REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK
On-Line tudományos folyóirat
XXVI. évfolyam 2014. 2. szám

Schuster György[1] – Terpecz Gábor[2]

INKREMENTÁLIS FEJLESZTÉS ÉS TESZTELÉS BIZTONSÁGKRITIKUS RENDSZEREK ESETÉN[3]

A szoftver kritikus sikertényező és mint ilyen alkalmazása szinte minden berendezésben elkerülhetetlen. A skála rendkívül széles a kapucsengőtől a kerékpárlámpán keresztül a járművek fedélzeti rendszerét érintve a nukleáris létesítményekig mindenütt találkozunk szoftverekkel. Sajnálatos módon tapasztalataink azt mutatják, hogy a megrendelők – tisztelet a kivételnek – nagyrészt képtelenek a feladatot kellő mélységben specifikálni, aminek az az eredménye, hogy kutatások szerint az átadott programok jó része további módosításra szorul, jelentős részüket nem használják és szinte elenyésző az a szám ahol az átadás után nincs szükség kisebb – nagyobb változtatásra. Ezen a jelenségen sajnos változtatni nem nagyon lehet, ezért számos esetben a fejlesztés során az inkrementális modellt vesszük alapul, akár biztonságkritikus alkalmazásokban is.

INCREMENTAL DEVELOPMENT AND TEST IN CASE OF SAFETY CRITICAL SYSTEMS

Summary: software is critical success factor therefore it is inevitable to apply in almost each device. This scale is very large from doorbell via bicycle lamp, vehicle board systems to atomic power plants. Unfortunately our experience shows that customers – except for fews – are unable to specify its problem in every detail. This leads to the result that dominant part of delivered programmes need later modifications or some of them is not used. Number of delivered and not modified software is tiny. That is the reason why the increment development model is considered as basic model even if safety critical systems.

BEVEZETŐ

A szoftvertechnológia számos életciklus modellt ismer, ezek mind rendelkeznek előnyökkel és hátrányokkal. Ezek a modellek:

- vízesés modell,
- V modell (más néven német V modell),
- spirál modell,
- evolúciós modell,
- inkrementális modell.

Az inkrementális modellt kivéve szinte mindegyik az első pillanattól egyértelmű és pontos specifikációt igényel. Erre a vízesés és a V modell a legjobb példa.Az inkrementális modell viszont nem ennyire merev, amikor rendelkezésre áll egy adott részterületről kellő mennyiségű információ a fejlesztés és a tesztelés megkezdhető.

[1] ÓE KVK MAI schuster.györgy@kvk.uni-obuda.hu
[2] ÓE KVK MAI terpecz.gabor@kvk.uni-obuda.hu
[3] Lektorálta: Prof. Dr. Pokorádi László egyetemi tanár, Óbudai Egyetem, pokoradi.laszlo@prosysmod.hu

A modell akkor alkalmazható, ha kellően nagy vonalakban ismerjük az elkészítendő szoftver által biztosított szolgáltatásokat, de a teljes rendszer még nem teljesen specifikált. A hangsúly a nem teljesen van. Ez azt jelenti, hogy tudjuk, hogy a rendszernek mit kell csinálni, de a részletek nem minden esetben állnak rendelkezésre.

Példának hozhatjuk azt az esetet, amikor egy UAV fejlesztése párhuzamosan történik a sárkány és a fedélzeti rendszer esetén.

A szoftver fejlesztést a megrendelés pillanatában el kell kezdeni, holott az lenne optimális, ha már a teljes mechanika az ismert dinamikus tulajdonságaival rendelkezésünkre állna.

Ez egyszerűen azt eredményezné, hogy a fejlesztési idő a megrendelő számára elfogadhatatlanul hosszúra nyúlna.

A szituáció mindenki számára ismert. Azonban kellő tapasztalatok birtokában lehetőség nyílik arra, hogy a programozók előre dolgozzanak.

Ennek oka az, hogy kevés kivétellel az adott feladatok számos részét előre meg lehet határozni. Ezeket prioritási sorrendbe rakva el lehet kezdeni fejleszteni, illetőleg tesztelni. Ebben segítséget nyújt a moduláris és az objektum orientált módszertan.

