapasztalt formalizálás szükségessége; elkülönített technikák és módszerek kellenek a dolgok és összefüggéseik elvonatkoztatására, valamint technológiák a dolgok megvalósítására. Ennek szellemét hordozza alapjaiban úgy az UML mint az SDL, előbbi a dolgok és összefüggéseik, utóbbi a dolgok és kapcsolatukban való történéseik leírására alkalmasabb.

A formális nyelvek Z. szabványcsaládját [2] az ITU (International Telecommunication Union – Telecommunications Standardization Sector) a kommunikáló protokollok és kommunikációs szolgáltatások [1] modellezésére fejlesztette ki és bővíti folyamatosan. A szabványcsalád több eleme az MDA képességű specifikáló és leíró nyelv, az SDL köré csoportosul. Az UML az egységesített modellező nyelv, az OMG [6] fejlesztésében szabványosított a szoftverrendszerek fejlesztésének elemzés és tervezés szakaszára, főként az információs rendszerek fejlesztésében terjedt el.

Az OMG és az ITU-T munkacsoportjai létrehozták az UML2.0 szabványt az UML, az SDL és az MSC szabványainak összetársításával. Így az UML2.0 az MDA modellezést gyakorlattá tevő általános modellező nyelv szabványa lett.

A szabványok nem kínálnak módszertant a nyelvek alkalmazására, ezért célszerű figyelembe venni a társulásba beemelt nyelvek fejlődésének korábbi szakaszából általánosan és az egyes célterületek problémáira nyert technológiákat és technikákat. Az alábbiakban röviden bemutatjuk az SDL és UML nyelveket. A komplemens jellegüket ismertetve mutatjuk meg alkalmazásuk variálhatóságát, a távközlés fejlődésének hatásait a nyelvek alkalmasságának oksági összefüggéseiben hivatkozzuk, mindezek előtt a modellezés természetének és ebben a formális nyelvek szerepének felvillantásával a két nyelv növekvő szerepét hangsúlyozzuk.

2. A modellezés természete a szoftverfolyamatban

„A számítástechnika nincs szorosabb kapcsolatban a számítógépekkel, mint a csillagászat a távcsövekkel."
Dijkstra

A modellezés alkotó tevékenység, amelynek elengedhetetlen tartozéka az alkotást lehetővé tevő módszerek és eszközök rendszere. A modellelemek átnézése, egybevetése, kombinálása közben mentális folyamatokkal új felismerésekhez jutunk, amelyek kifejezéséhez szükséges formalizmus nélkül nem teljes az ábrázolás. A modell jósága ily módon függ *a modellező nyelv kifejezőerejétől.*

A szoftverfolyamat egyik fontos követelménye a piackövető szoftvertermékek előállítása, amely teljesülni látszik a dolgok és összefüggéseik, valamint a megvalósításukhoz és működtetésükhöz szükséges technológiák elválasztásával. A szoftver közvetve vagy közvetetten piaci termékké változott, sorozatgyártása előtt prototípust célszerű megadni: *modellezéssel és automatizálással,* mennél gyorsabban és olcsóbban.

A modellezés előnye a szoftverfejlesztésben a szoftver azon tulajdonságából adódik, hogy a szoftver természeti törvényekkel nem határolható be, ezért a működéskori valóságra vonatkoztatott hipotéziseket és pre-

dikciókat is modellezéssel állítjuk elő, az úgynevezett követelménymodellt, majd ezek transzformációi adják a megvalósítani kívánt rendszermodellt. Manapság elvárás szintű a fejlesztőkörnyezetektől az automatizált modell-transzformációs támogatás.

A modell megvalósításának automatizálhatósága függ attól, mennyire jól definiált a modellező nyelv,. A jól definiált modellező nyelv a formális nyelv, amellyel a modellezés eredménye a formális specifikáció. A formális specifikáció a bemenete megfelelő társítású transzformációs eszköztárnak, amellyel automatizáljuk a szoftverfolyamat részeit [19,20,3]. A modelltranszformációk helyességéről meg kell győződni a hibátlannak tudott rendszermodellből generált kód előállítása előtt.

Az automatizált modellfejlesztésnek az egyik eszköze a formális modell, irányzata az MDA [21], a Model Driven Arcitecture, célja a hordozható, újrafelhasználható, platformfüggetlen szoftvertermék előállítása. A technológia-függetlenséghez el kell tudni választani a szoftvertermékre vonatkozó valós világbeli generikus tulajdonságokat a specifikus tulajdonságoktól már a modellalkotás szintjén.

