1 BEVEZETÉS

TÉMAKÖRÖK

Ezen fejezet feladata az, hogy bevezesse a szoftvertervezés témakörét. Ha elolvassuk ezt a fejezetet, a következő témákat ismerhetjük meg:

- megértjük, mi maga a szoftvertervezés, és hogy miért is fontos az;
- kulcskérdésekre kapunk válaszokat, bevezetésként a szoftvertervezésbe;
- szoftvertervezők számára fontos etikai és szakmai témákat érintünk és értünk meg.

TARTALOM

Napjainkban virtuálisan minden ország számítógép-alapú, összetett rendszerektől függ. Egyre több termék foglal magában számítógépeket és áll valamilyen formában szoftveres irányítás alatt. A szoftverek ezekben a rendszerekben a teljes rendszer költségének nagy és egyre növekvő hányadát teszik ki. Ennek következtében a szoftverek költséghatékony módon történő előállítása elengedhetetlen a nemzeti és nemzetközi gazdaság működéséhez.

A szoftvertervezés mérnöki tudományág, melynek célja a szoftverrendszerek költséghatékony fejlesztése. A szoftver nem anyagi dolog, nem irányítható törvényekkel vagy gyártási folyamatokkal. Bizonyos szempontból a szoftverfejlesztés leegyszerűsödik, hiszen nincs fizikai korlátja annak, hogy egy szoftvernek mit kell tudnia. Más szavakkal, a természetes megszorítások hiánya azt eredményezi, hogy egy szoftver könnyen szélsőségesen összetetté és így nehezen érthetővé válhat.

A *szoftvertervezés* fogalma először 1968-ban egy, a későbbiekben „szoftverkrízis" névvel emlegetett probléma tisztázása céljából tartott konferencián hangzott el. Ez a szoftverkrízis az (akkor még) erős, harmadik generációs hardverek bevezetésének köszönhető. Azok teljesítménye ugyanis az addig nem megvalósítható alkalmazásokat is megvalósítható feladatokká tette. Mindezek eredményeképpen a szoftverek nagyságrendekkel nagyobbak és komplexebbek lettek elődeiknél.

Az ilyen rendszerek építésekor korán megmutatkozott, hogy a szoftverfejlesztés hétköznapi megközelítése nem megfelelő. A nagyobb projektek néha éveket késtek. Költségeik jóval magasabbak voltak, mint azt előre jósolták. A termékek megbízhatatlanok, nehezen karbantarthatók, gyengén kivitelezettek voltak. A szoftverfejlesztés válságban volt. A hardverköltségek leestek, míg a szoftverek költségei magasra emelkedtek. Új technikákra és módszerekre volt szükség, hogy ellenőrizni lehessen a nagy, komplex rendszereket.

Ezek a technikák a szoftvertervezés részeivé váltak, s széles körben azok most is, de nem egyetemesen használtak. Mindazonáltal napjainkban még mindig probléma a komplex rendszerek előállítása során megfelelni a felhasználó elvárásainak a költségek és a határidő terén. Számos szoftverprojekt küzd problémákkal, melyek oda vezettek, hogy néhányan a szoftvertervezés krónikus betegségéről beszélnek.

Ahogyan a szoftvertervező képességeink növekedtek, úgy növekedett az igényelt szoftverrendszerek összetettsége. A számítógépek és a telekommunikációs rendszerek közeledése új kihívások elé állította a szoftvertervezőket, és ez új technológiákat eredményezett. Emiatt és azért, mert számos cég nem alkalmazott hatékony szoftvertervezési technikákat, a problémák továbbra is megmaradtak. A helyzet persze nem olyan sötét, mint amilyennek látszik, de egyértelműen szükség van a tökéletesítésre.

Úgy hiszem, a szoftvertervezés félelmetes fejlődésen ment keresztül 1968 óta, és ez nyilvánvalóan tökéletesítette szoftvereinket. Sokkal jobb rálátásunk van a szoftverfejlesztés tevékenységeire. Hatékony módszereket fejlesztettünk ki a szoftverspecifikáció, a szoftvertervezés és -implementáció problémáira. Az újfajta jelölések és eszközök csökkentik a nagy és összetett rendszerek előállításához szükséges erőfeszítéseket.

Pontosan tudjuk, hogy nincs „ideális" megközelítése a szoftvertervezésnek. A különféle típusú rendszerek és az őket használó szervezetek széles skálája miatt a szoftverfejlesztéseknek is hasonlóan széles skálán kell mozogniuk. Mindazonáltal a folyamatok és a rendszerek szervezésének alapfogalmai megegyeznek az összes technikában, és ezek képezik a szoftvertervezés esszenciáját.

A szoftvertervezők jogosan lehetnek büszkék eredményeikre. Összetett rendszerek nélkül nem derítenénk föl újabb területeket, nem lenne internet és modern telekommunikáció, és az utazás minden formája veszélyes és drága lenne. A szoftvertervezés nagymértékben közreműködött saját fejlesztési élettartamának lerövidítésében is. Személy szerint meg vagyok győződve arról, hogy ez a tudomány éretté vált, hatása a 21. században még jelentősebb lesz.

1.1. GYAKRAN ISMÉTLŐDŐ KÉRDÉSEK A SZOFTVERTERVEZÉSBEN

Ez az alfejezet arra hivatott, hogy választ találjunk néhány, a szoftvertervezésben felmerülő alapvető kérdésre. A gyakran ismétlődő kérdések (Frequently Asked Question, FAQ) formáját követjük. Ez a tárgyalási mód gyakran alkalmazott az internetes hírcsoportok körében, az újonnan betévedők gyakran feltett kérdéseit megválaszolandó. Úgy hiszem, ez hatékony módja annak, hogy tömören bemutassuk a szoftvertervezés témakörét.

Az 1.1. ábra összegzi, mely kérdésekre kapunk választ ebben a fejezetben.

