Információs rendszerek tervezésének módszertana

Komló Csaba

MÉDIAINFORMATIKAI KIADVÁNYOK

Információs rendszerek tervezésének módszertana

Komló Csaba

Eger, 2013

Korszerű információtechnológiai szakok magyarországi adaptációja

TÁMOP-4.1.2-A/1-11/1-2011-0021

Lektorálta:
Nyugat-magyarországi Egyetem Regionális Pedagógiai Szolgáltató és Kutató Központ

Felelős kiadó: dr. Kis-Tóth Lajos
Készült: az Eszterházy Károly Főiskola nyomdájában, Egerben
Vezető: Kérészy László
Műszaki szerkesztő: Nagy Sándorné

Tartalom

1. BEVEZETÉS

1.1 CÉLKITŰZÉSEK, KOMPETENCIÁK A TANTÁRGY TELJESÍTÉSÉNEK FELTÉTELEI

1.1.1 Célkitűzés

A hallgató rendelkezzen az információtechnológiai szakemberek számára elengedhetetlen, korszerű informatikai és számítástudományi műveltséggel, ennek részeként ismerje meg az információs rendszerek tervezésének módszertani kérdéseit. A *szakterülethez* kapcsolódóan legyen képes a hatékony és korszerű rendszertervezési folyamatok (rendszerelemzés, rendszerigények meghatározása, rendszerfejlesztési projektmenedzsment, rendszerösszetevők azonosítása, rendszerinterfészek tervezése, rendszertelepítés és -karbantartás) azonosítására, jellemzésére és gyakorlati alkalmazására.

1.1.2 Kompetenciák

- Információs rendszerek tervezése.
- A hallgatók informatikai látókörének szélesítése, készségeinek és képességeinek fejlesztése.
- Az informatikai projektszemlélet kialakulása.
- Absztrakciós képesség fejlődése.
- Rendszerszemlélet kialakulása.

1.1.3 A tantárgy teljesítésének feltételei

Az elméleti ismereteket magába foglaló feladatlap eredményes kitöltése és a gyakorlati feladat elkészítése: informatikai rendszer tervezése és dokumentálása.

1.2 A KURZUS TARTALMA

- Szoftverforrások
- A rendszerfejlesztési projekt menedzselése
- A rendszerfejlesztési projekt tervezésének kezdeti lépései
- A rendszerösszetevők meghatározása és implementáció
- Adatbázis-tervezés

- Adatbevitel és űrlaptervezés
- Interfész- és dialógustervezés
- Rendszerimplementálás és -telepítés
- Rendszerkarbantartás
-

2. LECKE: RENDSZERTERVEZÉS SZERVEZETI KÖRNYEZETBEN

2.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A második fejezet célja, hogy a hallgatók rendelkezzenek információval a rendszertervezés alapfogalmait illetően, ismerjék meg a rendszertervezés rövid történetét, legyenek tisztában az információs rendszerek tervezésének alapvető tudnivalóival és a tervezés ciklusaival. Ismerjék meg a legfontosabb rendszertervezési elveket, rendelkezzenek információval az alábbi metódusokról:

- Vízesés modell
- Evolúciós modell
- Komponensalapú és újrafelhasználható rendszerfejlesztés
- Spirális fejlesztés
- Gyors Alkalmazásfejlesztés (RAD)
- RUP (Rational Unified Process)
- eXtreme Programming
- Agilis szoftverfejlesztés

2.2 TANANYAG

2.2.1 A rendszertervezés alapjai

A rendszertervezés egy komplex folyamat, amelynek a célja egy adott cél elérése érdekében működő információs rendszer tervezése, létrehozása, telepítése és a működési feltételeinek a biztosítása. Az információs rendszerek rendszerint valamilyen vállalat működését segítik elő. Ezért az információs rendszerek tervezése szinte minden esetben szervezeti aspektusból történik, alkalmazkodva a szervezet méretéhez és információs igényeihez. Az optimálisan működő információs rendszer tervezéséhez ezért elengedhetetlen, hogy megértsük a szervezet felépítését és működését és az információs folyamatok lefutását a szervezeten belül.

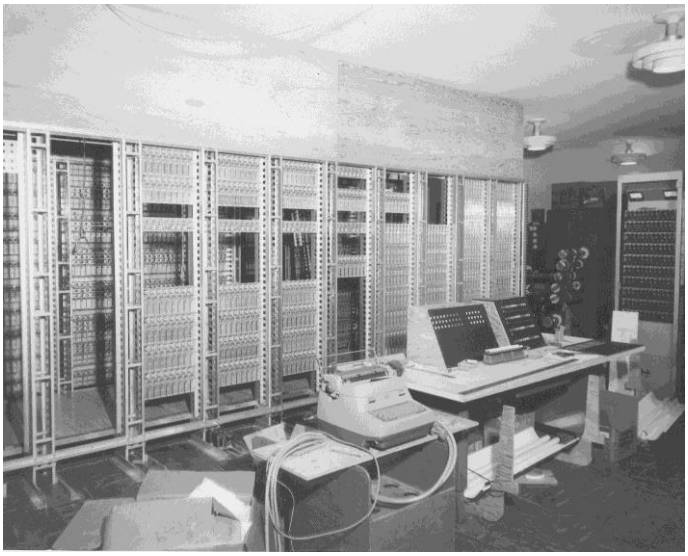
A rendszerelemzési és -tervezési folyamat egyik fontos eredménye egy olyan átfogó számítógépes alkalmazói szoftver megszületése, amely minden területen hatékonyan támogatja a szervezet feladatainak megvalósítását a bérügyek intézésétől kezdve a piackutatásig. Természetesen az információs rend-

szer nem csak ebből a szoftverből áll, hiszen szükség a hardverre, amelyen a szoftverünk fut, illetve a rendszerszoftverre, amely az információs rendszer-applikációnk futtatási környezettét biztosítja. De ide tartozik még a rendszer dokumentációja, a rendszer működtetésének és használatának elősegítését célzó tanfolyamok és a rendszer működtetéséhez szükséges új munkakörök megjelenése is.

Néhány tíz évvel ezelőtt a rendszertervezés minden szervezetnél egy kifinomult, már-már művészi tevékenység volt, ami hosszú időt és egyedi megoldások kifejlesztését igényelte. Azonban az információs rendszerek iránti igények exponenciális növekedésével a rendszertervezési folyamat egyre inkább tudományosan és gyakorlati tapasztalatok által megalapozott, jól körülhatárolható lépések és eljárások sorozatává szelídült, amelyből néhányat szeretnénk könyvünkben bemutatni.

2.2.2 A rendszertervezés története dióhéjban

A számítógép alapú rendszerek tervezése az ötvenes évekig nyúlik vissza. Akkoriban a számítógépek korlátozott teljesítménye miatt elsődleges fontosságú volt, hogy a szoftverek futásuk során hatékonyan használják ki a rendelkezésre álló erőforrásokat. Az ekkor használt számítógépek szobányi méretűek voltak és magas áruk ellenére nem minden esetben működtek megbízhatóan. A feladatok is egyszerűbbek voltak (beszerzés, raktározás stb.), és a rendszer volumene is gyakran egyetlen vállalti osztály feladatainak ellátására korlátozódott.