Ily módon kell a platformfüggetlen modell (PIM), melyből további módszerekkel és eszközökkel állítható elő az a platformfüggő modell (PSM), amely tartalmazza a szükséges technológiai információkat a modell megvalósításához a kijelölt platformon. A PIM szerepei (és az ezt támogató modellező eszközök) a formális specifikációból generálható kóddal váltak jelentőssé az összetett piaci változások követésére.

3. Az SDL specifikáló és leíró nyelv

Az *SDL (Specification and Description Language)* nyelvet az ITU-T a Z.100 szabvány [10] ajánlásaiban definiálták komplex rendszerek specifikálására. A rendszer működésének leírását a konkurens módon diszkrét jelekkel kommunikáló, valósidejű és interaktív folyamatok eseményvezérelt ábrázolásai alkotják.

Az SDL fejlődése a távközlés fejlődésében történt [10-13], létrehozása egy 1968-as ITU tanulmányból indul ki a programvezérlésű kapcsolórendszerek kezelésére vonatkozóan, aminek eredményeként 1972-ben egyetértés születik arról, hogy nyelvek szükségesek a gépek és berendezések interakcióinak leírására és programozására. Az első SDL szabvány 1976-ban az alap grafikus leírónyelv, majd négyévenként további fejlesztéseket jelentettek meg. A Z.100-as szabvány jelenleg az SDL-2000 verziót jelenti, amelyben növelték az objektumorientáltságot, valamint bevezették az ágens-elvet, viszont ez utolsó verzió absztrakt gépe még nincs megvalósítva, így az elterjedt fejlesztőkörnyezetek az SDL-96 szabványát támogatják.

Az SDL nyelvet 1996 óta használják a távközlési iparon kívül is, elsősorban az orvosi felszerelések iparágában, az autó- és repülőgépiparban. Az SDL eszközök piaca 1996-2000 között jelentősen növekedett. A fejlesztőkörnyezetek generálnak programozási nyelvekre forráskódokat (általában C/C++, Java), amelyeket be lehet szerkeszteni a valósidejű rendszerek termékgyártásába. Az SDL alkalmazását megkönnyíti, hogy az egymásnak kölcsönösen megfeleltethető, grafikus és szöveges nyelvi implementációi, az SDL/GR és az SDL/PR vannak használatban specifikálásra, tervezésre, dokumentálásra, megvalósításra. A megvalósítás alapja a hibátlan formális specifikáció.

Az SDL matematikai alapja a kiterjesztett véges automata (EFSM) modell, amely a rendszer működését gerjesztés-válasz módon határozza meg a rendszert alkotó kommunikáló automaták állapotátmeneteinek halmazán. A rendszer több egymással és a rendszerkörnyezettel kommunikáló automatából áll, ahol a kommunikáció jelekkel történik a FIFO elven működő csatornákon és jelútakon. Az SDL rendszer struktúráját a kommunikáló részegységek, a rendszer dinamikus viselkedését a kommunikáló automaták állapotátmenet-szekvenciái adják.

Az ábrázolandó rendszer szerkezetét hierarchikusan a *rendszer alatti blokk, processz és eljárás* egységek egymásba ágyazásai alkotják. A processz a hierarchia levélszintjén maga a kommunikáló automata, az eljárás algoritmusokat és csoportosított automata állapotátmeneteket tartalmazhat. Ezáltal bármely célú változtatás a faszerkezetben egyértelműsíthető.

Az SDL specifikációk jóságához szükséges az *MSC (Message Sequence Charts)* [14] nyelv használata, melynek szerkezetei segítik a rendszerentitások hierarchiába szervezését, valamint a dinamikus viselkedés formális analízisét. Az MSC teljes szabványát beemelték az UML 2.0 verziójába.

Az adatok ábrázolására az SDL-ben a beépített típusgeneráló lehetőséget ad bármely adat leírására, előnyös a beépített *ASN.1(Abstract Syntax Notation One)* adatleíró nyelvet alkalmazni [15]. Ennek nagy előnye a nyílt rendszerek ábrázolásakor érvényesítődik az együttműködő képesség fokozásával.

Az SDL fő szabványa a Z.100, további szabványokkal terjesztik ki a nyelvet a komponensalapú és platformfüggetlen fejlesztés támogatására. A Z.105-ös szabvány megadja az ASN.1 modulok kapcsolását az SDL adatleírásokba direkt módon, a Z.107-es szabványban rögzítik az ASN.1 beágyazását az SDL nyelvbe. A Z. 109-es szabvány definiálja az SDL-nek megfelelő UML profilt, így hozzáadja az UML elv szerinti ábrázolást. A Z.120-as szabványcsalád az MSC nyelvet hasonlóan definiálja.

