A Rosetta leszállóegység központi számítógépe szoftverének tesztelése

Anisics Zsolt

Mint már valószínűleg ismeretes az olvasók előtt, az Ariane 5 rakéta 2002. decemberi kudarca miatt a 2003. január 12-ére tervezett Rosetta-misszió indítása elmaradt a szonda esetleges elvesztésétől tartva. Jelenleg a lehetséges üstökös mint kutatási célpont, valamint az elérését biztosító pályák keresése folyik. Végleges döntés májusban várható. Az Ariane 5 rakéta hibaanalízise és a hibák kijavítása legkevesebb félévnyi csúszást eredményez a Rosetta-misszió indításában.

Jelenleg a 2004. év elejét tekintjük lehetséges indítási időpontnak, mert ekkor az előzetes számítások szerint különböző pályákon több üstökös is elérhető lesz.

Az űrkutatási eszközöknél utólagos javítás a legtöbb esetben nem lehetséges, ezért a megbízhatóság kiemelten fontos követelmény. A hardverelemek megbízhatóságát különböző környezeti hatásokkal terhelt, valamint a tápfeszültségek határértékén történő üzemeltetése során kapott eredmények alapján értékelik, illetve bizonyítják. Az ilyen környezeti tesztek kiterjednek az indítás során fellépő mechanikai és akusztikai, valamint a nyílt űrben fellépő vákuum- és hőhatások szimulálására.

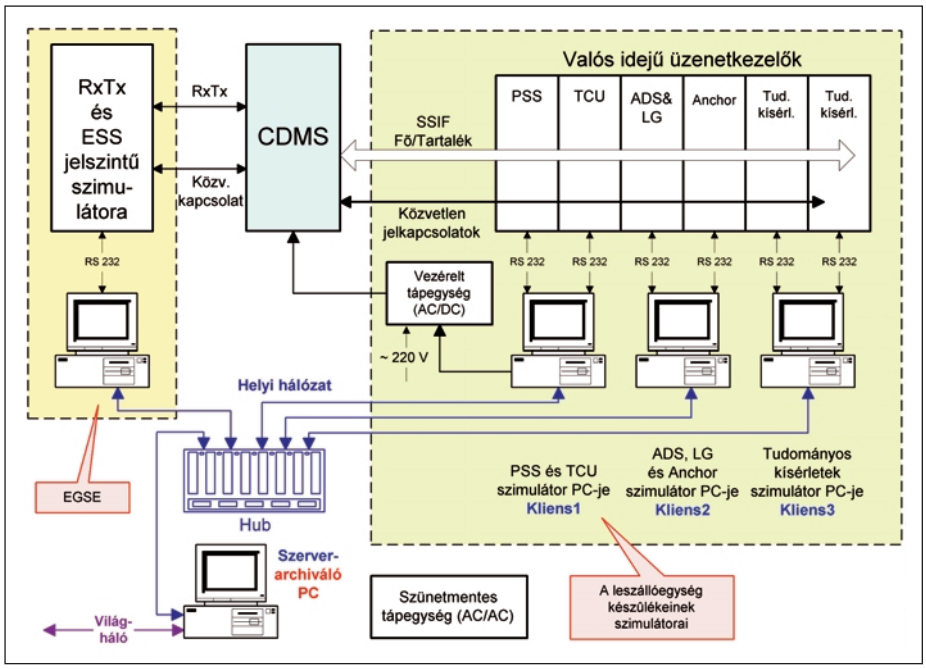
A mikroprocesszort tartalmazó rendszerek megbízhatósága nemcsak az elektronikus és mechanikus alkatrészek megbízhatóságán múlik. A rajta futó szoftver stabil működése is szükséges, azaz a rendszer megbízhatósága a szoftver megbízhatóságától is függ. Tartalékolt rendszerekben a szoftverre igen jelentős feladat hárul, ill. hárulhat az esetleges hibák felismerésekor és ezt követően a tartalékegységek – lehetőleg vezérléskiesés nélküli – zavarmentes aktivizálásánál. A szekvenciális felépítésű programok tesztelése (működésük ellenőrzése) többnyire véges időn belül elvégezhető, és annak megbízhatóságára jó konfidenciaszintű becslés adható. A valós idejű, sokfeladatos szoftverrendszerek teljes körű programfutási tesztelése a nagy számú szoftverkomponens konkurens futási kombinációja miatt szinte lehetetlen.

