INFOKOMMUNIKÁCIÓS PROTOKOLLOK

Dibuz Sarolta
a műsz. tud. kandidátusa, PhD, egyetemi docens
Ericsson Magyarország K+F Igazgatóság,
BME Távközlési és Médiainformatikai Tanszék
Sarolta.Dibuz@ericsson.com

Csopaki Gyula
a műsz. tud. kandidátusa, PhD, egyetemi docens
BME Távközlési és Médiainformatikai Tanszék
csopaki@tmit.bme.hu

Mind az üzleti világ, mind a magánélet szférájában kiemelt jelentősége van az információ átvitelének és feldolgozásának. Protokollokkal írjuk le azokat a szabályokat, amelyek szerint az infokommunikációs rendszerek elemei egymással kommunikálnak, információt cserélnek. A jelzési kapcsolatok felépítése és információk átvitele a nemzetközi ajánlások és szabványok által definiált protokollokkal történik, ezáltal biztosítható a világméretű és helyi kommunikáció. Az infokommunikációs rendszerekben számos protokollt valósítanak meg. Különböző gyártó cégek készíthetik az infokommunikációs rendszer elemeit, ezáltal nagyon fontos biztosítani az elemek együttműködését. A protokollok fejlesztése során ezért kiemelt jelentőségű a formális nyelvekkel történő specifikáció, valamint a kifejlesztett és megvalósított távközlési és informatikai szoftverek és protokollok tesztelése. Jelen cikkben áttekintjük a távközlési szoftverek tesztelésének és minőségbiztosításának folyamatát, foglalkozunk a tesztelés szervezési részével és a tesztelés automatizálásával.

1. Bevezetés

Az infokommunikációs rendszerek fontos részét képezik a kommunikációs protokollok. A protokollok határozzák meg azokat a szabályokat, amelyek szerint a rendszer egyes elemei egymással kommunikálnak, információt cserélnek. Tehát a megfelelő protokollokat minden rendszerelemnek ismernie kell. Mivel általában a rendszerek különböző gyártók által készített elemekből épülnek fel, szabványosítják a protokollokat, biztosítva, hogy megértsék egymást a rendszerelemek. A protokollszabványok meghatározzák, milyen üzenetek váltásával folyjon a kommunikáció, és azt is, hogy milyen egy helyes üzenetváltás. A gyártók ezeket a protokollszabványokat valósítják meg a termékeikben. A megfelelő együttműködés természetesen csak akkor lehetséges, ha a protokollmegvalósítások helyesen működnek, tehát nagyon ellenőrizni, hogy megfelelnek-e a szabványoknak.

A következőkben ismertetjük a távközlésben használt protokollok és egyéb szoftverek fejlesztésének és tesztelésének menetét. A harmadik fejezet arról szól, hogy hogyan kell előkészíteni a tesztelést a szoftverfejlesztő projektekben. A negyedik fejezet a tesztelés helyét és szerepét mutatja be a szoftverfejlesztés folyamatában. Az ötödik fejezetben a tesztautomatizálásról szólunk, ami nagymértékben javítja a tesztelés hatékonyságát. Végül a tesz-

telési folyamatban széleskörűen használt nyelvet, a TTCN-3-at (Testing and Test Control Notation), mely hatékonyan támogatja a tesztelési folyamatot, mutatjuk be röviden. A protokollmegvalósítások az esetek többségében nem automatikusan készülnek a specifikációból, ilyenkor még fontosabb a helyesség ellenőrzése.

2. A protokollok megvalósítása szoftverben

A kommunikációs protokollok fejlesztése során először el kell készíteni a követelményrendszer alapján a pontos specifikációt. A megvalósítás alapját képező specifikációt formális nyelvekkel írják le. A manapság használatos specifikációs nyelv az SDL (Specification and Description Language), mely a kommunikáló kiterjesztett véges automaták (CEFSM – Communicating Extended Finite State Machine) matematikai modell alapján biztosítja a követelményrendszer pontos leírását. Az SDL nyelvnek a felhasználók széles körében a GR grafikus folyamatábraszerű leírónyelve terjedt el. Az SDL GR-rel történő leírást programfejlesztő eszközök (tool) támogatják. Az SDL nyelven történt leírásokból a kommunikációs szoftverek automatikusan megvalósíthatók, melyeket gondosan meg kell vizsgálni, hogy megfelelnek-e a rendszertervező által támasztott követelményeknek, valamint a protokollt definiáló szabványoknak.