1. ábra: Számítógép a 60-as évekből

A szoftverek minden esetben gépi kódú vagy assembly alapúak voltak, és mivel még nem létezett a szoftveripar, ezért előre elkészített szoftverek vásárlásáról szó sem lehetett.

```
812 MAP18    DFB  $24,$FE,$1E,$32,$3F
    ,$23,$B2,$53,$12,$5E,$12,$13,$1F,
    $53,$1F,$13,$2F,$6E,$1F,$3E,$33,$
    42,$FE,$FE,$8E,$15,$2E,$22,$3F,$4
    3,$3F,$82,$53,$EE,$53,$12,$BE
813 MAP19    DFB  $22,$FE,$1E,$32,$2F
    ,$23,$D2,$43,$12,$6E,$43,$4F,$13,
    $1F,$CE,$12,$42,$FE,$FE,$8E,$52,$
    1F,$53,$2F,$A2,$43,$32,$1F,$32,$1
    E,$32,$3E,$53,$22,$BE
814 MAP20    DFB  $24,$FE,$1E,$42,$1F
    ,$13,$F2,$23,$22,$6E,$12,$83,$2F,
    $AE,$1F,$23,$42,$FE,$FE,$9E,$42,$
    2F,$33,$2F,$C2,$23,$1F,$19,$12,$4
    F,$12,$1E,$72,$43,$12,$1F,$BE
815 MAP21    DFB  $21,$FE,$2E,$F2,$62
    ,$13,$22,$6E,$22,$73,$12,$2F,$9E,
    $1F,$13,$52,$FE,$FE,$BE,$22,$1F,$
    33,$2F,$D2,$23,$4F,$2E,$2F,$1E,$7
    2,$33,$22,$1F,$BE
816 MAP22    DFB  $22,$FE,$3E,$F2,$72
    ,$8E,$2F,$23,$3F,$23,$12,$1F,$4E,
    $13,$4E,$3F,$42,$FE,$FE,$CE,$2F,$
    13,$1F,$23,$1F,$E2,$33,$12,$2F,$4
    E,$82,$23,$22,$1F,$BE
817 MAP23    DFB  $24,$FE,$4E,$F2,$62
    ,$9E,$4F,$1E,$2F,$13,$2F,$4E,$23,
    $4E,$12,$2F,$22,$FE,$FE,$1E,$1F,$
    CE,$1F,$13,$19,$13,$2F,$E2,$23,$3
    2,$2F,$1E,$A2,$13,$22,$2F,$BE
818 MAP24    DFB  $1F,$FE,$5E,$F2,$42
    ,$FE,$1E,$3F,$5E,$23,$3E,$15,$12,
    $1F,$13,$22,$FE,$FE,$1E,$1F,$13,$
    BE,$1F,$13,$3F,$F2,$62,$1F,$1E,$D
    2,$1F,$CE
819 MAP25    DFB  $1B,$FE,$5E,$B2,$3E
    ,$52,$FE,$9E,$12,$53,$12,$23,$22,
    $FE,$FE,$12,$2F,$13,$4E,$1F,$3E,$
    6F,$F2,$72,$2E,$D2,$1F,$CE
```

2. ábra: Gépi kódú programrészlet

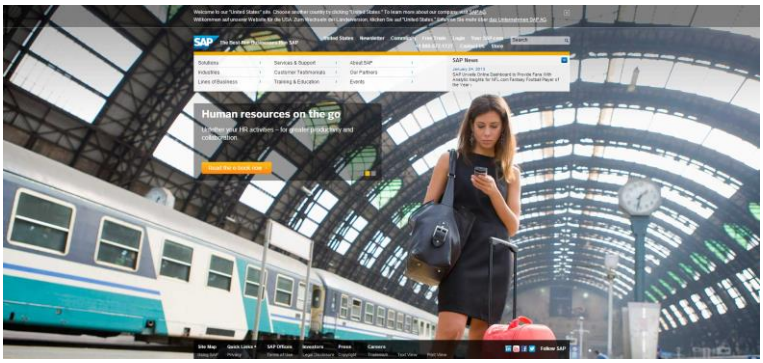
A hatvanas években megjelentek a harmadik generációs, más néven procedurális nyelvek, amelyek egyszerűbbé tették a programozást. A hardver tekintetében meg kell említeni, hogy a nagy és drága gépek mellett az évtized végén megjelennek a mini számítógépek. Annak ellenére, hogy a legtöbb cég még saját maga fejleszti az információs rendszerét működtető szoftverét, a

szoftveripar már bontogatni kezdi szárnyait, és egyre többen dolgoznak azon, hogy a rendszerfejlesztés tudományosan és gyakorlati tapasztalatokra alapozott diszciplínává váljon.

3. ábra: IBM PC

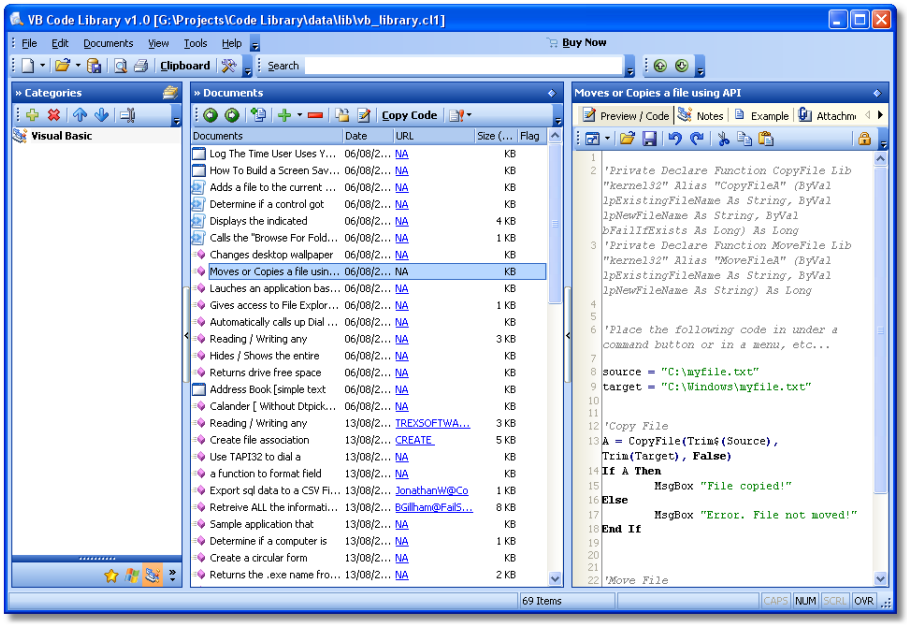
A nyolcvanas évek több áttörést is hozott a rendszerfejlesztésben. Egyrészt megjelentek a viszonylag olcsónak számító mikroszámítógépek, amelyek egyre gyakrabban számítottak a számítógépes rendszerek egyik legfontosabb építőelemének. Másrészt a negyedik generációs nyelvek megjelenése egyszerűbbé tette a programozást, és egyre többen írtak szoftvereket értékesítési szándékkal a legkülönfélébb feladatok ellátására.

A kilencvenes években a fő trend a rendszerfejlesztésben a rendszerintegráció volt. A szoftverfejlesztők egyre gyakrabban használtak grafikus fejlesztőkörnyezetet a felhasználói interfész (pl. Visual Basic) fejlesztésére. A szoftverek többnyire kliens-szerver architektúrára épültek és a szervereken olyan gyártók programjai futottak, mint pl. az Oracle, Microsoft, Ingres stb.), de lehetőség volt különféle modulokból álló teljes vállalati szoftverrendszerek megvásárlására is (pl. SAP).