Az *SDL fejlesztőkörnyezetek* legtöbbjének közös funkciói a grafikus szerkesztő, a szöveges és grafikus konverzió, a statikus elemző, a kódgeneráló, a szimuláló és validáló dinamikus elemzés, az MSC-vel kombinált támogatás [4].

Az SDL nyelvről ismertetést angolul az [4] fórumán, magyarul a Híradástechnika 2005/10. számában olvashattunk [5]. Az SDL kutatásának hazai képviselőinek eredményei elsősorban a specifikáció-bázisú tesztelés és validálás terén az automatikus tesztsor generálásra és szelektálásra adott számos algoritmus [8], az SDL

alkalmazása a hardver-szoftver együttes tervezésére [7], a konformancia teszteléssel összefüggésben [1], valamint formális módszertani összefüggésben [3].

4. Az UML egységes modellező nyelv

Az *UML (Unified Modelling Language)* nyelvben [21] a rendszermodell többféle modelltípusból tevődik össze, amelyek kötelező részletezettsége függ a modellezés céljától. Szimulációhoz vagy kódgeneráláshoz teljes és konzisztens rendszermodellt kell szerkeszteni.

Az egyes modelltípusokat különböző diagramokkal ábrázoljuk, amelyek közötti szemantikai összefüggések jóságát és helyességét modellverifikálással és validálással kapjuk meg. Az UML rendszermodellt alkotó modelltípusok és a modelltípus ábrázolásához szükséges diagramtípusok a következők:

- *A rendszerhasználat eseteinek modellezése a használati eset-, csomag- és osztály diagramokkal.*
  Ebben a modellben döntjük el a rendszer, alrendszer, komponens vagy osztály környezetét. Használati eset diagramban a kapcsolatok megadásával a használati esetek közötti általánosítást, magában foglalást, kiterjesztést, függőséget adjuk meg, míg a kollaborációval illusztrálhatjuk, hogy hányfajta különböző kombinációja lehet a szereplő interakcióinak más használati esetekkel vagy a rendszer többi részeivel (a tárgyakkal).
- *A rendszer objektumainak és ezek strukturális kapcsolódásainak modellezése a csomag-, osztály- és architektúra (kompozíciós) diagramokkal.*
  Ebben a modellben a nagy rendszerek leírásánál csomagokat szerkesztünk miután a rendszer elemeit azonosítottuk és osztályba foglaltuk. Egy osztálydiagram modell elemei az osztály, attribútum, operáció, (port, interfész, jel, jellista, időzítő csak a 2.0 verzióban), adattípus, választás, állapotgép és kapcsolat. Azokat az objektumokat, amelyek ugyanazt a tulajdonságot, viselkedést, és más objektumokkal ugyanazt a kapcsolatot mutatják, egybefogjuk, és annak az osztálynak az objektumaként modellezzük. Egy osztálydiagram az objektumtípusok közötti strukturális, és viselkedési kapcsolatokat mutatja meg, valamint a kompozíciókkal az aktív osztály kommunikációs jellemzőit strukturáljuk.
- *Rendszerviselkedések modellezése a tevékenység-, szekvencia- és állapotgép diagramokkal.*
  Ennek a modellezésnek a feladata, hogy megmutassa a viselkedést azzal, hogy ezt kis viselkedésegységekre bontja, leírva a vezérlő és adatfolyamokat ezen egységek között.
- *A rendszer elosztásának (komponenseinek) modellezése a komponesdiagrammal.*
  A komponensmodellezés feladata, hogy azonosítsuk a rendszer komponenseit, és modellezzük az interfészeiket, és kapcsolatukat. A rendszer statikus nézetét mutatja meg, a komponenseken, a realizált interfészeken, és a szükséges interfészeken keresztül. Itt csak a komponens fogalmát kell ismertessük, ami nem más, mint a rendszer egy jól elkülöníthető része, ami jól leírható szolgáltatásokat nyújt. Az UML nem nagyon különbözteti meg a komponenset az osztálytól, mindent amit megtehetünk egy osztállyal, azt megtehetjük a komponenssel is. Egy komponensdiagramban kettő vagy több elem között lehet társítás, aggregáció, kompozíció, függőség, általánosítás, implementálás.