Kérdés	Válasz
Mi a szoftver?	Számítógépprogramok és kapcsolódó dokumentációk. Szoftvertermékek fejleszthetők egyedi ügyfelek számára vagy akár általános piacra.
Mi a szoftvertervezés?	A szoftvertervezés mérnöki tudományág, mely a szoftvertermékek minden lehetséges aspektusát érinti.
Mi a különbség a szoftvertervezés és a számítógéptudomány között?	A számítógép-tudomány elméletekkel és alaptételekkel foglalkozik, a szoftvertervezés pedig a használható szoftverek fejlesztésének és leszállításának gyakorlati aspektusait érinti.
Mi a különbség a szoftvertervezés és a rendszertervezés között?	A rendszertervezés a számítógép-alapú rendszerek tervezésével foglalkozik, ideértve a hardver, a szoftver és a folyamatok tervezését. A szoftvertervezés ennek egy része.
Mi a szoftverfolyamat?	Tevékenységek halmaza, ahol a cél a szoftver kifejlesztése.
Mi a szoftverfolyamat modellje?	A szoftverfolyamat egyszerűsített, adott perspektívából alkotott reprezentációja.
Mi a szoftvertervezés költsége?	A költségek kb. 60 százaléka fejlesztési költség és 40 százalék tesztelési költség. Az egyedi szoftverek költségei és az evolúciós költségek gyakran meghaladják a fejlesztési költségeket.
Mik a szoftvertervezési módszerek?	A szoftverfejlesztés strukturált megközelítési módjai, amelyek rendszermodellt, jelölésrendszert, szabályokat, tervezési ajánlásokat és folyamatirányításokat foglalnak magukban.
Mi a CASE (Computer-Aided Software Engineering – számítógéppel támogatott szoftvertervezés)?	Szoftverrendszer, melynek célja a szoftverfolyamat tevékenységének automatizált támogatása. A CASE-rendszereket gyakran alkalmazzák a módszerek támogatására is.
Mik a jó szoftver tulajdonságai?	A szoftvert az elvárt funkciókkal és teljesítménnyel kell a felhasználó számára szállítani, ezenfelül karbantarthatónak, üzembiztosnak és használhatónak kell lennie.
Mik a szoftvertervezés főbb kihívásai?	Megbirkózni a meglévő rendszerekkel és a sokféleség növekedésével, megbirkózni a szállítási idő csökkentésének igényével.

1.1. ábra. Gyakran ismétlődő kérdések a szoftvertervezésben

1.1.1. A szoftver

A *szoftver* szót sokan egyenlőnek tekintik a számítógépes programokkal. Valójában ez túlságosan szigorú nézet. A szoftvert nem csak maguk a programok alkotják, hanem a hozzájuk kapcsolódó dokumentációk, illetve konfigurációs adatok, amelyek elengedhetetlenek ahhoz, hogy ezek a programok helyesen működjenek. Egy szoftverrendszer általában számos elkülönített programot tartalmaz; konfigurációs állományokat, melyek arra használatosak, hogy beállítsuk ezeket a programokat; rendszer-dokumentációt, amely leírja a rendszer szerkezetét; felhasználói dokumentációt, amely elmagyarázza a rendszer használatát; valamint webhelyeket, ahol újabb információkhoz jutunk a felhasználói termékkel kapcsolatban.

A szoftvertervezők szoftverfejlesztéssel foglalkoznak, azaz olyan szoftverekkel, amelyek később a fogyasztóknak eladhatók. A szoftvertermékeknek két nagyobb csoportja létezik:

1. *Általános termékek.* Ezek az egyedülálló rendszerek, amelyeket egy fejlesztő szervezet készít és ad el a piacon bármely vevőnek, aki azt képes megvásárolni. Gyakran úgy hivatkoznak ezekre, mint dobozos szoftverekre. Ez a típus foglalja magában például az adatbázis-kezelőket, a szövegszerkesztőket, a rajzoló csomagokat és a projektmenedzselési eszközöket.
2. *Egyedi igényekre szabott (rendelésre készített) termékek.* Az ilyen rendszerek egyéni megrendelők megbízásai alapján készülnek. A szoftver szállítója speciálisan a megrendelő igényei szerint fejleszti a terméket. Ilyen szoftverek például az elektromos eszközök vezérlőrendszerei, az egyéni üzleti folyamatokat támogató rendszerek és a forgalomirányító és ellenőrző rendszerek.

Fontos különbség a két szoftvercsoport között az, hogy az általános termékek esetében a szoftverspecifikációt az a cég tartja kézben, amelyik a terméket fejleszti. A rendelésre készített termékek esetén ezt általában az a cég adja meg és ellenőrzi, amelyik megvásárolja a szoftvert. A szoftverfejlesztőknek tehát ennek a specifikációnak az alapján kell dolgozniuk.

Mindazonáltal a kétfajta termék közötti választóvonal egyre inkább elmosódik. Egyre több szoftvercég kezd általános termékekkel, majd pedig azokat a különböző vásárlók igényeire szabja. A vállalatirányítási rendszerek (Enterprise Resource Planning, ERP), mint például az SAP-rendszer, nagyon jól mutatják ezt a megközelítési módot. Itt nagy és komplex rendszerekre kell gondolnunk, melyek egy adott cég számára adaptálásra kerülnek oly módon, hogy megfeleljenek azok üzleti szabályainak és folyamatainak, jelentésigényeinek stb.

1.1.2. A szoftvertervezés

A szoftvertervezés mérnöki tudományág, mely a szoftvertermékek minden aspektusát érinti a rendszer-specifikáció korai szakaszaitól a rendszerkarbantartáson át egészen a rendszer bevezetéséig. Ebben a definícióban két kulcsjelentőségű szókapcsolat található:

1. *Mérnöki tudományág.* Ezzel foglalkoznak a mérnökök. Elméleteket, módszereket és eszközöket alkalmaznak ott, ahol azok megfelelőek, de azokat előre jól megválasztják. Mindig megpróbálnak az olyan esetekre is megoldásokat találni, amikor nincs olyan alkalmazható elmélet vagy módszer, mely a probléma megoldását támogatná. A mérnököknek azt is fel kell ismerniük, hogy szervezeti és pénzügyi megszorítások közt kell dolgozniuk, azaz ilyen megszorítások mellett keresik a megoldásokat.
2. *A szoftvertermékek minden aspektusa.* A szoftvertervezés nemcsak a szoftverfejlesztés technikai folyamatait érinti, hanem az olyan tevékenységeket is, melyek a projekt menedzselésére, a szoftvert támogató eszközökre, elméletek és módszerek fejlesztésére is vonatkoznak.

Általánosan elmondható, hogy a szoftvertervezők szisztematikus és szervezett megközelítési módot vesznek át saját munkájukba, és gyakran ez a leghatékonyabb módja a jó minőségű szoftverek előállításának. Mondhatni, a tervezés nem más, mint a legmegfelelőbb módszerek kiválogatása az adott körülményekhez igazodóan, és a fejlesztés kreatívabb informális megközelítése, amely bizonyos körülmények között hatékony lehet. Ez a hétköznapi, informális fejlesztés különösen a webalapú e-kereskedelemi rendszerek fejlesztésében helyénvaló, mert ezek egyszerre követelik meg a szakértelmet mind a szoftver-, mind a grafikus tervezéshez.