A módszer lépései:

1. azon elemek kiválasztása, amelyek meghatározottak, jól körülhatárolhatók, tehát elegendő információ áll rendelkezésre, hogy az adott elemet meg lehessen valósítani,
2. a kiválasztott elemek logikai és fizikai tervezése – különös tekintettel a szükséges interfész felületekre,
3. a fenti tervek alapján a szoftver elem kódolása,
4. a szoftver elem tesztelése először a fejlesztői host-on, és amint lehetséges a target rendszeren is,
5. amennyiben lehetőség van a következő inkrementális lépés megkezdésére akkor annak végrehajtása és az interfész és együttműködési tesztek végrehajtása.

Az inkrementális fejlesztés az első pillantásra nem felel meg a biztonságkritikus filozófiának, de az időkényszer – ami sokszor az adott problémához nem, vagy igen kevéssé értő környezet által generált – nem teszi lehetővé a kívánt fejlesztési modell alkalmazását.

Azonban ennek a fejlesztési modellnek van előnye is. Azok a szoftver egységek, amelyek a fejlesztés adott szakaszában elkészülnek azonnal tesztelhetők és viselkedésükről, tulajdonságaikról tapasztalatokat lehet szerezni.

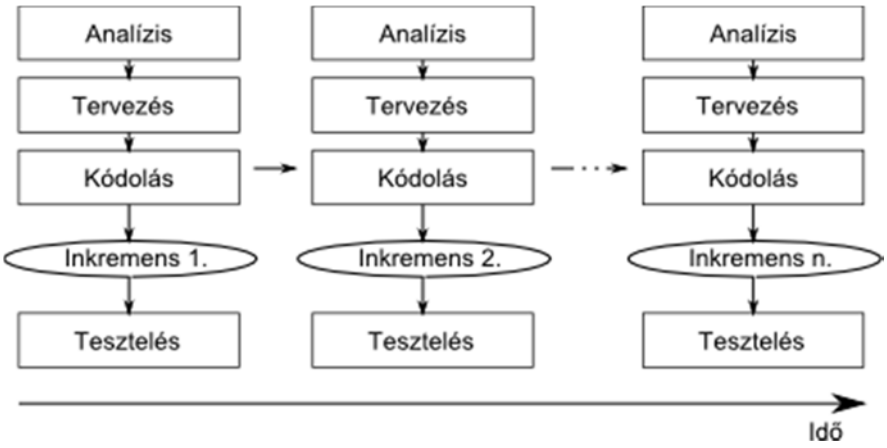
Ez nem csak a host-on való tesztelést, hanem a target-en történő futtatást is jelentik.

Ez például jelentheti azt is, hogy kellően korai időben a target nem elegendően erős a feladat ellátására.

Összefoglalva az inkrementális fejlesztés tulajdonságait:

6. korábban készülnek tesztelhető szoftver egységek, tehát a megrendelő korábban kap már tesztelhető elemeket,

7. lehetőség nyílik az egyes részek korai tesztelésére
8. mivel a target rendszeren korán futtatási teszteket lehet végrehajtani, időben kiderülhet a target alkalmatlansága,
9. a fejlesztés a projekt (összetett hardver, szoftver, projekt rendszerről beszélünk) korai időszakában megkezdődhet a rendelkezésre álló információk szerint,



1. ábra Az inkrementális fejlesztés vázlata

Azt ne felejtsük el, hogy a fejlesztés megkezdésekor a kérdéses szoftver egységek fejlesztési sorrendje a rendelkezésre álló információktól függ, de ekkor is célszerű a magasabb prioritású elemeket kiválasztani és ezekkel kezdeni a munkát.

Az inkrementális fejlesztés azonban nem program egysége független fejlesztését jelenti, bár ebből indulunk ki, de a fejlesztés a valóságban folyamatosan, viszonylag kis lépésekben történik, úgynevezett inkrementális szinteken történik.

Tesztelés

Ennek a fejlesztés modellnek az alapvető tulajdonsága az, hogy a szoftver egységek nem egyszerre készülnek el, illetve minden inkrementális lépés után – a lehetőségekhez mérten – működő kóddal rendelkezünk. Ez viszont azt eredményezi, hogy ez a kód már az első lépés után tesztelhető.