3. Kommunikációs protokollok és szoftverek tesztelése

A kommunikációs protokollok és szoftverek komoly tesztelési folyamaton kell átessenek, mielőtt a felhasználóhoz jutnak. Tapasztalati tény, hogy a távközlési szoftverek fejlesztési költségeinek több mint 50 %-át a tesztelés költségei teszik ki. Egy hiba kijavítása annál olcsóbb, minél hamarabb találják meg a tesztelési folyamatban. Általános tapasztalat, hogy ha egy hibát a tesztelés során találnak meg, akkor tízszer annyi idő, illetve költség azt kijavítani, mintha maga a fejlesztő találná meg. Ehhez képest is tízszeres a költsége azon hibák kijavításának, amelyeket már egy eladott szoftvertermékben a felhasználó fedez fel. Ekkorra még több idő telik el a fejlesztés óta, és a teljes szoftverrendszerben, ami nagyon sok elemből állhat, nagyon nehéz megtalálni, hogy melyik elem okozta a hibát. Ekkor már nem azok foglalkoznak a szoftverrel, akik írták, hanem az üzembe helyezők és a szoftverkarbantartást végzők. A kódolási hibák javítását persze kódolók végzik, de a legtöbb esetben nem az, aki eredetileg írta a kódot.

Teszteléssel a termék minőségileg nem lesz magasabb színvonalú, csak a hibákat lehet megtalálni. Ezért nagyon fontos, hogy a rendszer megtervezésével párhuzamosan már a tesztelés folyamatát is megtervezzék. A tesztek tervezése során meghatározzák, hogy milyen eszközökkel és milyen módon fognak tesztelni. Ezenkívül összegyűjtik a teszteseteket, azaz meghatározzák, hogy milyen tesztműveleteket fognak elvégezni. A teszteset végrehajtásának első lépése a tesztkörnyezet felállítása. Ezután következik a tesztfeladatok végrehajtása, majd az eredmények kiértékelése.

Megkülönböztetünk statikus és dinamikus tesztelést. Statikus tesztelés valójában a kód vagy kódmódosítás alapos áttekintését, ellenőrzését jelenti. Ezt mindig más kódoló végzi, mint aki a kódot írta. Dinamikus tesztelés során már működésbe hozzák a tesztelendő szoftvert. Input jelsorozatokkal gerjesztik a bemenetén a rendszert, és figyelik a válaszokat a kimeneten.

Beszélhetünk fehér doboz és fekete doboz tesztelésről. Fehér doboz tesztelés során kihasz-

náljuk azt, hogy látjuk a kódot, ismerjük annak felépítését, struktúráját. Fekete doboz tesztelés esetén csak a tesztelt szoftver külső viselkedése ismert. Ekkor szükséges a dinamikus tesztelés, mely során az interfészein keresztül gerjesztik a szoftvert és ellenőrzik, hogy a gerjesztés alapján várt helyes választ kapják-e.

4. A tesztelés helye a fejlesztésben és a különböző teszttípusok

A távközlésben használt szoftverek nagyon nagy méretűek. Fejlesztésük modulonként történik, később a modulokat összeépítik, és így áll elő az egyre nagyobb méretű kód. A tesztelést is érdemes fázisokra bontani, a szoftver fejlesztési folyamatának megfelelően. A különböző tesztfázisokat mutatja az *1. ábra.*

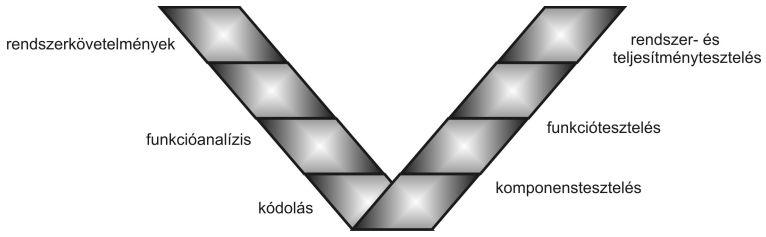
Az első tesztfázis a fejlesztés során az elkészült modulok tesztelése, ezt nevezik modul vagy komponens tesztnek. Ennek során a kódoló olyan teszteket végez el, amellyel ellenőrzi az általa írt kódot, függetlenül a rendszer többi részétől. Ezek a tesztek elsősorban fehér doboz tesztek.