1.1.3. Mi a különbség a szoftvertervezés és a számítógép-tudomány között?

A számítógép-tudomány főként a számítógépek és szoftverek alapjául szolgáló elméletekkel és módszerekkel foglalkozik, ezzel szemben a szoftvertervezés a szoftver előállításának gyakorlati problémáira keres megoldásokat. Bizonyos számítógép-tudományi ismeretek elengedhetetlenek a szoftvertervezőknél ugyanúgy, ahogy bizonyos fizikai ismeretek nélkülözhetetlenek a villamosmérnökök számára.

Ideális esetben minden szoftvertervezés alátámasztható számítógép-tudományi elméletekkel, de a valóságban ez a legtöbb esetben nem igaz. A szoftvertervezőknek gyakran kell *ad hoc* megközelítési módokat alkalmazniuk a szoftverfejlesztés folyamán. A számítógép-tudomány elegáns elméletei nem mindig alkalmazhatók a valós élet szoftvermegoldást kívánó összetett problémáira.

1.1.4. Mi a különbség a szoftvertervezés és a rendszertervezés között?

A rendszertervezés, vagy még pontosabban a számítógép-alapú rendszerek rendszertervezése a komplex rendszerek fejlődésének és fejlesztésének minden olyan aspektusát érinti, ahol a szoftver tölti be a legfontosabb szerepet. A rendszertervezés ennek következtében ugyanúgy foglalkozik a hardverfejlesztésekkel, az eljárásmódok és folyamatok tervezésével, rendszertelepítéssel, mint a szoftvertervezéssel. A rendszertervezők részt vesznek a rendszer-specifikációban, az átfogó architektúra definiálásában, valamint a különböző részek integrálásában, hogy elkészítsék a teljes rendszert. Kevésbé érintettek a rendszerkomponensek tervezésében (hardver, szoftver stb.).

A rendszertervezés régebbi tudományág, mint a szoftvertervezés. Az emberek már több mint 100 éve specifikálnak és szerelnek össze komplex ipari rendszereket, mint például vegyi üzemeket vagy vasutakat. Mindazonáltal a szoftverek jelenléte növekszik a rendszerekben, a szoftvertervezési technikákat, mint például a használati eset modellezést, a konfigurációkezelést stb., egyre elterjedtebben használják a rendszertervezési folyamatokban. A rendszertervezést részletesebben a 2. fejezet tárgyalja.

1.1.5. A szoftverfolyamat

A szoftverfolyamat tevékenységek és kapcsolódó eredmények olyan halmaza, amely szoftverterméket állít elő. Ezeket a tevékenységeket nagyobbrészt a szoftvertervezők hajtják végre. Négy alapvető tevékenységet különböztetünk meg (a könyvben később részletesen tárgyalom ezeket), amelyek minden szoftverfolyamatban közösek. Ezek a tevékenységek:

1. *Szoftverspecifikáció.* A szoftver működését és a működésre vonatkozó megszorításokat kell definiálni.
2. *Szoftverfejlesztés.* A specifikáció szerint el kell készíteni a szoftvert.
3. *Szoftvervalidáció.* Validálni kell a szoftvert, hogy biztosítsuk azt, hogy azt készítettük el, amit a vásárló megrendelt.
4. *Szoftverevolúció.* Fejleszteni kell a szoftvert a vásárló igényeinek megfelelően.

Ezeket a tevékenységeket a különböző szoftverfolyamatok különféleképpen szervezik, és különböző szintjeiken részletezik őket. Másként ütemezik a tevékenységeket, mint ahogy a tevékenységek eredményeit is. A különböző szervezetek különféle folyamatokat használhatnak ugyanazon típusú termék előállítására. Azonban vannak folyamatok, melyek jobban megfelelnek bizonyos típusú alkalmazások előállításához, mint mások. Ha nem megfelelő folyamatot alkalmazunk, nagy valószínűséggel csökkenni fog a fejlesztendő szoftvertermék minősége vagy használhatósága.

A 4. fejezet nagyobb részletességgel tárgyalja a szoftverfolyamatokat, a 28. fejezet pedig egy fontos témát, a szoftverfolyamatok tökéletesítését taglalja.

1.1.6. A szoftverfolyamat modellje

A szoftverfolyamat modellje a szoftverfolyamat egy bizonyos perspektívából adódó egyszerűsített leírása. A modellek – természetükből adódóan – egyszerűsítések, azaz a szoftverfolyamat modellje nem más, mint az aktuális folyamat egy meghatározott absztrakciója. A folyamatmodellek tevékenységeket foglalhatnak magukban, melyek a szoftverfolyamat, a szoftvertermék és a szoftvertervezésben részt vevő emberek szerepköreiből állnak. Íme, néhány példa az előállítható szoftverfolyamat-modellek típusaira:

1. *A munkafolyamat-modell.* Ez a modell a tevékenységek folyamatbeli sorrendiségét mutatja azok bemeneteivel, kimeneteivel és függőségeikkel. Ebben a modellben a tevékenységek emberi cselekményeket reprezentálnak.
2. *Az adatfolyam- vagy tevékenységmodell.* Ez a modell a folyamatot tevékenységek halmazaként reprezentálja, melyek mindegyike valamilyen adattranszformációt hajt végre. Azt mutatja be, hogyan alakul át a folyamat bemenete (például a specifikáció) kimenetté (például tervvé). Itt a tevékenységek alacsonyabb szintűek is lehetnek, mint a munkafolyamat-modellben, emberek vagy számítógépek által végrehajtandó transzformációkat reprezentálhatnak.
3. *A szerepkör-/cselekvésmodell.* Ez a modell a szoftverfolyamatban részt vevő emberek szerepköreit és felelősségük alá tartozó tevékenységeiket mutatja be.