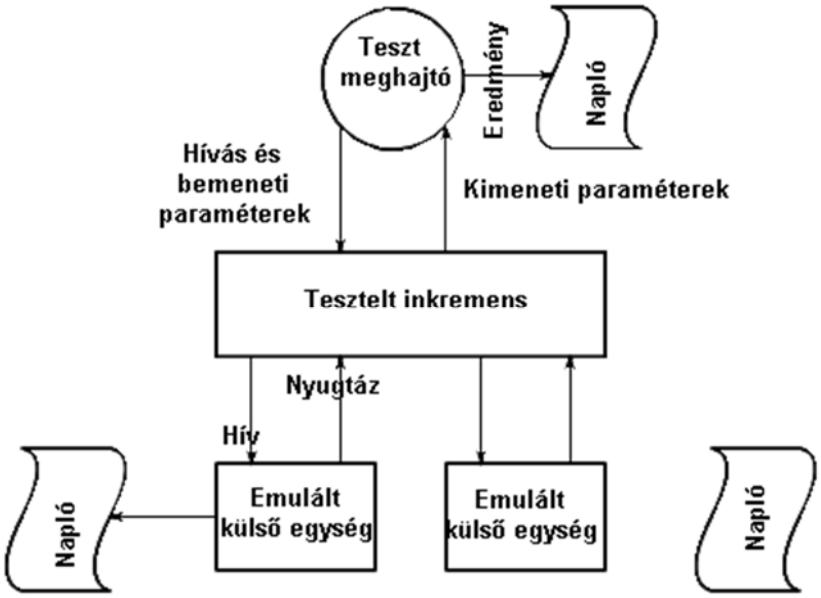
Célszerűen az adott szoftver elem adott inkrementális elem úgy kezelendő, mint egy a klasszikus unit tesztelésnél egy unit. Annyiban egyszerűbb a helyzet, hogy a már előző inkremensben leteszteltelt és várhatóan hibátlan szint (a teszt nem talált hibát, természetesen ettől még lehet) már nem teszteljük újra csak annyiban, amennyiben a kérdéses inkrementális szint megköveteli.

A statikus ismert unit tesztelés lépései alkalmazhatóak, úgymint:

- vezérlési út tesztelés,
- adatfolyam tesztelés,
- tartomány teszt,
- funkcionális teszt.

A dinamikus tesztelésnél célszerű eljárások:

- mutációs teszt (biztonságkritikus rendszereknél a mutáció elvileg nem fogadható el),
- funkcionális tesztelés.



2. ábra Alsó rétegben lévő inkremens tesztelése

Mivel a kérdéses szoftver elem egy része a szoftvernek, ezért feltételezzük, hogy interfészei nem csatlakozhatnak további valós szoftver elemhez. Továbbá nagy valószínűséggel ez az elem nem a legalsó, vagy a legfelső rétegben van. Ennek következménye, hogy a kérdéses elem interfészeit emulálnunk kell.

Nyilván valóan az emulált interfészek is szoftver elemek. Külön kérdést jelent ezeknek az előállítása, illetőleg helyessége.

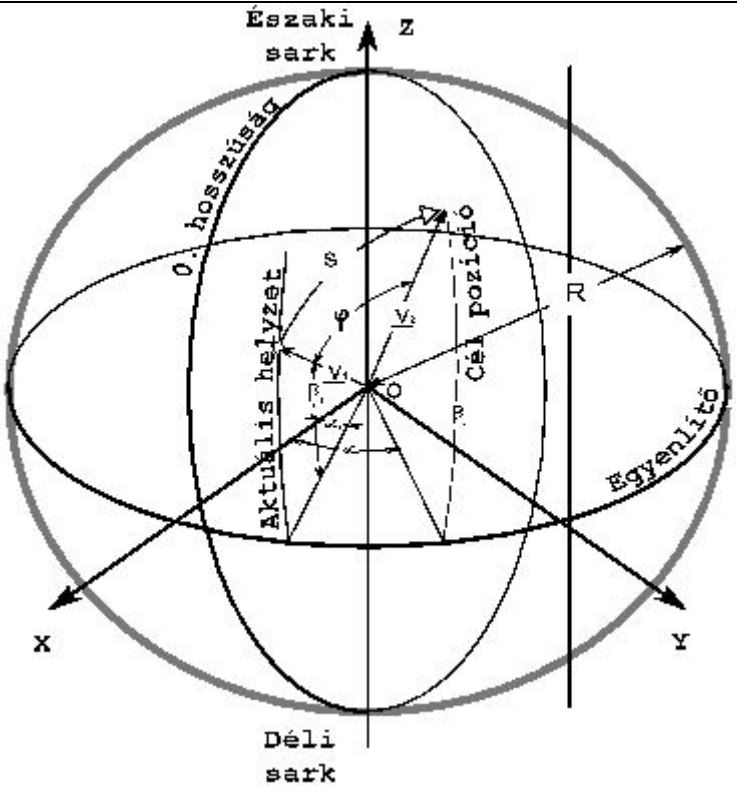
A 2. ábra egy nem tipikus tesztelési szituációt mutat, ahol a kérdéses szoftver elem közel van a fizikai eszközhöz, tehát emulált külső egységeket kell alkalmazni. Esetlegesen, ha rendelkezésre áll a hardver, akkor célszerű azon is az elemet kipróbálni. Így lehetőség nyílik az erőforrás használat és a futásidő mérésére is.