4. ábra: Az SAP cég weboldala

A kilencvenes évek közepétől egészen napjainkig egyre több szoftverfejlesztő kezdi kiaknázni az internet lehetőségeit, és azon belül is elsősorban a world wide webet. A vezeték nélküli hálózatok és mobilinformatikai eszközök egyre szélesebb körű terjedése pedig a „bárhol-bármikor” szemléletét csempészi be a számítógépes információs rendszerek tervezésébe. További újdonság, hogy a vállalatok a költséges szerverpark megvásárolása és üzemeltetése helyett egyre gyakrabban felhőalapú szolgáltatásokat vesznek igénybe információs rendszereik működtetésénél.



5. ábra: Visual Basic programrészlet

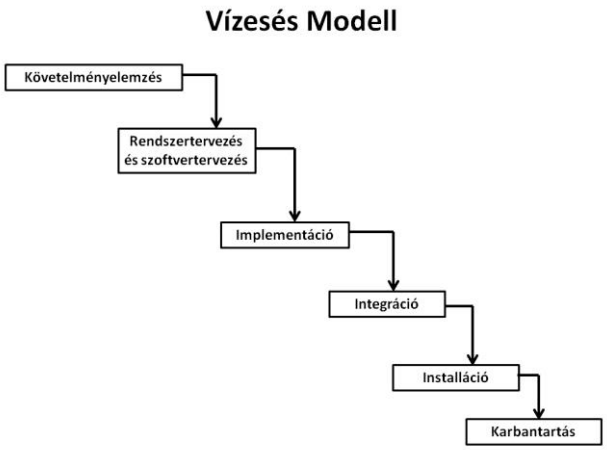
2.2.3 Információs rendszerek tervezése és a tervezés ciklusai

A legtöbb szervezet nagyon fontosnak tartja, hogy a rendszertervezési folyamat általánosan elfogadott módszertanának megfelelően alakítsa ki információs rendszerét. Ahogyan a termékek és szolgáltatások, úgy az információs rendszerek is jól meghatározható életciklussal rendelkeznek. Az információs rendszerek életciklusai globálisan gyakran jellemezhetőek egy olyan folyamattal, amelyben a szervezet első információs rendszerének tervezése az első lépcső, amelyet a rendszer létrehozása, implementálása, üzemeltetése majd karbantartása követ, és ahol az említett rendszer életciklusa végén már az új rendszer bevezetésének előkészítése történik meg, amelynek a tervezési folyamata már a régi rendszer működtetésével párhuzamosan megkezdődött.

Az információs rendszerek tervezésére számos modellt fejlesztettek ki az utóbbi néhány évtizedben.

2.2.4 A vízesés modell

A vízesés modell az első publikált rendszerfejlesztési modell, a hetvenes években Winston W. Royce publikálta „Managing the Development of Large Software Systems” címen. A vízesésmodell egy szekvenciális fejlesztési modell, amely jól elkülönülő lépésekre osztja a rendszertervezés egymás utáni lépcsőit (innen származik az elnevezés, bár az eredeti publikációban még nem így nevezték).



6. ábra: Vízesés Modell

A modell lényege, hogy a következő lépcsőfokra nem léphetünk, amíg az előző fázis be nem fejeződött. A gyakorlatban természetesen van lehetőség az előző lépcsőfokra visszatérni (ha funkcionálisan nem kielégítő az eredmény), illetve a lépcsőfokok részben átfedhetik egymást. A lépcsőfokok végét többek között a dokumentáció elkészülte is jelzi. Ha vissza kell térni egy korábbi lépcsőfokhoz, az gyakran az adott lépcsőfok (és ebből adódóan az összes többi) tervezésének újrakezdését jelentheti.

A vízesés modell előnye, hogy a részletes dokumentáció következtében a rendszer átgondolt, redundáns elemektől mentes, könnyen karbantartható lesz. A modell hátránya, hogy a követelményeket a rendszertervezés elején pontosan meg kell határozni, és bármilyen változás a tervezési folyamat újrakezdését jelentheti. Vannak olyan kritikai vélemények, amelyek szerint ez a metódus (csak azután továbblépni a következő lépcsőfokra, miután az előző tökéletesen elkészült) csak triviális rendszerek esetén alkalmazható, a valóságban nem.

2.2.5 A vízesés modell lépcsői

- Követelményelemzés: Milyen követelményeknek kell megfelelnie a rendszernek, milyen szolgáltatásokat kell nyújtania a felhasználó felé.
- Rendszertervezés és szoftvertervezés: Ebben a szakaszban szétválasztjuk a hardver- és a szoftverkomponensek funkcióit.
- Implementáció: A szoftverkomponensek létrehozása, és a komponensek funkcionális ellenőrzése.
- Integráció: a különálló komponensek integrálása egységes rendszerré és a működés globális tesztelése.
- Installáció: a rendszer üzembe helyezése.
- Karbantartás: a rendszer működési feltételeinek biztosítása.

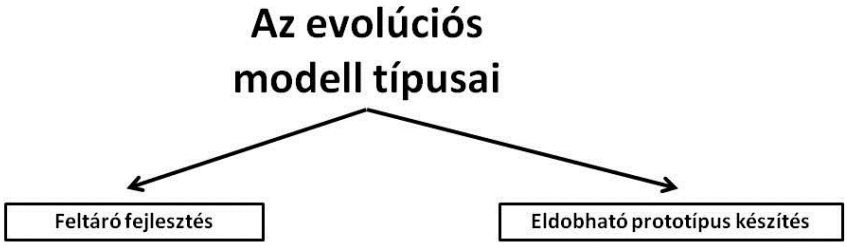
2.2.6 Evolúciós modell

Az evolúciós rendszerfejlesztési modell lényege, hogy a fejlesztők létrehoznak egy kezdeti információs rendszert, majd azt a felhasználókkal véleményeztetik, majd sok-sok verzión keresztül addig finomítják, amíg a minden igényt kielégítő rendszert el nem érik.

Az evolúciós fejlesztésnek két különböző típusa ismert:

Feltáró fejlesztés: A felhasználóval történő egyeztetés után az adott alapkövetelményeknek megfelelő működőképes rendszert adnak át a felhasználóknak. A rendszer továbbfejlesztése során az alaprendszer működését látva a

felhasználók újabb és újabb követelményeket határoznak meg a rendszer fejlesztői csoport számára, közösen kialakítva a rendszer végleges felépítését.



7. ábra: Evolúciós modell

Eldobható prototípus készítése: A fejlesztés alapja ebben az esetben a felhasználó elvárásainak minél pontosabb megismerése, amelyekre alapozva pontosan definiálhatók azok a tulajdonságok, amelyeket a rendszernek tudnia kell. Azokat a követelményeket, amelyek nem érthetőek teljesen pontosan, egy-egy prototípusban valósítják meg, és a felhasználó véleménye alapján finomodik a követelmények meghatározása és a rendszer funkcionalitása.

A módszer hátránya, hogy nehezen átlátható a fejlődési folyamat és a változatok nagy száma miatt a pontos dokumentáció szinte lehetetlen, ami megnehezíti az üzemeltetési és karbantartási feladatokat.