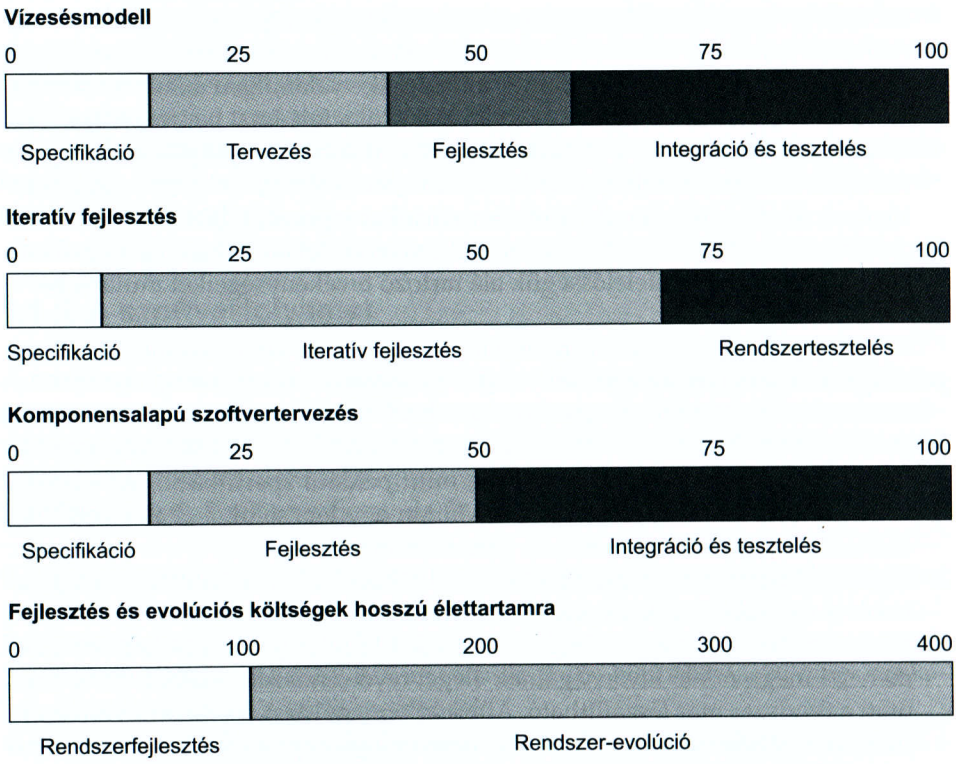
A legtöbb szoftverfolyamat-modell az alábbi három általános modell, illetve paradigma valamelyikére épül:

1. *A vízesés megközelítési mód.* Ez veszi a fenti tevékenységeket, és a folyamat különálló fázisaiként reprezentálja őket, mint például specifikáció, szoftvertervezés, implementáció, tesztelés stb. Miután egy lépcsőfok a „befejezett" státusba ért, a fejlesztés a következő lépcsőfokon folytatódik.
2. *Iteratív fejlesztés.* Ez a megközelítési mód összefésüli a specifikáció, a fejlesztés és a validálás tevékenységeit. Nagyon absztrakt specifikációból a kezdeti rendszer igen gyorsan kifejleszthető. A vásárló finomításai után ez lesz a bemenet a megrendelő kívánságainak eleget tevő rendszer előállításához. Ezek után a rendszer már leszállítható. Alternatív megoldásként valamilyen strukturáltabb megközelítési módot alkalmazva újraimplementálható robusztusabb és könnyebben karbantartható rendszerré.
3. *Komponensalapú szoftvertervezés (Computer-based software engineering, CBSE).* Ez a technika feltételezi, hogy a rendszer részei már léteznek. A rendszerfejlesztési folyamat ezen részek integrálására koncentrál, nem pedig a teljes fejlesztésre. A CBSE-modellel a 19. fejezetben foglalkozom.

A 4. és a 17. fejezetben még visszatérek ezekre az általános folyamatmodellekre.

1.1.7. A szoftvertervezés költsége

Erre a kérdésre nincs egyszerű válasz, mint ahogy a költség szoftverfolyamaton belüli pontos eloszlására sem, ugyanis ezek függnek az alkalmazott folyamattól és attól, hogy milyen típusú szoftvert fejlesztünk. Például a valós idejű szoftverek validálása és tesztelése általában sokkal költségesebb, mint a webes rendszereké. Mindazonáltal minden egyes különböző általános szoftverfejlesztési megközelítési mód különböző profillal rendelkezik a szoftverfolyamat tevékenységeinek költségeit illetően. Amennyiben egy összetett rendszer fejlesztésének teljes költségét 100 egységnek tekintjük, úgy a költségek eloszlása az 1.2. ábra szerinti módon alakulhat a különböző esetekben.



1.2. ábra. A szoftvertervezési tevékenységek költségeinek eloszlása

A vízesésmodell esetében a specifikáció, tervezés és implementáció költségét külön mértük. Jegyezzük meg, hogy a legdrágább fejlesztési tevékenység a rendszer-integráció és a tesztelés. Normális esetben ez körülbelül 40 százaléka a teljes fejlesztési költségnek, bár bizonyos kritikus rendszerek esetében megközelítheti akár a teljes fejlesztési költség 50 százalékát is.

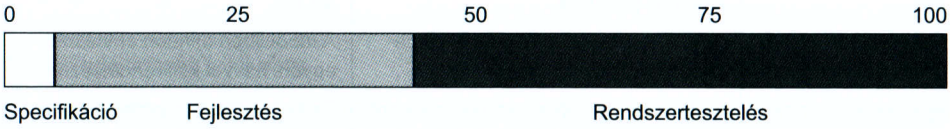
Amennyiben a szoftvert iteratív megközelítési mód szerint fejlesztjük, úgy nincs erős határ a specifikáció, a tervezés, valamint az implementáció között. A specifi-

káció költségei csökkentek, ugyanis ebben a megközelítési módban csak egy első szintű specifikáció készül el. A tényleges specifikáció, tervezés, implementálás, integráció és tesztelés egy fejlesztési tevékenységen belül párhuzamosan kerül végrehajtásra. Ezen túl még szükséges egy különálló tesztelési tevékenység a kezdeti implementáció végén.

A komponensalapú szoftvertervezés széles körben még csak rövid ideje van használatban. Ezért nincs pontos ábránk arról, hogyan oszlik el a költség az ilyen szoftverfejlesztés különféle tevékenységei között. Mindazonáltal annyit tudunk, hogy a fejlesztési költségek relatíve csökkennek az integráció és a tesztelés költségeihez képest. Az integráció és a tesztelés költségei pedig azért emelkednek, mert biztosítani kell, hogy az alkalmazott komponensek megfelelnek a specifikációjuknak, és a többi komponenssel együtt az elvárásoknak megfelelően működnek.

A fejlesztési költségek tetejében még a bevezetés után történő szoftverváltoztatások esetében is újabb költségek merülnek föl. Számos hosszú élettartammal rendelkező szoftverrendszer esetében az ilyen költségek akár háromszor vagy négyszer is meghaladhatják a fejlesztési költséget.

A költségeloszlások az olyan megrendelt szoftverek esetén állnak fenn, melyeket a szállító a vásárló specifikációi alapján fejleszt. Azon szoftvertermékek esetében, melyeket legtöbbször PC-khez adnak el, a költségprofil várhatóan különbözni fog. Az ilyen termékeket általában valamilyen külső specifikáció szerinti evolúciós megközelítési módot alkalmazva fejlesztik. A specifikációs költségek hozzávetőlegesen alacsonyak. Mindamellett mivel ezeket különféle konfigurációk széles skálájára szánták, alapos tesztelésnek kell őket alávetni. Az 1.3. ábra az ilyen termékeknél várható költségprofil típusát mutatja be.