Az *UML szabványai* az UML1.x és a kommunikáló automata alapokra helyezett UML 2.0 filozófiájában eltérnek egymástól. Az UML2.0 szabvány kiindulópontja az ITU-T Z.109-es szabványa, amely definiálja az SDL-nek megfelelő UML profilt. Az UML és SDL házasságából a legnagyobb haszon a mindkét oldalról beemelt hozzátartozók fejlesztés-támogató eszköztára.

Az UML 2.0 formális nyelv, amely alkalmas a formális specifikáció előállítására.

Egyszerűsítve mondhatjuk, hogy az UML2.0 szekvencia diagramja lett a teljes MSC-2000 nyelv, valamint az UML2.0 állapotgép diagramja lett az SDL processzdiagramja, az osztálydiagram elemei közé létrehozták az kommunikáció egyértelműsítéséhez szükséges port, interfész, jel, jellista, időzítés elemeket. Mindez maga után

1. táblázat

Az SDL elemek UML megfelelői

SDL	*UML*
Rendszerdiagram	Osztálydiagram
Blokkdiagramban	Architektúra diagram
Processzleírás	Állapotdiagram
Rendszer	Csomag az osztálydiagramon, vagy tárgy a Használati eset diagramon
Blokk	Aktív osztály
Processz	Egy aktív osztály operációja
Csatorna	Konnektor az Osztálydiagramon, vagy Architektúra diagramon
Jelút	Konnektor az Osztálydiagramon, vagy Architektúra diagramon
Környezet	A Használati eset diagram szereplői

„húzta” a TTCN-3 [16] szabványát az ITU berkeiből. A nyereség azonnal adódott: a validáló és szimulációs eszközök, a tesztelés támogatása, az automatikus kódgenerálás.

Az UM2.0 2003/2004-re megújult négy nagy strukturális részei láttatják alkalmazhatóságának sokrétűségét:

– *„Infrastructure”:* az UML nyelvi specifikációjának alapját adó szabványokat azonosítja,
– *„Superstructure”:* az UML nyelvi specifikációja,
– *„Diagram Interchange Model”:* egyfajta diagramkapcsolati interfész a többféle nyelv és fejlesztőkörnyezet közötti átjárásra,
– Az *OCL (Object Constraint Language)* objektumspecifikációs nyelv, amely utasításszerű specifikációs eszköze az UML megvalósításokhoz szükséges pontosabb feltételrendszer megadásának. Az OCL által az UML2.0 követelménytervezés leírása algoritmikussá válik, pontosított adatbázistervet tudunk származtatni.

Az UML profilok sajátja a különböző fő iparágak számára beépített UML sztereotípusokkal megadott szakterületi fogalmak és műveletek rendszere, amelyben az ábrázoló eszköztárhoz kifejlesztik az adott szakterületre jellemző, követelmény-elemzést támogató fejlesztőkörnyezetet. A szakterület-specifikus fejlesztéstámogató eszköztárakkal az UML nyelv az úgynevezett domén-specifikus nyelvek felé tolódik el, általában új nevet is kap az átszabott nyelv. UML profilok jellemzően az üzleti modellezés, az autóipar, repülőgépipar, a monitorozó- és vezérlőrendszerek iparában születnek. Bővebben a [21], magyarul a [22] irodalmak tájékoztatnak.

5. Az SDL és az UML egymás komplemensei

Mindkét nyelv az inkrementális fejlesztést nyújtja – nem dobunk el köztes ábrázolásokat – az SDL és UML nyelvek diagramszemlélete közötti nagy különbség ellenére mindkettőben jól hangolható a spirális fejlesztés, ami nagyfokú eloszthatóságot jelent időben és földrajzi egységben. Megjegyezzük, hogy ehhez vannak a modellező nyelvben nyelvi eszközök, a szoftverfolyamat módszertanát nem, csak eszközeit adja az UML, ugyanakkor az SDL nyelv kötöttségei diktálják a fejlesztés módszertanát is.

Az SDL és UML elemek megfeleltetését az *1. táblázatban,* az UML és SDL elemek megfeleltetését pedig a *2.táblázatban* mutatjuk be.