Amikor a modulokat integrálják, további tesztek következnek. Ilyenkor már a kódolóktól független teszterek ellenőrzik, hogy a kód, illetve a több modulból összeállított rendszer a specifikációnak megfelelően működik-e, valamennyi új funkcionalitását ellenőrzik, tipikusan fekete doboz megközelítéssel. Először helyes adatokkal gerjesztik a rendszert, majd azt is megnézik, hogy hogyan reagál hibás bemenetekre.

Ezután következik annak ellenőrzése, hogy a rendszer hogyan fog viselkedni várhatóan a valóságos környezetében. Fogja-e bírni azokat a forgalmi terhelési viszonyokat, ami elvárható, illetve milyen teljesítménnyel fogja végrehajtani a funkciókat? Ez a teljesítménytesztelés feladata, ahol a tesztelést végző eszközön kívül a tesztkonfiguráció gyakran tartalmaz háttérforgalmat generáló eszközt is, ezzel biztosítva azt, hogy különböző forgalmi helyzetekre el tudjuk végezni a mérést.

Kommunikációs szoftverek esetében mindig fontos a szabványos interfészek ellenőrzése, és az, hogy együtt tud-e működni a rendszerünk más gyártók hasonló célra fejlesztett szoftverével. Ezt a konformancia (ITU-T X.290 OSI) és az együttműködési (ETSI TS 102237) tesztek során érik el. A konformancia-tesztelés azt ellenőrzi, hogy a szoftver megvalósítása megfelel-e a specifikációnak (a követelményrendszernek). Konformancia tesztelés során a megvalósított szoftver belső működését nem ismerjük, ahhoz csak meghatározott interfészeken keresztül férhetünk. A konformancia-tesztelést szabványos interfészeken keresztül végzik, így a konformancia-tesztsorozatok implementációtól függetlenek.

A konformancia-tesztelés után általában együttműködési teszteket is végeznek. Az együttműködési tesztek a több szállító ter-



1. ábra • A szoftverfejlesztés és tesztelés kapcsolata

mékeivel való együttműködést kívánják ellenőrizni. Az együttműködési tesztek különösen elterjedtek az internetprotokollok világában, ahol kevésbé szigorúak a protokollszabványok.

A fejlesztés során sokszor újra és újra végre kell hajtani az egyes teszteket. Amikor új funkcionalitást valósítanak meg a szoftverben, vagy hibát javítanak, nem elég az új működést tesztelni, hogy az jól került-e megvalósításra, hanem azt is ellenőrizni kell, hogy nem rontottunk-e el valamit a rendszer régi működéséből. Ezt nevezik regressziótesztelésnek. A regressziótesztelés tulajdonképpen minden olyan esetben nagyon hasznos, ha már letesztelt, ellenőrzött szoftverhez újabb szoftverrészeket adunk.

5. A tesztelési folyamat automatizálása

A tesztelés automatizálása azt jelenti, hogy automatikusan hajtjuk végre a teszteket, és automatikusan értékelődik ki a tesztelés eredménye. Ehhez egy tesztelést végrehajtó eszközre, szoftverre van szükség, valamint arra, hogy meghatározzuk, előre definiáljuk, hogy hogyan tesztelünk, azaz elkészítsük a tesztsorozatot. Így pontosan átgondolt, alapos teszteket lehet létrehozni, amiket emberi beavatkozás nélkül sokszor, gyorsan végre lehet hajtani. Ezzel sok emberi munkát takarítunk meg, hiszen ugyanazokat a teszteket többször kell végrehajtani. Az egyre bonyolultabbá váló kommunikációs szoftverek tesztelésében már nem képzelhető el a megfelelő tesztek végrehajtása automatizálás nélkül. Gondoljunk például a teljesítménytesztelésre, ahol a jelentős terhelést nem is lehet automatizált teszteszközök nélkül létrehozni. A tesztautomatizálás fontos kelléke a tesztsorozatok leírására szolgáló programozási nyelv (például a TTCN-3).