1.3. ábra. Termékfejlesztési költségek

Az általános szoftvertermékek evolúciós költségei elég nehezen becsülhetők. A legtöbb esetben a terméknek rövid szabályszerű evolúciója van. Amint a termék valamely verziója kibocsátásra kerül, megindulnak a következő kiadás munkálatai. Marketing okokból ez utóbbi nagy valószínűséggel egy új, de az előzővel kompatibilis termékként jelenik meg inkább, és nem a felhasználó által már megvásárolt termék módosított változataként. Így az evolúciós költségek külön-külön nem becsülhetők meg, mint a megrendelésre készülő szoftverek esetében, hanem a rendszer következő verziójának fejlesztési költségeként jelentkeznek.

1.1.8. Szoftvertervezési módszerek

Egy szoftvertervezési módszer nem más, mint a szoftverfejlesztés strukturált megközelítése, melynek célja, hogy elősegítse a jó minőségű szoftverek költséghatékony előállítását. Az olyan módszereket például, mint a Structured Analysis (DeMarco, 1978) vagy a JSD (Jackson, 1983) az 1970-es években fejlesztették ki először. Ezek a módszerek a rendszerek alap funkcionális komponenseit próbálták meg felismerni; napjainkban is széles körben használatosak. Az 1980-as, 1990-es években objektumorientált módszerekkel egészítették ki őket, mint ahogy azt Booch (1994) és Rumbaugh (Rumbaugh *és mások*, 1991) javasolta. Ezek a különböző megközelítési módok mára a Unified Modeling Language (UML) (Fowler és Scott, 1997; Booch *és mások*, 1999; Rumbaugh *és mások*, 1999a, 1999b) köré épült egyetlen egységes megközelítési módba integrálódtak.

Nincs ideális módszer, a különféle módszerek különféle területeken alkalmazhatók. Például az objektumorientált módszerek legtöbbször az interaktív rendszerekhez megfelelők, de nem azok a szigorú valós idejű követelményeknek eleget tevő rendszerekhez.

Minden módszer a rendszer fejlesztési modelljének ötletén alapszik, melyet grafikusan reprezentál, és a rendszer-specifikáció, valamint a tervezés területén vet be. A módszerek számos különböző komponenst tartalmazhatnak (1.4. ábra).

Komponens	Leírás	Példa
Rendszermodell leírásai	A fejlesztendő rendszer modelljének leírása és a modellek definiálására használt jelölések.	Objektummodellek, adatfolyam modellek, állapotátmenet-modellek stb.
Szabályok	A rendszermodellre mindig alkalmazandó megszorítások.	A modellben minden entitásnak egyedi névvel kell rendelkeznie.
Ajánlások	Jó tervezési szokásokat meghatározó heurisztikák a módszerben. Ezeket az ajánlásokat követve jól szervezett rendszermodellhez juthatunk.	Egy objektumhoz hétnél több alárendelt objektum nem kapcsolható.
Folyamatirányítás	A rendszermodell fejlesztése alatt követendő tevékenységek leírása, illetve azok szervezése.	Az objektumok attribútumait dokumentálni kell az objektumhoz kapcsolódó műveletek definiálása előtt.

1.4. ábra. Módszerkomponensek

1.1.9. Mi a CASE?

A CASE a Computer-Aided Software Engineering (számítógéppel támogatott szoftvertervezés) rövidítése. Valójában számos különféle programot takar, melyek a szoftverfolyamat tevékenységeit hivatottak támogatni, mint például a követelményelemzést, a rendszermodellezést, a nyomkövetést és a tesztelést.

A CASE-technológiához kapcsolódnak a módszer jelölésrendszeréhez tartozó szerkesztőprogramok, a rendszermodellt a módszer szabályai szerint ellenőrző elemzőmodulok és a rendszer-dokumentáció elkészítésében segítő jelentésgenerátorok is. A CASE-eszközök kódgenerátort is tartalmazhatnak, amely a rendszermodellből és a folyamatirányításból automatikusan generálja a forráskódot. Esetünkben a folyamatirányítás tanácsokat ad a szoftvertervezőnek, hogy mi legyen a következő lépés.

1.1.10. A jó szoftver tulajdonságai

A szoftvertermékeknek az általuk támogatott szolgáltatásokon túl számos egyéb kapcsolódó tulajdonságaik is léteznek, melyek tükrözik a szoftver minőségét. Ezek a tulajdonságok nincsenek közvetlen kapcsolatban a szoftver által elvégzett tevékenységgel. Sokkal inkább tükrözik annak működés alatti viselkedését, szerkezetét, a forráskód szervezését és a kapcsolódó dokumentációt. Ezek a gyakran nemfunkcionális tulajdonságoknak nevezett tulajdonságok például a szoftver válaszideje egy felhasználói kérdésre vagy például a programkód érthetősége.

A tulajdonságok jellegzetes halmaza, amit egy szoftvertől elvárhatunk, nyilvánvalóan függ magának a szoftvernek az alkalmazásától. Következésképpen például egy banki rendszernek biztonságosnak kell lennie, egy interaktív játéknak könnyen irányíthatónak, egy telefonkapcsoló rendszernek megbízhatónak stb. Ezek a tulajdonságok általánosíthatók és az 1.5. ábra által mutatott halmazokba gyűjthetők, melyekről úgy hiszem, hogy egy jól tervezett szoftverrendszer lényeges tulajdonságai.

Terméktulajdonság	Leírás
Karbantarthatóság	A szoftvert oly módon kell megírni, hogy fokozatosan követni tudja az ügyfél igényeinek változásait. Ez elég kritikus tulajdonság, ugyanis az üzleti környezet változásának elkerülhetetlen következménye a szoftver változása.
Üzembiztonság	A szoftver üzembiztonsága valójában tulajdonságok egy sora, mely magában foglalja a megbízhatóságot, a biztonságosságot és a védelmet. Az üzembiztos szoftver rendszerösszeomlás esetén nem okozhat fizikai vagy gazdasági kárt.
Hatékonyság	A szoftver nem pazarolhatja a rendszer-erőforrásokat, mint például a memóriát vagy a processzoridőt. A hatékonyság magában fogalja az alkalmazkodást, számítási időt, memóriakihasználást stb.
Használhatóság	A szoftvernek használhatónak kell lennie anélkül, hogy a szoftverrel megcélzott felhasználói rétegnek indokolatlan erőfeszítéseket kellene tennie ennek érdekében. Ez azt jelenti, hogy megfelelő felhasználói interfésszel és a szükséges dokumentációval kell rendelkeznie.