2.2.7 Komponensalapú és újrafelhasználható rendszerfejlesztés

A komponensalapú rendszerfejlesztés alapja az, hogy szoftverek jelentős részében ismétlődnek egyes szoftverkomponensek. Az újrafelhasználás során egy már elkészült rendszer egyes komponenseit az új feladatnak megfelelően átalakítják, és így kerülnek az új rendszerbe, amely jelentősen felgyorsíthatja a rendszerfejlesztés menetét. A fejlesztés gyorsasága és a követelményeknek való megfelelés szoros kapcsolatban van egymással, optimális esetben viszonylag gyors fejlesztési folyamat is eredményezhet funkcionálisan jól működő rendszert.

8. ábra: Komponensalapú és újrafelhasználható rendszertervezés

Ez a rendszerfejlesztési mód tehát arra épül, hogy a rendszerkövetelményeknek milyen mértékben tudjuk megfeleltetni, és egységes rendszerbe integrálni a rendelkezésre álló szoftverkomponenseket. A folyamat lépései a következők:

Komponenselemzés: A rendszerkövetelmények alapján a rendelkezésre álló szoftverkomponensek közül megkeressük azokat, amelyek teljesíthetik az elvárásokat.

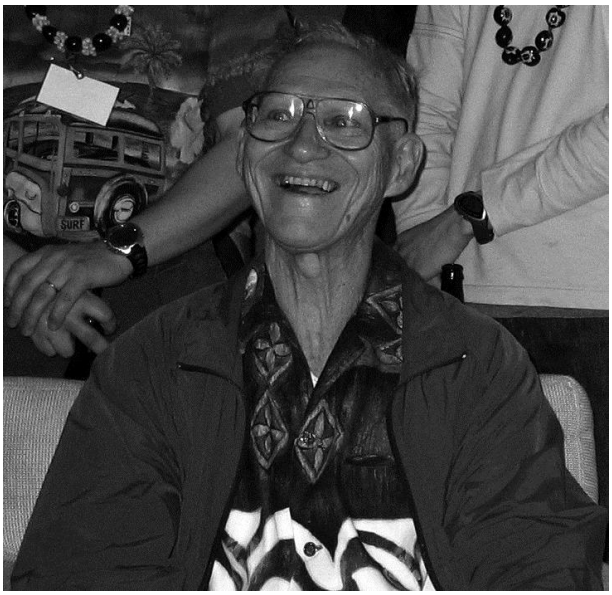
Követelménymódosítás: A szoftverkomponensek funkcionális leírását felhasználva megfeleltetjük az elvárásokat a rendelkezésre álló komponenseknek. Azokban az esetekben, amikor az elvárásnak nem felel meg teljes egészében a rendelkezésre álló komponens, meg kell vizsgálni, hogy a követelmény módosítható-e olyan mértékben, hogy a rendszer integritása megmaradjon, és megfeleljen a komponens funkciójának. Ha ez nem lehetséges, akkor a komponenst kell módosítani, vagy új szoftverkomponenst kell kifejleszteni.

Rendszertervezés a komponensek újrafelhasználásával: ebben a szakaszban összeállítjuk a rendszer elemeit, a korábban már definiált újrafelhasználható komponenseket beépítjük a rendszerbe, a hiányzó elemek funkcióját pedig a fejlesztés számára dokumentáljuk.

Az utolsó lépésben a hiányzó elemek fejlesztése, rendszerbe integrálása és a rendszer tesztelése történik.

2.2.8 Spirális fejlesztés

A spirális fejlesztési modellt Barry Boehm amerikai matematikus publikálta a nyolcvanas évek végén „A Spiral Model of Software Development and Enhancement” című cikkében. A modell alapgondolata két dologban is eltér az eddigiektől: egyrészt a szoftverfejlesztési folyamatot nem egyirányú tevékenységfolyamatnak tekinti, hanem egy spirálként reprezentálja. A spirál részei a szoftverfejlesztési folyamat egy-egy ismétlődő fázisát reprezentálják.



9. ábra: Barry Boehm

A másik jelentős különbség, hogy a modell kiemelten foglalkozik a fejlesztés kockázati tényezőinek felmérésével, a kockázatból származó problémákra való felkészüléssel. A spirál minden egyes ciklusát négy szektorra oszthatjuk fel:

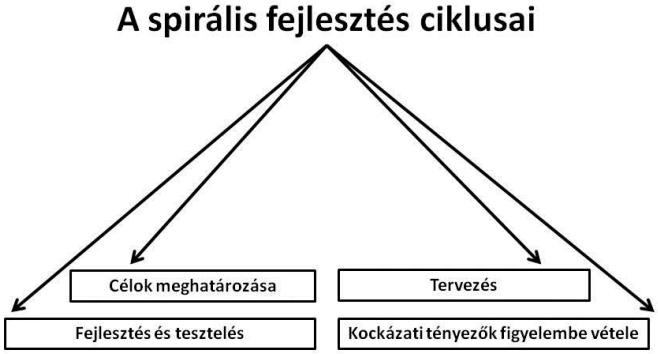
1. Célok meghatározása: az adott fázis által kitűzött célok meghatározása.

2. Kockázati tényezők figyelembe vétele: minden egyes felismert kockázati tényező esetén lépéseket kell tenni a kockázat csökkentése érdekében.

3. Fejlesztés és tesztelés: a kockázat kiértékelése után egy fejlesztési modellt kell választani a problémának megfelelően. Adott esetben minden egyes

spirálban más-más fejlesztési modell alkalmazható, a kockázatanalízis eredményeként választható a fejlesztési modell.

4. Tervezés: a tervezési folyamat során ekkor kell eldönteni, hogy folytatódjon-e a fejlesztési folyamat egy következő ciklussal, vagy sem. Ha a folytatás mellett döntünk, akkor kezdődik a következő kör a célok meghatározásával.



10. ábra: A spirális fejlesztés ciklusai

A modell egy tipikus alkalmazási területe a számítógépes játékok fejlesztésének iparága.

2.2.9 Gyors Alkalmazásfejlesztés (RAD)

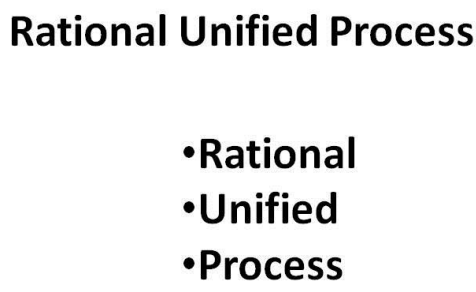
A Rapid Application Development (RAD) egy iteratív, folyamatalapú szoftverfejlesztési modell, mely rendkívül gyors fejlesztési folyamatot tesz lehetővé. A módszert James Martin publikálta 1990-ben, New Yorkban megjelent Rapid Application Development című könyvében.

11. ábra: A gyors alkalmazásfejlesztés (RAD) fejlesztés elemei

A módszer elemei: ciklikus fejlesztés, működő prototípusok létrehozása, és a szoftverfejlesztést támogató számítógépes programok, például integrált fejlesztői környezetek használata. A gyors alkalmazásfejlesztés hagyományosan rendszerint kompromisszumokkal jár a használhatóság, tulajdonságok és a program futási sebessége terén, de ezt ebben az esetben ellensúlyozza a jelentősen lecsökkent fejlesztési idő, a felhasználói igények magas fokú kielégítése, és a grafikus felhasználói felület.