6. A TTCN-3 nyelv

A TTCN-3 nyelv (ETSI ES 201873-1), (Szabó, 2001) elődjét elsősorban a konformancia tesztsorozatok szabványosítására dolgozták ki. Az elképzelés az volt, hogy a gyártók miután megvalósítják a protokollt leíró szabványt, a konformancia tesztsorozat végrehajtásával ellenőrzik a megvalósítás helyességét, és azt, hogy megfelel-e a szabványnak. A protokollmegvalósítástól független tesztsorozatokat szabványosítottak, amihez szükség volt egy szabványos tesztsorozat-leíró nyelvre. Ez volt a TTCN-2, az ISO szabványa. Azonban a széleskörű elterjedésnek gátat szabott a nyelv nehézkes (táblázatos) formátuma. Elsősorban ezért kezdődött el az ETSI-ben (European Telecommunication Standards Institute) egy új tesztleíró nyelv szabványosítása. A TTCN-3 nyelven leírt tesztek könnyen átláthatóak, és számos olyan konstrukciót építettek a nyelvbe, mely segíti a tesztelők dolgát. Így nemcsak a konformanciatesztek leírására alkalmas, hanem együttműködési és teljesítménytesztek programozására is.

A TTCN-3 nyelven könnyen, áttekinthetően adhatunk meg olyan teszteket, melyek a rendszer viselkedését vizsgálják. TTCN-3 nyelven könnyű egy tesztbe beépíteni üzenetek küldését, fogadását, alternatív események figyelését, valamint időzítők indítását, *timeout* (időzítő lejár) esemény kezelését. Ezenkívül lehetővé teszi, hogy a tesztekbe ítéleteket programozzanak be, tehát a teszt futása után rendelkezésre áll az ítélet arról, hogy a teszt sikeresen futott-e. Egyik legfontosabb jellemzője a nyelvnek, hogy olyan teszteket írhatunk benne, melyekben számos egymástól független tesztkomponens működik dinamikusan. Így szinte tetszőlegesen bonyolult tesztrendszer alakítható ki és programozható

a TTCN-3 nyelvvel. A nyelv moduláris felépítése segíti a már megírt tesztek újrahasznosítását regressziós tesztként, vagy olyan rendszerek tesztelésénél, ahol ugyanazt a protokollt vagy funkciót kell tesztelni.

7. Összefoglalás

A cikk röviden ismerteti az infokommunkációs protokollok és szoftverek feladatát és a megvalósítási folyamat lépései közül elsősorban a tesztelés fontosságát tárgyalja. Bemutatja a tesztelés helyét a szoftverfejlesztésben, továbbá azt, hogy miért fontos a tesztelésautomatizálását bevezetni a szoftverfejlesztő projektekben. Ez a tesztelésben alapvető szemléletváltással járó lépés elkerülhetetlen, amit nem kérdőjeleznek meg egyetlen szoftverfejlesztő projektben sem. Azt azonban, hogy mely teszteket érdemes, illetve lehet automatizálni, és milyen módon, mindig az adott fejlesztési környezet és a szoftverrel szemben támasztott követelmények döntik el. Mindazonáltal nagyméretű szoftvereket alapos tesztelés nélkül nem szabad működtetni, mivel nem kellő ellenőrzés esetén váratlan, nehezen elhárítható hibák fordulhatnak elő. A TTCN-3 nyelv jó lehetőséget ad mind a tesztek automatizálására, mind pedig arra, hogy bonyolult tesztrendszereket alakítsunk ki a segítségével.

Kulcsszavak: *kommunikációs protokoll, protokollspecifikáció, protokollmegvalósítás, konformanciatesztelés, SDL, TTCN, együttműködéstesztelés, regressziótesztelés, teljesítménytesztelés, tesztautomatizálás*

IRODALOM

ETSI TS 102237-1 v4.1.1 Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON) Release 4: Interoperability Test Methods and Approaches.

ETSI ES 201873-1 The Testing and Testcontrol Notation version 3; Part 1: TTCN-3 Core Language.

ITU-T X.290 OSI Conformance Testing Methodology and Framework for Protocol Recommendations for ITU-T Applications; General Concepts.

Szabó János Zoltán (2001): A TTCN-3 tesztleíró nyelv. Magyar Távközlés. február