Az UML a jelek útjai között nem tesz olyan megkülönböztetést, mint például az SDL, ami jelutat és csa-

UML	*SDL*
Használati eset (Use case) diagram	Rendszerdiagram
Szekvencia diagram	MSC leírás, az SDL állapotátmenetekből generálható
Osztálydiagram	Rendszerdiagram
Arhitektúra diagram, Kompozíció	Blokkdiagramban
Állapotdiagram , Állapotgép	Processz leírás
Szövegdiagram	Nincs konkrét megfelelője
Használati eset	Blokkok vagy processzek halmaza
Szereplő	Környezet
Tárgy	Rendszer
Csomag	Blokk
Osztály	Blokk
Operáció	Blokk egy processze
Aktív osztály	Blokk egy processze
Port	Konnektor
Interfész	Csatorna irányonként,
Jel	Jel
Rész	Processz
Konnektor	Jelút
Viselkedési port	Jelút

2. táblázat

Az UML elemek SDL megfelelelői

tornát használ, ugyanakkor az UML, a bemenő és kimenő interfészekbe különítéssel egyértelműsíti az SDL-beli csatorna irányoknak megfelelő jelcsoportosítást, többletfeltételekkel megerősítve elérésüket az egyes interfész-csoportosító portokkal, ilymódon a tesztesetek és ellenőrzésük egyértelműbbé válik. Az UML interfészek definiálása bonyolultabb kommunikáció leírását teszi egyértelművé azáltal, hogy osztályszinten tudjuk megadni a jellistát és az operációkat a küldő/fogadó osztályra.

Ezeket az interfészeket az osztályok blokkjaihoz kapcsoljuk, így ezek együtt felfoghatók az SDL-beli csatornáknak, vagy jelutaknak, és azok egyfajta kiterjesztéseként, mintha csatorna-alrendszert deklarálnánk SDL-ben. A jelek halmazát mindkét nyelvben a jellistával (signallist) definiálhatjuk, de UML-ben lehetőség van arra is, hogy összefogott attribútumok halmazát egy jelnek tekintsünk.

Az SDL rendszerdiagram környezet fogalma is másképpen jelenik meg az UML-ben azáltal, hogy a struktúrák ábrázolása nem szigorúan hierarchikus, és a társítások foka ábrázolható. Ezért látványosabb az alsóbb szintű struktúrák konkrét környezete, azaz viselkedés szempontból is konkrétan megnevezhetők a környezeti elemek. Az SDL-ben alkalmazott automata egyedi azonosító az UML-ben típusleírással és feltételekkel megerősítve új eszköz lett az eseményvezérelt folyamatok komponenseinek fejlesztésére.

Összehasonlítva a két nyelvet láthatjuk, hogy nagy, komplex rendszereknél kifizetődőbb UML-ben modellezni, mint SDL-ben. Mivel eléggé hasonló módon lehet létrehozni az UML osztály- és állapotdiagramját és az SDL rendszer-, blokk-, és proceszdiagramját, kijelenthetjük, hogy azoknak is megéri áttérni erre a nyelvre, akik eddig SDL-ben fejlesztettek. Az UML nyelvben abból a szempontból is kifizetődőbb a fejlesztés, hogy rendszeralternatívákat a fejlesztés bármely fázisában tudunk viszonylag kevés munkaráfordítással megadni a megrendelők felé, ami SDL esetén már újratervezést jelent, ellenben könnyebb a feladat a hierarchiából adódóan.

SDL-ben fejleszteni kifizetődőbb a protokollfejlesztésekben, ahol a szabványokban rögzített az ellenőrzött követelményterv, amely, általában MSC-, gyakran SDL specifikációkat is tartalmaz az egyértelmű olvasatukhoz. Arra a kérdésre keresve a választ, hogy megéri-e esetleg először SDL-ben létrehozni az egyes diagrammokat, majd azt UML-be importálni, kijelenthetjük, hogy nem, kivéve, ha az SDL fejlesztő az áttérés stádiumában, UML módszertani ismeretek hiányában, SDL-ből képezi le az UML modellnézeteket.

A két nyelv között az automatikus átjárás mindkét irányban lehetséges a Telelogic fejlesztőkörnyezetein, manuálisan a diagramok összevetése és együttes használata is gyorsítja a modellezés folyamatait. Az SDL modellek újratervezését érdemes UML-ben megadni: a meglévő SDL rendszer importálásából megkapjuk a modellelemeket és UML-ben fejleszthetjük az újratervezést.