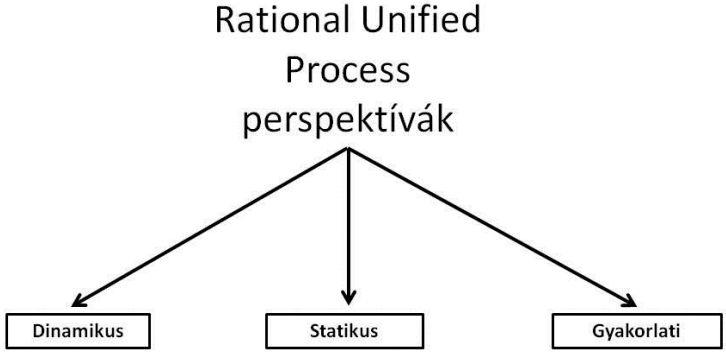
2.2.10 RUP (Rational Unified Process)

A Rational Unified Process (RUP) szoftverfejlesztési eljárást az IBM által felvásárol Rational Software Corporation fejlesztette ki.



12. ábra: A RUP fejlesztési eljárás elemei

A folyamatközpontú RUP nem szigorúan végrehajtandó egymás utáni eljárások sorozata, sokkal inkább egy flexibilis keretet a fejlesztési folyamat kézbentartásához.



13. ábra: RUP a rendszerfejlesztési folyamat

A RUP a rendszerfejlesztés folyamatát alapvetően három dimenzióval írja le:

1. dinamikus perspektívával, amely a modell fázisait mutatja;

2. statikus perspektívával, amely a végrehajtandó folyamattevékenységeket mutatja;

3. gyakorlati perspektívával, amely jól hasznosítható gyakorlatokat javasol a folyamat alatt.

Az időbeliség alapján az RUP a rendszerfejlesztést négy nagyobb egységre, négy diszkrét fázisra bontja:

2.2.11 RUP rendszerfejlesztési fázisok

- Kezdeti fázis

A kezdeti fázis (inception) során a rendszerrel szembeni elvárásokat és a rendszer funkcionalitását olyan mélységig kidolgozzák, hogy az alapján a fejlesztési folyamat részletesen tervezhető lesz.

- Kidolgozási fázis

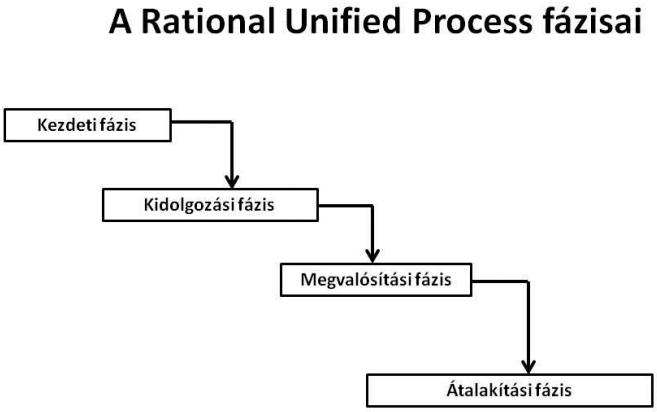
A kidolgozási fázisban (elaboration) részletekbe menően meghatározzák, hogy a felhasználók hogyan fogják használni a rendszert, és a rendszer hogyan fog felépülni, valamint elkészül az ún. stabil alaparchitektúra terve (architecture baseline). Az alaparchitektúra segítségével a teljes fejlesztés folyamata ütemezhető és a költségei is tisztázhatók.

- Megvalósítási fázis

A megvalósítás fázis (construction) során a teljes rendszer kifejlesztik. A folyamat a rendszertervre, a programozásra és a tesztelésre fókuszál. A rendszer különböző részei párhuzamosan fejleszthetők, majd ez alatt a fázis alatt integrálhatók. A fázis végére elkészül a működőképes szoftverrendszer, és a hozzá kapcsolódó dokumentáció.

- Átalakítási fázis

Az átadási fázis (transition) során a rendszer működésének tesztelése zajlik, melynek tapasztalatait részletesen dokumentálják. A tesztelés eredményeképpen döntenek a rendszer átalakításáról vagy késszé nyilvánításáról.



14. ábra: RUP fejlesztési fázisok

2.2.12 eXtreme Programming

Az 1990-as evek elején Kent Beck es Ward Cunningham a Chrysler autógyárnak készítette el azt a komplex bérügyi rendszert, amely át tudta venni az eddig használt, több szoftverből álló rendszer minden feladatát. Az 1997-ben bevezetett rendszert röviden C3-nak nevezték (Chrysler Comprehensive Compensation System), és az ennél a szoftvernél alkalmazott metódust nevezik azóta eXtreme Programming fejlesztésnek, vagy röviden XP-nek.

15. ábra: Ward Cunningham

Az eXtreme Programming egy olyan szoftverfejlesztési megközelítés, amely proaktív szerepet szán a megrendelőnek, szinte egyenrangú félként bevonva a fejlesztési folyamatba.

Az inkrementális fejlesztési alapokon nyugvó eXtreme Programming egyik sajátossága, hogy a fejlesztés szinte valamennyi fázisát egy folyamatba olvasztja össze, illetve a szoftver fejlesztése és tesztelése időben nagyon közel kerül egymáshoz (akár mindössze néhány órára is csökkenhet). A rendszer sajátossága az is, hogy a fejlesztők párban dolgoznak, és a szoftver fejlesztői saját programkódjaikat ellenőrzik.

16. ábra: Kent Beck

Ennek a metódusnak az előnyei között szokták említeni a magasabb szintű produktivitást, a jobb kommunikációt a fejlesztők, illetve a fejlesztők és a megrendelő között, továbbá a magasabb színvonalú végeredményt.

Hátrányai között szokták említeni, hogy nem alkalmazható minden esetben, kiválóan képzett programozók kellenek a megvalósításához, és az intenzív kommunikáció nagyon sok időt és energiát von el a fejlesztési folyamattól, illetve a sajátos fejlesztési metódus miatt a fejlesztés dokumentációja rendszerint nem kielégítő.

2.2.13 Agilis szoftverfejlesztés

Az Agile (agilis) szoftverfejlesztési módszer egy sikertelen tanácskozás eredménye, amelyet 2001 februárjában a vezető szoftverfejlesztő cégek szakemberei tartottak az amerikai Utah államban. A tanácskozás során teljesen új módszertant nem tudtak kidolgozni, de létrehozták az Agile Manifesto kiált-

ványt, amely összefoglalása azoknak az elgondolásoknak, amelyeket a résztvevők fontosnak tartottak leszögezni:

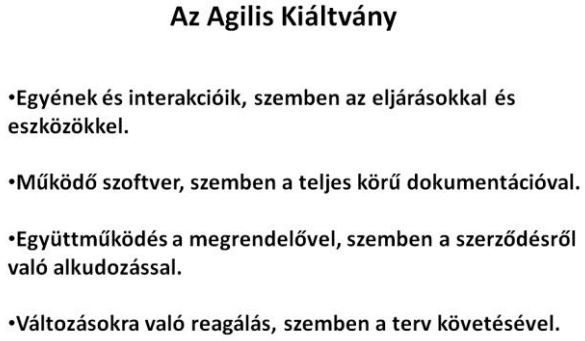
Az Agilis Kiáltvány

A szoftverfejlesztés jobb módjait fedezzük fel azáltal, hogy csináljuk, és segítünk másoknak is csinálni. Ennek során az alábbi hangsúly-eltolódásokat találtuk:

- Egyének és interakcióik, szemben az eljárásokkal és eszközökkel.
- Működő szoftver, szemben a teljes körű dokumentációval.
- Együttműködés a megrendelővel, szemben a szerződésről való alkudozással.
- Változásokra való reagálás, szemben a terv követésével.