1.5. ábra. A jó szoftver lényeges tulajdonságai

1.1.11. A szoftvertervezés főbb kihívásai

A 21. században a szoftverfejlesztésnek az alábbi három főbb kihívással kell szembenéznie:

1. *A heterogenitás kihívása.* A rendszerek egyre inkább igénylik a hálózaton keresztüli, több különböző számítógépen több különféleképpen támogatott rendszeren történő osztott működés lehetőségét. Gyakran szükséges az új szoftverek más nyelven implementált meglévő rendszerekhez történő integrálása is. A heterogenitás kihívásának megoldásra váró feladata olyan technikák fejlesztése, amelyek segítségével üzembiztos, a heterogenitás kezeléséhez megfelelően flexibilis szoftverek építhetők.
2. *A szállítás kihívása.* A hagyományos szoftvertervezési technikák többsége igen időigényes. Hosszú idő szükséges a kívánt szoftverminőség eléréséhez. Az üzleti élet napjainkban gyors, így az őket támogató szoftvereknek rugalmasan és gyorsan kell változniuk. A szállítás kihívása azt jelenti, hogy rövidíteni kell azt az időt, amely alatt a nagy összetett rendszerek minőségbeli kompromisszumok nélkül leszállíthatókká válnak.
3. *A bizalom kihívása.* Ahogy a szoftverek egymásba fonódnak az életünk egyéb aspektusaival, szükségessé válik, hogy megbízzunk bennük. Ez különösképpen igaz az olyan távoli szoftverrendszerekre, melyeket weblapokon vagy webszolgáltatási interfészeken keresztül érünk el. A bizalom kihívása azt jelenti, hogy olyan technikákat kell kifejleszteni, melyek demonstrálják, hogy a szoftverekben megbízhatnak azok felhasználói.

Természetesen ezek a kihívások nem függetlenek egymástól. Például egy meglévő rendszer gyors megváltoztatásához biztosítani kell a hálózaton keresztüli elérhetőséget. Hogy megfeleljünk ezeknek a kihívásoknak, szükségünk van új eszközökre és technikákra csakúgy, mint a meglévő szoftvertervezési módszerek kombinálására és új módon történő használatára.

1.2. SZAKMAI ÉS ETIKAI FELELŐSSÉG

Mint más tervezőmérnököknek, a szoftvertervezőknek is adott jogi és szociális keretek között kell dolgozniuk, melyek limitálják szabadságukat. El kell fogadniuk, hogy a munkájuk az egyszerű technikai ismeretek alkalmazásán túl tágabb kötelezettségeket foglal magában. A szoftvertervezők nyilvánvalóan kötelesek szakemberekhez méltóan etikailag és morálisan is felelősségteljesen viselkedni.

Nem használhatják fel tisztességtelen módon szaktudásukat és képességeiket, rossz hírét keltve a szoftvertervezői hivatásnak. Mindamellett léteznek az életnek olyan területei, ahol az alkalmazható szabványos viselkedési formák nem szabályozottak törvényileg, csak a szakmai felelősség kézzel nem fogható nézetei alapján. Nézzünk ezekből néhányat.

1. *Bizalmasság.* Tisztelnünk kell munkaadóink, illetve ügyfeleink bizalmát függetlenül attól, hogy formálisan aláírták-e a titoktartási nyilatkozatot vagy sem.
2. *Hozzáértés.* Nem tüntethetjük fel hamis színben hozzáértésünket, tudatosan nem fogadhatunk el olyan munkát, mely meghaladja képességeinket.
3. *Szellemi tulajdonjogok.* Ismernünk kell a szellemi tulajdonokra, például szabadalmakra, szerzői joggal védett szellemi termékekre stb. vonatkozó helyi szabályzásokat. Biztosítanunk kell, hogy a munkáltatók és kliensek szerzői jogai ne sérüljenek.
4. *Számítógépes visszaélés.* Nem használhatjuk fel technikai szaktudásunkat arra, hogy mások számítógépeivel visszaéljünk. A számítógépes visszaélések skálája a relatíve triviális visszaélésektől (számítógépes játékok a munkaadók gépein) akár a szélsőségesen komoly esetekig terjedhet (vírusterjesztés).

A szakmai társaságok és intézmények fontos szerepet játszanak ebben. Az olyan szervezetek, mint például az ACM, az IEEE (Institute of Electrical and Electronic Engineers), valamint a British Computer Society nyilvánosságra hoztak egy szakmai magatartási, illetve etikai szabályzatot. Ezen szervezetek tagjai vállalják, hogy követik a szabályzat előírásait. Ezek a magatartási szabályzatok átfogóan foglalkoznak az alapvető etikus viselkedéssel.

Az ACM és az IEEE összefogott egy együttes etikai és szakmai gyakorlati előírások létrehozása érdekében. Az előírás-gyűjteménynek létezik egy rövidebb (1.6. ábra), illetve egy hosszabb formája is (Gotterbarn *és mások*, 1999), mely részletezi és pontosítja a rövidebb verziót. Ezen előírás-gyűjtemény értelme összegezve megtalálható a hosszabb változat első két paragrafusában:

> *A számítógépeknek nagy és egyre növekvő, központi szerepe van a kereskedelemben, az iparban, a kormányzatban, az egészségügyben, az oktatásban, a szórakoztatásban és a társadalomban általában. A szoftvertervezők azok, akik ebben közreműködnek közvetlen részvétellel, tanítással, és részt vesznek a szoftverrendszerek elemzésében, specifikációjában, karbantartásában és tesztelésében. A szoftverrendszerek fejlesztésében betöltött szerepük által a szoftvertervezőknek rengeteg lehetőségük adódik arra, hogy jól dolgozzanak vagy kárt okozzanak, lehetőségük adódik arra is, hogy megengedjék másoknak a károkozást, illetve befolyásolják őket rossz vagy jó irányban. Azért, hogy a lehető legnagyobb mértékben biztosítsák, hogy erőfeszítéseik jó cél érdekében legyenek felhasználva, a szoftvertervezőknek el kell kötelezniük magukat amellett, hogy a szoftvertervezést hasznos és megbízható hivatássá tegyék. Az állásfoglalásukkal összhangban a szoftvertervezőknek ragaszkodniuk kell a Szoftvertervezői etikai és szakmai gyakorlati előírásokhoz.*
>
> *Az előírás-gyűjtemény nyolc alapelvet tartalmaz a hivatásos szoftvertervezők döntéseivel és viselkedéseivel kapcsolatban, beleértve a szakma gyakorlóit, oktatókat, menedzsereket, rendszergazdákat és vezetőket ugyanúgy, mint a gyakornokokat és a hallgatókat, akik még csak most tanulják a szakmát. Az alapelvek rögzítik az etikailag felelősségteljes kapcsolatokat, melyekben az egyének, csoportok és szervezetek részt vesznek, valamint az ezekben a kapcsolatokban fellépő elsődleges kötelezettségeket. Ezek a kötelezettségek egyrészt a szoftvertervezők emberiességén (különös gondosko-*

Szoftvertervezői etikai és szakmai gyakorlati előírások
ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

BEVEZETŐ
A szabályzat ezen rövid változata törekvéseinket magas absztrakciós szinten összegzi. A teljes verzióban megtalálható cikkelyek részletezik és példákkal illusztrálják, hogyan változtatják meg ezek a törekvések a szoftvertervező szakemberek tetteit. A törekvések nélkül a részletezések szárazak és unalmasak, a részletezések nélkül a törekvések jól hangzanak, de üresek. A törekvések és a részletezések összetartoznak, együttesen alkotják az előírásokat.