Az űrprogramoknál legtöbbször, mint a Rosetta-program esetében is, csillagászatilag meghatározott, néhány napra korlátozódó ún. indítási ablak időpontjától visszafelé származtatott időpontra a teljes rendszernek el kell készülnie, hogy egy mindenre kiterjedő tesztelési időszak álljon rendelkezésre. A szoftverek fejlesztése szempontjából ez korai időpontra adódik különösen akkor, ha figyelembe vesszük, hogy a fedélzeti szoftvert (vagy annak egy részét) egyszer írható PROM-memóriákba égetik be. Ezekre a memóriákra pedig már a számítógép hardver összeállítása során szükség van. Sajnos azonban a szoftverrel szemben támasztott követelmények még ezután is változhatnak; a véglegesített PROM-változat után folyamatosan készülnek el az újabb, EEPROM-memóriákba földi parancssorozattal feltölthető szoftverváltozatok.

Egy meghatározott szoftverállapot elérése után az újabb és újabb szoftverváltozatokon a teljes működési tesztsorozatot meg kell ismételni. Ezek a tesztsorozatok egyrészt igen időigényesek, másrészt a hibátlan utasításszekvencia betartása és a válaszreakciók gondos ellenőrzése miatt a tesztelő személyzettől nagy figyelmet követelnek.

Ezt a folyamatot automatizáltuk a Rosetta leszállóegységének központi számítógépe köré kiépített tesztkörnyezet létrehozásával.

Az ismétlődő tesztek elvégzésének szükségessége és azok meggyorsítása indokolta egy automatizált tesztelési folyamatokat végrehajtani képes rendszer, az ún. *testbed* létrehozását. A *testbed* leszállóegység központi számítógépének (CDMS) környezetét szimulálja oly módon, hogy helyettesíti a szolgálati alrendszereket és kísérleteket, valamint a földi kommunikációs alrendszert is. Ezáltal lehetővé teszi a tesztfeladatok programozott végrehajtását. A szimulációkat valós idejű jelszinten egy-egy beágyazott processzort tartalmazó kártya valósítja meg. A teljes környezet szimulálását összesen hat kártya végzi, amelyek önálló készülékházban kaptak helyet. Magát a logikai szimulációt személyi számítógépeken futó egyedi



1. ábra A testbedrendszer blokkvázlata. Az ábrán szereplő rövidítések jelentése:

SSIF (SubSystem Interface) a műszerek adatgyűjtő és vezérlő soros kapcsolata; PSS (Power SubSystem) tápellátó rendszer; TCU (Thermal Control Unit) hőmérséklet-szabályozó rendszer; ADS (Active Descent System) aktív leszállító rendszer; LG (Landing Gear) hármas láb, amely a talajt érési energiát elnyeli; Anchor rögzítőhorgony; (Receiver-Transmitter) rádió adó-vevő; ESS (Electrical Support System) a leszállóegység illesztője a keringő egységen (orbiter); EGSE (Electrical Ground Support Equipment) földi ellenőrző berendezés

programok valósítják meg. A PC-ken futtatható vezérlő szoftver segítségével olyan hibás állapotokat is szimulálhatunk, amelyeket más úton nem lehet tesztelni.

- A tesztkörnyezetnek meg kell oldania a földi vezérlőparancsok CDMS-hez juttatását, a leszállóegység által generált telemetriaadatok vételét, valamint a CDMS és az általa vezérelt egységek közötti adatforgalom ellenőrzését. Ebbe beleértjük az egységek szimulálásából adódó adatokat, valamint annak ellenőrzését, hogy a CDMS az elvárásoknak megfelelően vezérli-e a fedélzeti egységeket.
- A teszt lefutása során nemcsak a parancsok idősorrendben történő kiadását és a vett telemetriaadatok naplózását kell elvégezni, de a válaszokat folyamatosan ki is kell értékelni.

2. ábra Az elkészült rendszer

Fotó: KFKI RMKI

- A rendszernek automatikusan fel kell ismernie a hibás válaszokat vagy működési állapotokat, és könnyen felismerhető módon meg is kell ezeket jelenítenie.
- A naplózás a helyi hálózaton keresztül elérhető központi szerveren történik. Az adatfolyamokat természetesen – tárolásuk mellett – az operátor számára könnyen olvasható formában meg is kell jeleníteni.
- A mérési eredményeket a jó visszakereshetőség érdekében a naplózás mellett adatbázisban is tároljuk, valamint az előforduló eseményekről statisztika is készül. A fontosabb analóg jellegű adatok változását például idődiagramon jeleníthetjük meg.
- Egy tesztszekvencia leírása az egymást követő parancsok relatív időzítéseit, az előírt válaszok tartalmát és időhatárait, esetleges ismétlési paramétereit, valamint az egyes szimulátoregységek vezérlését (válaszreakcióit) leíró információkat is tartalmazza.