Ez azt jelenti, hogy a jobb oldalon szereplő értékek is fontosak, de a bal oldalon lévőket fontosabbnak tartjuk.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas



17. ábra: Az Agilis kiáltvány

2.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

2.3.1 Összefoglalás

A második fejezet célja az volt, hogy a hallgatók rendelkezzenek információval a rendszertervezés alapfogalmait illetően, ismerjék meg a rendszertervezés rövid történetét, legyenek tisztában az információs rendszerek tervezésének alapvető tudnivalóival és a tervezés ciklusaival. Ismerjék meg a legfontosabb rendszertervezési elveket, rendelkezzenek információval az alábbi metódusokról:

- Vízesés modell
- Evolúciós modell
- Komponens alapú és újrafelhasználható rendszerfejlesztés
- Spirális fejlesztés
- Gyors Alkalmazásfejlesztés (RAD)
- RUP (Rational Unified Process)
- eXtreme Programming
- Agilis szoftverfejlesztés

2.3.2 Önellenőrző kérdések

1. Ismertesse a rendszertervezés történetét röviden!
2. Mit tud az információs rendszerek tervezéséről és a tervezés ciklusairól?
3. Hasonlítsa össze az alábbi modelleket:
4. Vízesés modell
5. Evolúciós modell
6. Komponens alapú és újrafelhasználható rendszerfejlesztés
7. Spirális fejlesztés
8. Gyors Alkalmazásfejlesztés (RAD)
9. RUP (Rational Unified Process)
10. eXtreme Programming
11. Agilis szoftverfejlesztés

3. LECKE: SZOFTVERFORRÁSOK

3.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A harmadik lecke célja, hogy a hallgatók megismerkedjenek a vállalati információs rendszerek szoftvereinek forrásaival. Ennek során szót ejtünk az outsourcing jellemzőiről, beszélünk arról, hogy milyen előnyei és hátrányai lehetnek annak, ha a szoftverfejlesztést külső IT-cégek segítségével végezzük. A fejezet második részében megvizsgáljuk az előre elkészített szoftvercsomagok szerepét, és vállalatirányítási információs rendszerek számítási felhőbe történő integrálásának lehetőségeit is. Szót ejtünk a nyílt forráskódú szoftverek alkalmazásáról, és a házon belüli szoftverfejlesztés jellemzőiről is. A fejezet végén megvizsgáljuk a szoftvervásárlás legfontosabb szempontjait.

3.2 TANANYAG

- Szoftverforrások
- Outsourcing
- Szoftverfejlesztés IT-cégek segítségével
- Előre elkészített szoftvercsomagok
- Vállalatirányítási információs rendszerek
- Vállalatirányítási információs rendszerek a számítási felhőben
- Nyílt forráskódú szoftverek alkalmazása
- Házon belüli szoftverfejlesztés
- A szoftvervásárlás szempontjai
- Költségek
- Funkcionalitás
- Rugalmasság
- Dokumentáció
- A szoftverforgalmazó megítélése

3.2.1 Szoftverforrások

A számítógép alapú rendszerek tervezése az ötvenes évekig nyúlik vissza. Ahogyan az előző leckében már említettük, néhány tíz évvel ezelőtt a rendszertervezés minden szervezetnél egy kifinomult, már-már művészi tevékenység

volt, ami hosszú időt és egyedi megoldások kifejlesztését igényelte, és ez minden esetben egyet jelentett azzal, hogy a cégeknek saját maguknak kellett információs rendszereiket kifejleszteni, hiszen ekkoriban még nem létezett a szoftveripar. Az információs rendszerek iránti igények exponenciális növekedésével és a szoftveripar megjelenésével a rendszertervezési folyamat egyre inkább tudományosan és gyakorlati tapasztalatok által megalapozott, jól körülhatárolható lépések és eljárások sorozatává szelídült, ami lehetővé tette, hogy információs rendszereink beszerzésénél többféle forrásból is válasszunk.

3.2.2 Outsourcing

Amikor egy informatikai cég egy másik cég számára hardveres és/vagy szoftveres megoldásokat nyújt az információs rendszer működtetésére, ezt a gyakorlatot nevezzük outsourcingnak. Magyarul a fogalmat kiszervezésnek fordították le, de ez nem igazán vált széles körben elfogadottá, a legtöbb esetben az angol verziót használjuk.

Az outsourcing-megoldások többfélék lehetnek, a legmagasabb fokú kiszervezést az jelenti, amikor egy cég mind a hardver-, mind a szoftverelemeket rendelkezésre bocsátja a feladat elvégzéséghez, és a szolgáltatást igénybe vevő vállalatnak csak a bemeneti adatokat kell biztosítania, ezekből készen kapja meg a kért kimeneti adatokat (kimutatások, jelentések stb.).

A gyakorlati életben erre megoldásra lehet említeni példaként a bérügyek kiszervezését, ami azt jelenti, hogy a szolgáltatást igénybe vevő vállalat szerződést köt az outsourcing céggel, hogy az térítés ellenében a munkáltató és a munkavállalók adatai alapján minden hónapban előállítja a bérrel kapcsolatos összes információt (jelentések, kimutatások) és előkészíti a munkabérek és a járulékok utalását.

A kiszervezés egy másik szintje valósul meg abban az esetben, amikor a kiszervezést igénybe vevő vállalat rendelkezik a megfelelő hardver infrastruktúrával, és csak a szoftvert vásárolja meg (vagy bérli) a szolgáltatást nyújtó informatikai cégtől. Ebben az esetben rendszerint a szerződésbe foglalják, hogy a szoftver működtetésén túl a szolgáltatást nyújtó cég milyen egyéb kötelezettségeket vállal (humán erőforrás betanítása, adatmentés, szoftverfrissítések biztosítása stb.).

18. ábra: Szoftverforrások

3.2.3 Szoftverfejlesztés IT-cégek segítségével

A gyakorlati életben gyakran előfordul, hogy az outsourcing valamilyen okból nem jelent megoldást, és jelenleg nincs a szoftverpiacon olyan információs rendszerszoftver, amely a vállalat miden igényét kielégíti. Ebben az esetben érdemes egy információs rendszerszoftver-fejlesztőcég szolgáltatásait igénybe venni. Ez a megoldás abban különbözik az outsourcingtól, hogy a szoftvert a vállalat igényeihez igazítva hozzák létre, és nem kell kompromisszumokat kötni az outsourcing szolgáltatást nyújtó cég által biztosított lehetőségek és az elvárásaink között. A fejlesztőcég az előző fejezetben ismertetett módszerek valamelyikét alkalmazva létrehozza az információsrendszer-szoftvert, telepíti azt a vállalat gépeire (illetve beszerzi a rendelkezésre nem álló eszközöket), és betanítja annak használatát és karbantartja a rendszert.

Információs rendszerek tervezésével és telepítésével számos informatikai cég foglalkozik, talán meglepő, de a legnagyobb bevételt ebben az iparágban 2007-ben az IBM érte el (74 millió dollár), ami alátámasztja azt a feltételezést, hogy a „Nagy Kék” egyre inkább az informatikai szolgáltatások biztosításában látja a jövőt, és nem a hardvergyártásban.