A szoftvertervezők elkötelezik magukat, hogy az elemzést, specifikációt, tervezést, fejlesztést, tesztelést és karbantartást hasznos és tiszteletre méltó hivatássá teszik. A közösség egészsége, biztonsága és jóléte érdekében tett állásfoglalásaikkal összhangban a szoftvertervezők betartják az alábbi nyolc alapelvet:

1. NYILVÁNOSSÁG. A szoftvertervezők következetesen a közösség érdekében cselekszenek.
2. ÜGYFÉL ÉS MUNKÁLTATÓ. A szoftvertervezők oly módon cselekszenek, hogy az az ügyfél és a munkáltató érdekeit legjobban szolgálja a közösség érdekeivel összhangban.
3. TERMÉK. A szoftvertervezők biztosítják, hogy az általuk elkészített termékek és a hozzá kapcsolódó módosítások amennyire lehet, megfelelnek a legmagasabb szakmai szabványoknak.
4. MEGÍTÉLÉS. A szoftvertervezők gondoskodnak szakmai nézeteik egységességének és függetlenségének megtartásáról.
5. MENEDZSMENT. A szoftvertervező menedzserek és vezetők aláírják, hogy csatlakoznak a szoftverfejlesztés és karbantartás menedzsment valamilyen etikus megközelítési módjához, és elősegítik azt.
6. HIVATÁS. A szoftvertervezők megőrzik a szakma sértetlenségét, elősegítik jó hírnevét, a közösség igényeit szem előtt tartva.
7. MUNKATÁRSAK. A szoftvertervezők becsületesek és segítőkészek munkatársaikkal szemben.
8. ÖNMAGA. A szoftvertervező szakmája gyakorlását tekintve élete végéig tanul, és támogatja a hivatása gyakorlásának etikus megközelítési módjait.

1.6. ábra. ACM/IEEE etikai kódex (©IEEE/ACM 1999)

dással viseltetve azon emberek iránt, akik magukon viselik a szoftvertervezők munkájának hatását), másrészt a szoftvertervezés gyakorlatának egyedi elemein alapszanak. Ezeket a kötelezettségeket az előírás-gyűjtemény mindenki számára előírja, aki szoftvertervező kíván lenni.

Bármilyen szituációban is, ahol különböző embereknek különböző nézőpontjaik és céljaik vannak, etikai dilemmákkal fogjuk magunkat szembe találni. Például ha az alapelveket tekintve nem értünk egyet a vállalaton belüli legfelsőbb vezetéssel, hogyan kell reagálnunk? Természetesen ez az érintett személyektől és az ellentét jellegétől is függ. Mi a legjobb: ha a saját pozíciónk mellett érvelünk egy szervezeten belül, vagy ha lemondunk az elvekről? Ha úgy érezzük, hogy problémák merültek föl egy szoftverprojekt kapcsán, mikor fedjük fel azokat a

vezetőség előtt? Ha akkor tárgyaljuk meg őket, mikor még éppen csak a problémák gyanúja merült föl, túlreagálhatjuk a szituációt. Ha túl későre halasztjuk, lehetetlenné válhat a nehézségek leküzdése.

Az ilyen etikai dilemmák mindannyiunk életében felmerülnek szakmánk gyakorlása közben, de szerencsére a legtöbb esetben ezek a problémák relatíve kicsik, vagy nagyobb nehézségek nélkül megoldhatók. Amikor a problémák nem megoldhatók, lehetséges, hogy a tervezők egy másik problémával találják szembe magukat. Az elvi alapokon álló tettek és cselekedetek akár arra is késztethetik őket, hogy lemondjanak munkájukról, de ez kihathat akár partnereikre, gyerekeikre is.

A hivatásos tervezők életében különösen nehéz helyzetek merülhetnek föl akkor, ha munkáltatójuk valamit etikátlan módon visz véghez. Mondjuk egy cég felelős kifejleszteni egy biztonságkritikus rendszert, és az idő szorítása miatt meghamisítják a biztonság validálási rekordjait. A tervezőnek kell eldöntenie, hogy megtartja a bizalmas információkat, vagy pedig riasztja az ügyfelet, esetleg nyilvánosságra hozza a tényt, hogy a szállított rendszer nem biztonságos.

Itt a probléma az, hogy abszolút módon nem meghatározható, mikor válik egy rendszer biztonságossá. Nem biztos, hogy a rendszerek az előre definiált kritériumok szerint validáltak, ha ezek a kritériumok túl szigorúak. Mégis, a rendszerek tulajdonképpen akár minden baj nélkül is működhetnek teljes élettartamukon keresztül. Persze az is előfordulhat, hogy a rendszer pontosan validált, mégis meghibásodik és váratlan komplikációkat okoz. A problémák korai leleplezése kárt okozhat a munkáltatónak, illetve a beosztottaknak, míg annak elmulasztása másoknak okozhat károkat.

Saját véleményt kell alkotnunk az ilyen esetekkel kapcsolatban. Ilyenkor a károkozás potenciális lehetőségének, a kár nagyságának és a kár által befolyásolt embereknek hatással kell lenniük a döntésre. Ha a szituáció nagyon veszélyes, akár országos kiadványokon keresztül történő nyilvánosságra hozatala is indokolt lehet. Akárhogy is, mindig arra kell törekednünk, hogy megoldjuk az adott problémát, miközben figyelembe vesszük munkáltatónk jogait.

Ebben a helyzetben fontos, hogy mind a munkaadónak, mind a beosztottnak előre ismernie kell a másik nézeteit. Amennyiben egy szervezet katonai vagy nukleáris munkában vesz részt, úgy meg kell egyezniük, hogy a beosztott bármilyen munkabeosztást elfogad. Ugyanígy, ha a beosztottak viszont felvállalják és tisztázzák, hogy nem kívánnak a számukra kiosztott rendszereken dolgozni, úgy a munkáltató nem kényszerítheti őket arra még egy későbbi időpontban sem.