Abban az esetben, ha a fejlesztési lépés valamilyen algoritmus, akkor a host-on való futtathatóság megkönnyíti az algoritmus helyességének ellenőrzését. A következő fejezetben erre mutatunk egy példát.

Példa

Ebben a fejezetben egy algoritmus nem szokványos tesztelését mutatjuk be. A feladat egy GPS alapú navigáció megvalósítása és ellenőrzése.

A navigáció célja az, hogy a kitűzött és hosszúsági és szélességi adatokkal megadott célt a légijármű a lehető legrövidebb úton, vagyis ortodrómát követve érje el. A pozíció számítás minden GPS leolvasáskor megtörténik.



3. ábra Navigációs szituáció

Jelmagyarázat:

- O a bolygó középpontja,
- α_1 az aktuális pont hosszúsági koordinátája
- β_1 az aktuális pont szélességi koordinátája,
- α_2 a célpont hosszúsági koordinátája,
- β_2 a célpont szélességi koordinátája,
- $\underline{v_1}$ az aktuális pont helyvektora,
- $\underline{v_2}$ az célpont helyvektora,
- R a bolygó sugara,
- φ az aktuális pálya (aktuális ponttól a célpontig) szöge az ortodrómán,
- s az aktuális pálya (aktuális ponttól a célpontig) hossza az ortodrómán,

Az aktuális pozíció helyvektora:

$$\underline{v_1} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$, ahol rendre: $z_1 = R\sin\beta_1$, $x_1 = R\cos\alpha_1\cos\beta_1$, $y_1 = R\sin\alpha_1\cos\beta_1$

A célpont helyvektora:

$$\underline{v}_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}$$, ahol rendre: $z_2 = R\sin\beta_2$, $x_2 = R\cos\alpha_2\cos\beta_2$, $y_2 = R\sin\alpha_2\cos\beta_2$

Az ortodróma sík normál egységvektora:

$$\underline{n}_p = \frac{\underline{v}_2 \times \underline{v}_1}{\|\underline{v}_2 \times \underline{v}_1\|} = \frac{\underline{v}_2 \times \underline{v}_1}{R^2 \sin\phi}$$

Az aktuális pályaszög és a hátralévő út az ortodrómán:

$$\phi = \arcsin\frac{\underline{v}_2 \times \underline{v}_1}{R^2} \quad \text{és} \quad s = R\arcsin\frac{\left\|\underline{v}_2 \times \underline{v}_1\right\|}{R^2}$$

Az aktuális hosszúsághoz tartozó ortodróma normál egységvektora:

$$\underline{n}_l = \begin{pmatrix} -\sin\alpha_1 \\ \cos\alpha_1 \\ 0 \end{pmatrix}$$

A kérdéses pozícióban a geológiai irányszög a keleti féltekén:

$$\delta = \begin{cases} \arccos\langle \underline{n}_p, \underline{n}_l \rangle, & \beta_2 > \beta_1 \\ \pi/2, & \beta_2 = \beta_1 \\ \pi - \arccos\langle \underline{n}_p, \underline{n}_l \rangle, & \beta_2 < \beta_1 \end{cases}$$

A számítás után az eredmény radiánban érhető el, ezt könnyen át lehet számítani fokra:

$$\delta_{gr} = 180 \quad \delta/\pi$$

Amennyiben valamelyik hosszúsági kör a nyugati, illetve a szélességi kör déli féltekére esik a kérdéses paramétert negatívnak kell venni.