6. Az SDL és az UML konvergenciájának MDA jellegű hozamai

Az UML egységes modellező nyelv rendszerspecifikálásra és architektúra-szintű tervezésre alkalmas, kiterjeszteni a nyelv képességét a szerkesztett modell implementálására a sztereotípusok beépítésével lehetséges. Az UML2.0 szabvány támogatja a modellvezérelt paradigmát (MDA), az UML profilokban az automatizált transzformációkkal és egyéb szakterület-specifikus környezettámogatásokkal. Elvárt gyakorlat a diagramcserék átjárhatóságát biztosítani a fejlesztés különböző szakaszaiban, a különféle fejlesztőkörnyezetek között.

Az SDL komponensek üzenetcsere alapú kommunikációja valamint a komponens elvű fejlesztés eredendően tartalmazza az SDL fejlesztések MDA jellegét. Ennek megfelelő UML2.0 elemek az objektumtervezésben kapnak hangsúlyt az aktív osztályokhoz tartozó attribútumok és a metódusokhoz tartozó interfész-, jel- és idő- osztályok szerepében.

Ugyanis a spirális fejlesztéssel a részletes objektumterv előtti szimuláció kínálja a platformfüggetlen fokozatokat. További kérdéssé válik, hogy a fejlesztőkörnyezetek vizuális (egér-) programozású támogatásai előnyt vagy hátrányt szenvednek-e a részletes objektumterv spiráljaiban. Például a Telelogic Tau G2 2.6-os változatában a részletes objektumterv többlépcsősre bontásával (inkrementális és spirális szoftverfolyamat) előnyt veszítünk a modellelemek készletezéséből nyerhető egérhúzásos technikák terén.

Az ITU-T nyelvek fejlesztőcsoportja System Design Language néven az MDA paradigma mentén határozta meg a Z. szabványcsalád azon elemeit, amelyek a teljes fejlesztési folyamatot támogatják. Az ITU-T szabványkészlete és módszertana az URN-MSC-UML-ASN.1-SDL-TTCN szabványokra épülő inkrementális módszertant ajánlja [2].

A rendszer életciklusaiban alkalmazható formális nyelvek kapcsolatát szemlélteti az *1. ábra,* amelyben megmutatkozik az SDL nyelv kohéziós szerepe az implementációs jellegéből adódóan. Az SDL-Forum és az ITU-T munkacsoportjai dolgoznak a nyelvek egy fejlesztőkörnyezetbe integrálásán.

A Z. 150-153. szabványok *User Requirements Notation (URN)* szabványa két formális nyelvet, a *Use Case Map (UCM)* és *Goal Requirements Language (GRL)* formális nyelveket tartalmazza, rendre a funkcionális követelmények és nem-funkcionális követelmények ábrázolására.

A modellezés folyamatában a lépések az UCM leképezése MSC diagramokra, amelyek többféle módon építhetők be az UML modellbe, (osztály-, szekvencia-, tevékenység diagramként), az SDL modellbe áttranszformálhatók a statikus és dinamikus leírásba. Az ASN.1 deklarációk által többféle módon gyorsul és egyszerűsödik a *Test and Test Control Notation (TTCN-3)* modulok alkalmazása a validálás és teszteset generálás folyamataiban. Az SDL specifikáció az alapja a TTCN tesztgenerálásnak és a forráskód készítésének C++

vagy JavaScript nyelven. A Deployment and Configuration Language (DCL) telepítő és konfigurálás leíró nyelv a kidolgozás folyamatában van, elvei és elemei az objektum (eODL) és interfész leíró (IDL) nyelvekhez kapcsolódnak.

7. Összefoglalás

Az SDL és az UML a legelterjedtebben használt szabványosított modellező nyelv, profiljaikat számos területen alkalmazzák [6].