3.2.4 Előre elkészített szoftvercsomagok

A hatvanas évek közepétől kezdődően a szoftvercégek hihetetlen növekedésének lehetünk tanúi. A világ legnagyobb informatikai cégei közül számos szinte kizárólag szoftverek értékesítésével foglalkozik. Jó példa erre a Microsoft, amely 2007-ben a második legnagyobb informatikai cég volt, és a bevételeinek

98%-a szoftverértékesítésből származott. Ennek a jövedelemnek a legnagyobb része a Windows operációs rendszerből és a hozzá szorosan köthető Office irodai programcsomagból ered.

A Microsofthoz hasonló szoftvercégek (SAP, Oracle stb.) gyakran kínálnak instant szoftvercsomagokat a legkülönfélébb információs rendszerfeladatok megoldásra. A Microsoft esetében példaként említhetjük pl. a Microsoft Project programcsomagot, amelyet elsősorban a projektmenedzsment területén alkalmaznak, de számtalan üzleti informatikai feladatra találunk megoldást a többi gyártó termékei között is. Ezek a csomagok széleskörűen alkalmazhatóak különböző platformok esetén is, hiszen a személyi számítógépektől kezdve egészen a mainframe-ekig terjed a termékpaletta. Ezek között találunk olyat, amelyet csupán egy néhány fős mikrovállalkozás használhat hatékonyan, és olyat is, amely egy több tízezer fős nagyvállalat igényeit is képes kielégíteni.

Az előre elkészített szoftvercsomagok egyetlen hátránya, hogy nem testre szabhatóak a vállalatok speciális igényeit figyelembe véve. Gyakorlati tapasztalatok szerint ezek a szoftverek a vállalat igényeihez maximum 70%-ban igazodnak, azaz a legjobb esetben is a feladatok legalább 30%-ára más megoldást kell keresni.

3.2.5 Vállalatirányítási információs rendszerek

A vállalatok jelentős része vásárol dedikált vállaltirányítási szoftvereket, amelyek a vállalt teljes tevékenységi körének információs menedzsment feladatait ellátják. A szakirodalom nem egységes abban a tekintetben, hogy mennyire helytálló az elnevezés, hiszen van olyan megközelítés, amely az ERP (Enterprise Resource Planning) rendszereket azonosítja a fenti fogalommal, míg más megközelítés szerint az üzleti alkalmazások (enterprise application software (EAS)) tartozik szorosan ide, és az ERP ennek csak az egyik összetevője.

A vállalatirányítási információs rendszerek modulokból épülnek fel, minden hagyományos üzleti tevékenységhez tartozik egy-egy szoftvermodul, így az alaprendszer az igényeknek megfelelően tetszőlegesen bővíthető. A vállalatirányítási információs rendszerek használatának az előnye, hogy az adatok konzisztens módon kerülnek feldolgozásra, tárolásra és megjelenítésre, ami megkönnyíti a modulok közötti adatkommunikációt, és az archiválást.

A vállalatirányítási információs rendszerek alkalmazásának hátrányai között meg kell említenünk, hogy a rendszer használata komplex ismereteket kíván, ezért annak elsajátítása csak a forgalmazó által szervezett konzultációk során lehetséges, ami bizonyos esetekben költséges lehet. Továbbá a vállalatnak meg kell bíznia a forgalmazóban, mert az a rendszer telepítése során bizalmas adatokba nyerhet betekintést.

3.2.6 Vállalatirányítási információs rendszerek a számítási felhőben

A vállalatok egyre növekvő része dönt úgy, hogy nem vásárolja meg a vállalatirányítási információs rendszerek működtetéséhez szükséges hardver- és szoftverelemeket, hanem az interneten keresztül veszi igénybe valamelyik informatikai cég ilyen irányú szolgáltatását. A szolgáltatásért rendszerint havidíjat, más módon kalkulált díjat (pl. alkalmak száma, adatmennyiség stb.) fizet a felhasználó.

A számítási felhő üzleti potenciálja évről évre nő, egyes jóslatok szerint 2013-ra a vállalati informatikai rendszerek számítási teljesítményének 12%-át a felhő fogja szolgáltatni. A számítási felhőhöz köthető üzleti forgalmat 160 milliárd dollárra becsülik, amelyből 95 milliárd üzleti informatika, 65 milliárd dollár pedig reklámbevételként fog realizálódni.

A vállalatirányítási információs rendszerek számítási felhőben való működtetésével kapcsolatosan több előnyt is említhetünk: egyrészt a vállalat informatikai humán erőforrása mentesül az információs rendszerrel kapcsolatos fejlesztési és üzemeltetési feladatok alól, másrészt a felhő szolgáltatásainak igénybe vétele jelentős költségmegtakarítást eredményez, ha a hardver- és szoftverelemek beszerzésének költségeivel összehasonlítjuk.

Hátrányként rendszerint egyetlen dolgot említhetünk: a vállalat adatai kikerülnek egy „külső” cég adattárolóira, amely biztonsági kockázatot jelenthet.

3.2.7 Nyílt forráskódú szoftverek alkalmazása

A nyílt forráskódú szoftverek jelentősen különböznek az eddig említett megoldásoktól: nemcsak a szoftver, de annak forráskódja is térítés nélkül használható, illetve átdolgozható. További különbség, hogy ezeket a szoftvereket rendszerint nem fizetett alkalmazottak, hanem a téma iránt érdeklődő szakemberek készítik. Az informatika szinte minden területén találunk nyílt forráskódú szoftvereket, legyen szó operációs rendszerről, irodai programcsomagról, adatbázis-kezelőről stb. A legismertebb nevek között említhetjük a Linux, MySQL, Firefox, OpenOffice stb. alkalmazásokat. A nyílt forráskódú szoftverek készítői esetenként rendkívül nagy alkotóközösségbe tartoznak, amelyeket többféle módon szerveznek. Az egyik legismertebb ilyen szervezet a SourceForge.net közösség, amelynek 2009-ben kétmillió regisztrált felhasználója volt, és több mint 180 000 projekt tartozott a közösséghez.

3.2.8 Házon belüli szoftverfejlesztés

Az előzőekben számos lehetőséget megvizsgáltunk a külső informatikai cégek illetve szolgáltatók által nyújtott megoldásokkal kapcsolatban. Ahogyan a történeti részben már szó volt róla, az ötvenes években szinte kizárólag a saját fejlesztésű szoftverek jelentették a megoldást a rendszertervezési problémákra, de a szoftveripar fejlődésével egyre csökkent az önálló fejlesztések szerepe, míg mára ez lett a legkisebb szegmens ezen a területen.

A házon belüli szoftverfejlesztés mellett szólhat, hogy az összes többi megoldással összehasonlítva (megfelelően képzett humán erőforrás esetén) ez a rendszer illeszkedik legjobban a vállalat igényeihez.

A másik gyakran hangoztatott érv, hogy a fejlesztési költségek rendszerint alacsonyabbak, mint a külső fejlesztőcégek vagy szolgáltatók igénybevétele esetén. A mélyebb elemzések ezzel szemben azt mutatták ki, hogy az alacsony fejlesztési költségekkel szembe kell állítanunk a magasabb karbantartási költségeket és a hosszabb karbantartási idő miatti produktivitáscsökkenést, amelynek tükrében már nem biztos, hogy ez a legköltségkímélőbb megoldás.

Éppen ezért a gyakorlatban ritkán találkozunk teljes egészében belső fejlesztésű információs rendszerekkel, sokkal gyakoribbak a hibrid megoldások, amikor az önállóan fejlesztett és a külső informatikai cég vagy szolgáltató által nyújtott rendszerelemek egyaránt megtalálhatóak az információs rendszerben. Ez utóbbi megoldás egyetlen hátránya a rendszerek közötti zökkenőmentes kommunikáció biztosításának a nehézsége.