Az algoritmus ellenőrzésre több módszer is használható, ezek:

- ideális gömbön az algoritmus alkalmazása előre meghatározott koordináta értékekkel pontról pontra,
- a google-maps, illetve a google-earth programon a szimulált navigációs számítások ellenőrzése,

Az első módszer nagyon egyszerű a két adott koordináta között gyakorlatilag igen kis lépésekkel ellenőrizzük az útvonalat, majd az eredményt összevetjük az ideális ortodoxával. Ha ezt kellő felbontással képernyőre tesszük igen látványos az esetleges hiba.

A második módszer a google-earth használata. Ekkor szintén adott pontból, adott sebességet feltételezve indítjuk az algoritmust.

A célpontként kijelölt helyet elfogadható pontossággal kell elérni. Esetünkben a Pécs-Pogány repülőtérről (LHPP pályaközép Lat.: 45°59'21.01"N, Lon.: 18°14'31.99"E) Budaörs repülőtérre (LHBS 27L várópont Lat.: 47°26'58.99"N, Lon.: 18°59'12.98"E) 120 km/h-s sebességet és percenkénti négy mintavételt feltételezve ellenőriztük a navigációt.

Ennél a természetesen egy valós GPS jóval sűrűbben biztosít információt.

Az eredmény többszöri futtatás után 10–30 méteres hibákat mutatott.

Azt mindenképpen szeretnénk megjegyezni, hogy ez nem egy valós navigációs szituáció, hanem egy algoritmus teszt, tehát csak és kizárólag a navigációt teszteltük a légtereket figyelmen kívül hagyva. Ez a teszt virtuális voltát tekintve megtehető.

ÖSSZEFOGLALÓ

Az inkrementális fejlesztés biztonságkritikus rendszerek esetén közel sem optimális fejlesztési modell, azonban a néha ésszerűtlen határidők betartását elősegítheti.

Optimális lenne, ha a fejlesztendő rendszer kellő módszerességgel, egyenletes ütemezésben készülne.

Tapasztalatink azt mutatják, hogy kellő módszerességre szinte soha nincs idő. Ezért a fejlesztők már akkor neki kezdenek a tervezésnek és kódolásnak, amikor a rendszer még bőven tartalmaz tisztázatlan részeket.

A módszer előnye, hogy ezzel idő takarít meg, mert az adott inkremens korábban készül el és tesztelhető. A legnagyobb felelősség az interfész tesztet végző munkatársakra hárul, nekik kell azokat a lehetséges problémákat felfedezni, amelyek a projekt sikerét és esetlegesen a felhasználók egészségét veszélyeztetik.

FELHASZNÁLT IRODALOM

[1] FODOR ATTILA - VÖRÖSHÁZI ZSOLT BEÁGYAZOTT RENDSZEREK ÉS PROGRAMOZHATÓ LOGIKAI ESZKÖZÖK. EGYETENI TANANYAG ISBN 978-963-279-500-3

[2] ROGER S. PRESSMAN, PH.D. SOFTWARE ENGINEERING A PRACTITIONER'S APPROACH ISBN MCGRAW HILL ISBN 978–0–07–337597–7

[3] CEM KANER JACK FALK - HUNG QUOC NGUYEN TESTING COMPUTER SOFTWARE WILEY ISBN ISBN-10: 0471358460 | ISBN-13: 978-0471358466

[4] JÓZSEF KOPJÁK - JÁNOS KOVÁCS: TIMED COOPERATIVE MULTITASK FOR TINY REAL-TIME EMBEDDED SYSTEMS IEEE 10TH JUBILEE INTERNATIONAL SYMPOSIUM ON APPLIED MACHINE INTELLIGENCE AND INFORMATICS, HERL'ANY, SLOVAKIA, ISBN:978-1-4577-0197-9 PP. 377-382

[5] KSHIRASAGAR NAIK, PRIYADARSHI TRIPATHY, SOFTVWARE TESTING AND QUALITY ASSURANCE, WILEY ISBN ISBN 978-0-471-78911-6