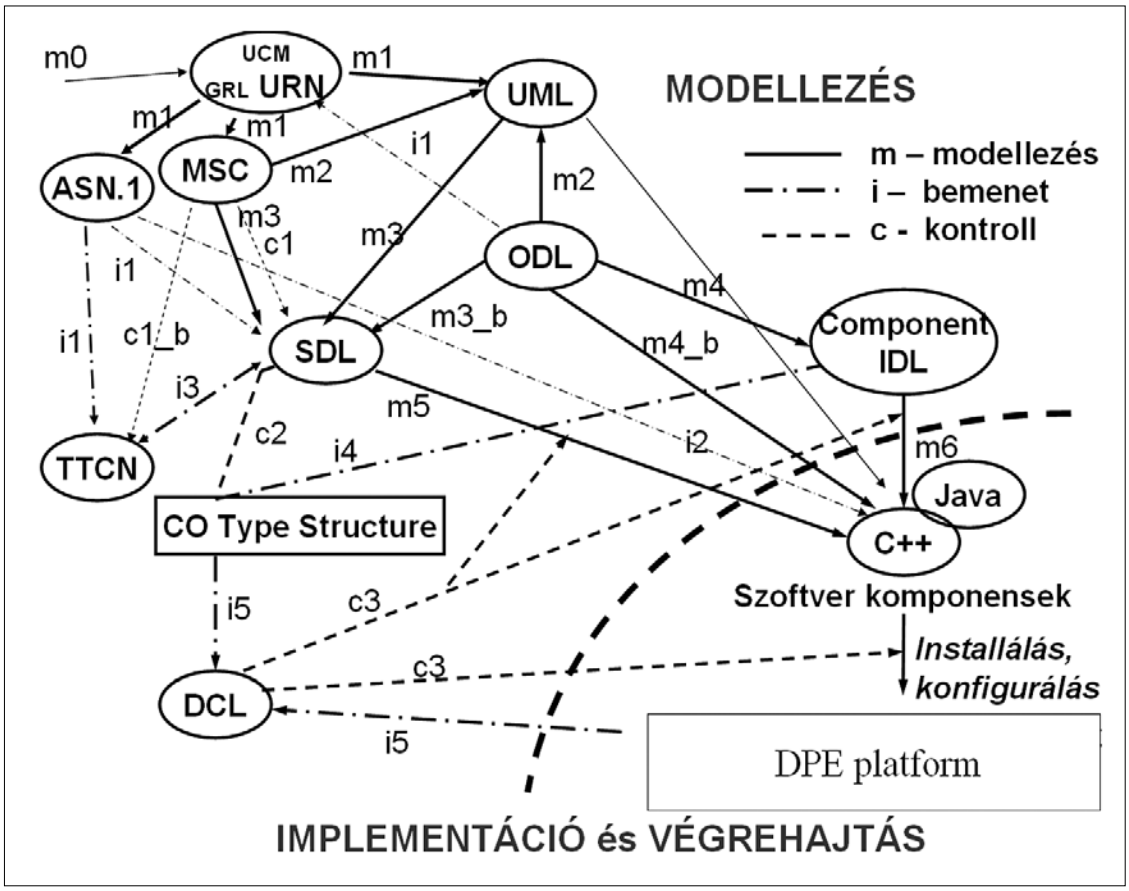
A SDL, a kommunikáló protokollok fejlesztésére létrehozott nyelvként, a távközlés fejlődésének hatására tartalmazza mindazon programozásnyelvi elemeket, amelyek alkalmassá teszik a rétegződés elvén szervezett és együttműködő-képességet fokozó architektúrák megvalósítására. A formálissága és adatdefiníciós eszköztára alkalmassá teszi bármely rendszer modellezésére. Több évtized alatt kifejlesztett validációs és verifikációs technikái beépültek a komponens-elvű szoftverfejlesztés technikáiba. Az UML objektumábrázoló elvein megerősített SDL specifikáló ereje fokozza a modellező nyelv terjedését az ipari hálózatok és beágyazott rendszerek fejlesztésében.

Az UML létrejötte és fejlődése, az OO programozástechnológiák elterjedésének hatására vált szükségessé, fejlődésének fő katalizálója a teljes fejlesztési ciklus lefedésének igénye. Az UML2.0 szabványba az MSC és SDL nyelvek beemelésével az MSC által a dinamikus elemzés, az SDL által az implementálás fázisok támogatása erősödött. Ugyanakkor, az SDL specifikációkra épített technológiák mind beemelhetők a fejlesztési ciklusba, így a távközlés világára kidolgozott tesztelési eljárások és tesztkörnyezetek is, megerősítve az UML nyújtotta követelményvalidálás automatizálásával. Ennek hozadéka a követelménytervezésből derivált tesztelési terv, indirekt módon pedig a szoftverfolyamat költségeinek csökkentése.

30 éves szerepével a távközlés és a modellezés fejlődésében, az SDL specifikáló és leíró nyelve napjainkra az ipari automatizálás egyik modellező eszközévé vált SDL-RT és ezSDL néven, jelenlétével az UML2.0-ban az úgynevezett beágyazott rendszerek fejlesztésére ad technikákat, a hálózattechnológiák és szolgáltatások összefonódásával szerepet kap az üzleti folyamatok és ügymenetek modellezésében is.