A *testbed*-rendszer blokkvázlata az 1. ábrán látható.

Közvetlen jelkapcsolat a tápellátó rendszerrel (PSS) és a leszállásban részt vevő egységekkel van (ADS és LG). A szimulátor számítógépek és a tesztszerver közötti adatkapcsolat egy lokális Ethernet-hálózaton, TCP/IP-protokollon keresztül valósul meg. A tesztelést irányító számítógép tárolja a mérések eredményeit is.

Az éppen figyelt eseményeket (pl. a válaszreakciót) egy folyamatosan frissülő listán tároljuk a teszt leírásában megadott ideig. Amikor az időzítés lejár, megadott eseményeknek kell bekövetkezniük ahhoz, hogy a tesztelő program ne adjon hibaüzenetet. Nem várt események (válaszok) bekövetkezése szintén hibaüzenetet eredményez.

Egy bekövetkezett esemény lehet a telemetria-adatfolyam egy meghatározott szavának, egyetlen bitjének vagy akár több szavának megváltozása is. A rendszer viselkedését a tesztszekvenciák és a bekövetkezett események hivatkozásokkal történő összekapcsolásával megfelelően dinamikussá tudjuk tenni. A szimuláció során tehát megvizsgálhatjuk, hogy a CDMS helyesen reagál-e a repülés alatt fellépő különböző külső és belső eseményekre, vagy akár az esetlegesen előforduló hibajelenségekre.

A telemetria-adatfolyam egy része szolgálati információ (hőmérséklet, feszültség, áramértékek, pozíciók stb.), azaz ún. *house keeping*-adat. Ezek az adatok egy adott készülék állapotának fontos jellemzői. A szolgálati berendezések és a kísérletek szolgálati információinak egységes kezelésére (könnyen olvashatóvá tételére) kialakítottunk egy szöveges, definíciós fájlokon alapuló dekódoló megoldást. Ezáltal az értelmező program forráskódjának módosítása nélkül, egy könnyen kezelhető szöveges fájl segítségével változtathatjuk a különböző szolgálati adatfolyamok (telemetriafájlok) értelmezését.

A *testbed*-rendszer személyi számítógépeken futó programrendszere „C” nyelven készült. Programfejlesztői környezetként a National Instrument LabWindows/CVI-rendszerét használtuk, amely jelentősen megkönnyítette a grafikus kezelői felület, valamint a lokális hálózati programrészek megírását.

A jelszintű szimulátorok az ún. transzputer (transputer = *trans*istor com*puter*) alapú beágyazott processzoros áramkörök. A transzputer olyan mikroprocesszor, amelynek belső architektúrája támogatja párhuzamos feladatok futtatását. A párhuzamos számítási megoldásokat úttörőként valósította meg beépített áramköri megoldásaival és utasításkészletével. Négy nagy sebességű soros busza révén további transzputerek hálózatával bővíthető. Egyetlen utasításciklus alatt végzi el a párhuzamos feladatok közti átkapcsoláskor szükséges nagyszámú állapotregiszter és programváltozó mentését.

A transzputereken futó programokat OCCAM-nyelven fejlesztettük, amely a transzputerek igen fejlett makronyelve, s a többfeladatos szoftverek fejlesztésének megkönnyítésére alkották meg. Sajnos az igen perspektivikus transzputer-architektúra a mindent elsöprő Intel processzorok nagy tömegével szemben alulmaradt, gyártásuk befejeződött.

Egy jelszintű szimulátor több egységet is szimulálhat. Ez annak köszönhető, hogy a feladatok közt a transzputer átváltási ideje igen rövid. A megvalósított rendszerben például csupán két beágyazott processzoros rendszer látja el a nyolc tudományos kísérlet szimulálását. Maguk a szimulátorok szabványos soros porton keresztül tartják a kapcsolatot az asztali számítógépekkel. Egy számítógép két szimulátoregységet vezérel.

Az ismertetett rendszert a német űrkutató központ, a DLR (Deutsches Zentrum für Luft- und Raumfahrt) megrendelésére fejlesztettük ki. A Rosetta leszállóegység számítógépének szoftverfejlesztői az elkészült rendszert (2. ábra) a KFKI RMKI űrkutatási osztályán használják.

Céljuk: az új szoftverváltozatok kötelező tesztjeinek elvégzése a fedélzeti EEPROM-memóriába történő feltöltése előtt.

SGF Kft.
(Space & Ground Facilities Ltd.)
1121 Budapest
Pipiske u. 1–5/20.
Tel.: (06-1) 398-0190
E-mail: info@sgf.hu