Az általános etika és a szakmai felelősség területe az egyike azon dolgoknak, amelyek növekvő figyelmet élveznek az elmúlt néhány év óta. Akár filozófiai nézőpontból is nézhetjük a dolgokat. Alapelveknek az etika alapelveit tekintjük, a szoftvertervezői etikát pedig ezen főbb alapelvekre hivatkozva tárgyaljuk meg. Ezt a megközelítési módot Laudon (Laudon, 1995), illetve kisebb mértékben Huff és Martin (Huff és Martin, 1995) alkotta meg.

A magam részéről ezt a megközelítési módot túlságosan absztraktnak és nehézkesnek találom ahhoz, hogy a mindennapi gyakorlatban alkalmazzuk. Nagyobb előnyben részesítem a konkrétabb, viselkedési és gyakorlati előírások for-

májában megtestesült megközelítési módokat. Úgy hiszem, hogy a számunkra szükséges etikai megközelítési mód a szoftvertervezéssel összefüggésben jobban tárgyalható, mint különálló témaként. Ilyen módon ez a könyv nem tartalmaz absztrakt etikai fejtegetéseket, de ahol szükséges, példákat ad arra, mit lehet egy etikai okfejtés alapjának tekinteni.

Kulcsfogalmak

- A szoftvertervezés mérnöki tudományág, mely a szoftver előállításának minden lehetséges aspektusát érinti.
- A szoftvertermék kifejlesztett programokból és hozzájuk kapcsolódó dokumentációkból áll. A termék lényeges tulajdonságai a karbantarthatóság, az üzembiztonság, a hatékonyság és az elfogadhatóság.
- A szoftverfolyamat a szoftvertermék fejlesztésében közreműködő tevékenységekből áll. A magasabb szintű tevékenységek a specifikáció, a fejlesztés, a validálás és az evolúció, melyek minden szoftverfolyamatban megtalálhatók.
- A szoftver előállítása köré szerveződnek a módszerek. Javaslatokat adnak a követendő folyamatra, a használandó jelölésrendszerre, a rendszer meghatározását irányító szabályokra, illetve a tervezési irányokra.
- A CASE-eszközök olyan szoftverrendszerek, melyek a szoftverfolyamatban található szokásos tevékenységek támogatását célozzák meg, ellenőrzik a diagramok konzisztenciáját és nyomon követik az aktuálisan futó program tesztelését.
- A szoftvertervezőknek hatalmas felelősségük van a tervezői hivatás és általában a társadalom területén, s így nem pusztán technikai ügyekkel kell foglalkozniuk.
- A szakmai társadalom előírás-gyűjteményeket hoz nyilvánosságra, hogy tagjaitól milyen szabványos viselkedési és magatartási formákat vár el.

További irodalom

Fundamentals of Software Engineering. Általános szoftvertervezési mű, mely más szemszögből szemléli a szoftvertervezést, mint ez a könyv. (Ghezi, C. *és mások,* 2003, Prentice Hall.)

„Software Engineering: The state of the practice". Az *IEEE Software* speciális kiadása. Számos cikket tartalmaz a szoftverfejlesztés jelenlegi gyakorlatáról és arról, hogyan változott meg, ahogy egyre több új szoftvertechnológiát használunk. (*IEEE Software,* 20 (6), November 2003.)

Software Engineering: An Engineering Approach. Általános mű, mely számos hasznos esettanulmányt tartalmaz. (Peters, J. F. és Pedrycz, W., 2000, John Wiley & Sons.)

Professional Issues in Software Engineering. Kiváló könyv, amely szakmai, jogi és etikai kérdéseket tárgyal. (Bott, F. *és mások,* 3. kiadás, 2000, Taylor & Francis.)

„Software Engineering Code of Ethics is approved". Ez a cikk az ACM/IEEE etikai kódex rövid és hosszú változata fejlesztésének hátterét tárgyalja. (Gotterbarn, D. *és mások, Comm. ACM,* October 1999.)

„No silver bullet: Essence and accidents of software engineering". Kora ellenére ez a cikk a szoftvertervezés problémáinak nagyon jó, átfogó bemutatása. A cikk lényegi mondanivalója, miszerint nincs egyszerű válasz a szoftvertervezés kérdésére, nem változott. (Brooks, F. P., *IEEE Computer,* 20 (4), April 1987.)

Feladatok

1.1. Hivatkozva az 1.1.6. alfejezetben tárgyalt szoftverköltség-eloszlásokra, magyarázza meg, miért helyes a szoftvert többnek tekinteni, mint a rendszer egy végfelhasználó által futtatható programjának.

1.2. Mi a különbség az általános és a testre szabott szoftvertermékek fejlesztése közt?

1.3. Mi a szoftvertermék négy legfontosabb tulajdonsága? Ajánljon négy másik tulajdonságot, melyek szintén jelentősek lehetnek.

1.4. Mi a különbség a szoftverfolyamat modellje és maga a szoftverfolyamat között? Adjon meg két olyan lehetőséget, amelyekben a szoftverfolyamat modellje segítségünkre lehet a folyamat lehetséges továbbfejlesztésének felismerésében.

1.5. Fejtse ki, hogy a rendszer tesztelésének költsége miért jelentősen magas az általános, széles piacra szánt szoftvertermékek esetében.

1.6. A szoftvertervezési módszerek gyakran csak akkor terjednek el szélesebb körben, ha elérhetővé válnak az őket támogató CASE-technológiák. Adjon meg öt olyan lehetőséget, ahol a CASE-eszközök támogathatják a módszereket.

1.7. Eltekintve a meglévő rendszerek kihívásaitól, a heterogenitás és a rendszerek gyors leszállítása egyéb problémákat és megoldásra váró feladatokat is felvethet, melyekkel egy szoftvertervező a 21. században nagy valószínűséggel szembetalálkozhat. Nevezze meg ezeket.

1.8. Vizsgálja meg, lehetséges-e az, hogy a szoftvertervezők ugyanolyan módon kapják meg képesítésüket, mint az orvosok vagy az ügyvédek.

1.9. Illusztrálja megfelelő példákkal az 1.6. ábrán bemutatott ACM/IEEE etikai kódex minden egyes cikkelyét.

1.10. A terrorelhárítók munkájának segítése érdekében számos ország tervez és fejleszt olyan számítógéprendszereket, amelyek alkalmasak a polgárok és azok tevékenységeinek nyomon követésére. Tisztázza ezek hatását a magánéletre. Vitassa meg, etikus-e ilyen típusú rendszerek fejlesztése.