Az UML további fejlődését követhetjük az OMG keretében. Az UML modellező nyelv megerősítése, a kommunikáló automatákra alapozott formális specifikációval, felgyorsította a komponensalapú és eseményvezérelt programozástechnológiák alkalmazását, valamint a rendszerfolyamatok platformfüggetlen ábrázolásával az MDA paradigma kutatását és a SysML nyelv fejlesztését.

Az SDL további fejlesztését követhetjük az ITU-T és az SDL Forum keretében megvalósuló bővítésekkel, amelyek a (NGN) következő generációs hálózatok és a beágyazott hálózatok modellezésére súlyozottak, a folyamatok időbeliségének pontosabb ábrázolásával és tesztelésével.



1. ábra

Az SDL kohéziós szerepet kap a távközlő rendszerek fejlesztésére szabványosított ITU-T formális nyelvek és az OMG UML szabvány egymásra épülésében [18]

Köszönetnyilvánítás

Hálával tartozom Dr. Németh Gézának az elmúlt 25 év eszmecseréivel megvilágított szakmai irányvonalakért a programozástechnológiák és hálózattechnológiák világában, az „ezt kellene olvassad” kísérőszavú Ashton-Tate, Wirth, Kernigham-Ritchie, Angszter, Dijskra könyvekért és CEBIT-beszámolókért.

Ugyancsak sok köszönettel adózom többek között Dr. Kozmann György tanításaiért és támogatásáért, valamint a Híradástechnika főszerkesztőjének tanácsaiért és segítségéért a cikk véglegesítésében.

A téma kutatását a PE MIK és az IRIT Toulouse intézmények együttműködésében, a TÉT Alapítvány F-16/04 magyar-francia kormányközi szerződése támogatja.

Irodalom

[1] Tarnay K.
Kommunikációs protokollok modellezése és konformancia vizsgálata.
Doktori értekezés, 1990.

[2] www.itu.int.org/recommendation/zseries/languages

[3] Pataricza A.,
Formális módszerek az informatikában,
Typotex, Budapest 2004.

[4] http://www.sdl-forum.org; www.sdl-forum.org/tools

[5] Medve A.,
SDL – a kommunikációs folyamatmodellek egyik szabványosított implementációs eszköze,
Híradástechnika 2005/10.

[6] OMG.org, ISO.org, ITU.int.org., IEC.org, IEEE.org, CEN.org, WHO.org

[7] Gy. Csopaki, K. J. Turner,
Modelling Digital Logic in SDL.
In Proc. Formal Description Techniques X/Protocol Specification, Testing and Verification XVII,
Chapman and Hall, London, UK, November 1997.

[8] S. Dibuz
Testing protocols,
Application of Formal Description Methods,
Publ. VEAB,Veszprém, chapter 5.; 2001.

[9] www.telelogic.com, SysML.org

[10] ITU-T Recommendation Z.100 (1995)
“Specification and Description Language (SDL)”

[11] ITU-T Recommendation Z.100 (1999)
“Specification and Description Language (SDL)”

[12] ITU-T Recommendation Z.105 (1995)
“SDL Combined with ASN.1 (SDL/ASN.1)”

[13] ITU-T Recommendation Z.109 (1999)
“SDL Combined with UML (SDL/UML)”

[14] ITU-T Recommendation Z.120 (1999),
Message Sequence Chart (MSC)

[15] ITU-T Recommendation X.680 (1994)
“Data Networks and open System communications – OSI networking and system aspect –
Abstract syntax Notation One (ASN.1)”

[16] ITU-T Recommendation X.292 (1998),
Z.140-Z.149 (2003):
“OSI conformance testing methodologhy and framework for protocol Recommendation for ITU-T applications – The Tree and Tabular Combined Notation (TTCN)”

[17] ITU-T Recommendation Z.150-153.(2003),
“User Requirements Notation” (URN),
“Use Case Map” (UCM),”
Goa-Requirements Language” (GRL).

[18] Medve A.,
A formális módszerek szerepe a távközlési szoftverek fejlesztésében,
Networkshop 2001, www.niif.hu/networkshop

[19] D. Latella, I. Majzik, and M. Massink,
Automatic verification of UML statechart diagrams using the SPIN model-checker.
Formal Aspects of Computing, 11(6):637–664, 1999.

[20] D. Varró,
Automated formal verification of visual modeling langauges by model checking.
Software Systems Modelling, 3:85–113, 2004.

[21] www.omg.org/mda ,
www.omg.org/uml

[22] Raffai Mária,
UML 2 Modellező nyelvi kézikönyv,
Objektumtechnológia sorozat 4. kötet,
Palatia Nyomda és Kiadó, 2005.