3.2.9 A szoftvervásárlás szempontjai

Ha az előzőekben vázolt lehetőségek közül végül a szoftvervásárlás mellett döntünk, akkor érdemes végiggondolni, hogy mely szempontok játszhatnak szerepet a végső döntésben.

3.2.10 Költségek

A költségek magukban foglalják a szoftver megvásárlásán vagy a szolgáltatás igénybevételi díján felül a telepítés, a szoftverfrissítések és a felhasználók oktatásának díját is. Amikor a költségeket vizsgáljuk, nem szabad elfeledkeznünk az előzőekben már említett várható karbantartási idő és a produktivitás közötti összefüggésekről sem.

3.2.11 Funkcionalitás

Ennél a szempontnál a vizsgálat középpontjában a szoftver használhatósága áll, vagyis milyen mértékben képes az kielégíteni a vállalat információs szükségleteit. Ahogyan a korábbiakban már beszéltünk róla, optimális esetben is legfeljebb 70% körüli az átfedés a vállalati igények és a készen megvásárolható információsrendszer-szoftverek között. A döntést segítheti, ha megvizsgáljuk, hogy az igények prioritási sorrendjében a legfontosabb, vagy a kevésbé fontos elemeket építették be a szoftverbe?

3.2.12 Rugalmasság

A szoftver flexibilitása szorosan kapcsolódik az előző szemponthoz: testre szabható-e, illetve milyen mértékben az a megvásárolt szoftver? A flexibilitás megnyilvánul a funkcionális elemek megváltoztathatóságán túl az adatkezelésben, az űrlapok struktúrájában, a grafikai megjelenésben és még a számos helyen. A flexibilitás mértéke erősen kihat a munkafolyamatok szervezésére, azaz minél alacsonyabb szintű a szoftver flexibilitása, annál inkább szükséges a munkafolyamatok átszervezése.

3.2.13 Dokumentáció

A dokumentáció magába foglalja a szoftver technikai dokumentációját (futtatási környezet, hardverfeltételek stb.) és a felhasználói útmutatót. A dokumentáció megítélésénél szempont lehet a részletesség, a könnyen érthető nyelvezet, a naprakészség (a szoftver frissítéséből adódó verzióváltást mennyire követi a dokumentált változat), illetve hány példányban áll rendelkezésre az oktatóanyag (nagy létszámú vállalatoknál a papíralapú oktatóanyag előállítása rendkívül költséges lehet).

A dokumentációnak tartalmaznia kell a telepítési instrukciókat is, amelyből világossá válik, hogy a mennyire bonyolult folyamat a szoftver telepítése, szükség van-e külső segítségre?

3.2.14 A szoftverforgalmazó megítélése

Bár utoljára hagytuk, ez az egyik legfontosabb szempont. A rendszerszoftvereket rendszerint több éven keresztül szeretnék használni a vállalatok, ezért csak olyan forgalmazó szoftvereit érdemes megvenni, amely már régóta jelen van a szoftverpiacon. Akármennyire is kecsegetető lehet egy kevéssé ismert új informatikai cég ajánlata, nincs garancia arra, hogy a cég nem tűnik hónapokon belül az informatika üzleti szférájából.

Legalább ugyanilyen fontos, hogy a nagynevű informatikai cégek terméktámogatási rendszere sokkal jobban kidolgozott, és a cég jó híre érdekében feltehetőleg mindent megtesznek a szoftverhasználati szerződés keretein belül a vállalat igényeinek kielégítésére (a szoftver rendszeres hibajavítása, fejlesztése, a szoftverfrissítések telepítése stb.).

Egyetlen hátránya lehet annak, ha a fenti metódust követve egy nagynevű informatikai cég szoftverét választjuk: a sok ügyfél és a hierarchikus döntéshozatal miatt nagyon kicsi esélyünk van arra, hogy a vállalat igényeit jobban kielégítő szoftvermódosításokat érjünk el a forgalmazónál.

3.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

3.3.1 Összefoglalás

A harmadik lecke célja az volt, hogy a hallgatók megismerkedjenek a vállalati információs rendszerek szoftvereinek forrásaival. Ennek során szót ejtettünk az outsourcing jellemzőiről, beszéltünk arról, hogy milyen előnyei és hátrányai lehetnek annak, ha a szoftverfejlesztést külső IT cégek segítségéve végezzük. A fejezet második részében megvizsgáltuk az előre elkészített szoftvercsomagok szerepét, és vállalatirányítási információs rendszerek számítási felhőbe történő integrálásának lehetőségeit is. Szót ejtettünk a nyílt forráskódú szoftverek alkalmazásáról, és a házon belüli szoftverfejlesztés jellemzőiről is. A fejezet végén megvizsgáltuk a szoftvervásárlás legfontosabb szempontjait.

3.3.2 Önellenőrző kérdések

1. Milyen szoftverforrásokat ismer?
2. Ismertesse az outsourcing előnyeit és hátrányait!
3. Ismertesse az előre elkészített szoftvercsomagok előnyeit és hátrányait!
4. Ismertesse a nyílt forráskódú szoftverek alkalmazásának előnyeit és hátrányait!
5. Ismertesse a házon belüli szoftverfejlesztés előnyeit és hátrányait!
6. Ismertesse a szoftvervásárlás szempontjait!

4. LECKE: A RENDSZERFEJLESZTÉSI PROJEKT MENEDZSELÉSE

4.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A negyedik fejezet célja, hogy a projektmenedzsment legfontosabb ismereteivel felruházza a hallgatókat. A lecke kapcsán szó lesz többek között a projektszervezet felépítéséről, a projekt életciklusról, a projekt szakaszairól. A lecke második részében megtárgyaljuk a projektfolyamat sajátosságai, ejtünk szót a projekt-előkészítés szakaszairól, a projekt céljának megfogalmazásáról, a megvalósíthatóság vizsgálatáról és a megvalósíthatósági tanulmány elkészítéséről.

A lecke harmadik részében szó lesz a célrendszer és sikerkritériumok meghatározásáról, az erőforrásbecslésről és a projekttervezés szakaszairól.

4.2 TANANYAG

- Projektmenedzsment
- A projektet megvalósító szervezet struktúrája
- A projekt megvalósításának szakaszai
- A projekt megvalósításának jellemzői
- A projekt megvalósításának első szakasza
- A projekt céljának megfogalmazása, a megvalósíthatóság vizsgálata, a megvalósíthatósági tanulmány elkészítése
- A projekt céljainak hierarchiája
- A projekt hatókörének meghatározása
- Időterv és erőforrásbecslés
- A projektcsoport felállítása
- A feladatok meghatározása
- A feladatmátrix megtervezése
- Kockázatok kezelésének tervezése
- A kommunikáció tervezése
- A logisztikai feladatok tervezése
- Az előkészítési szakasz lezárása
- A végrehajtási szakasz

- A projekt zárása
- A projekt átadását követő tevékenységek meghatározása
- A projektzáró értekezlet
- A projektzáró jelentés

4.2.1 Információs rendszerek létrehozása projektek keretében

Az információs rendszerek tervezésének menete rendszerint projektek keretében zajlik. A projektek rendszerint egy adott cél elérés érdekében jönnek létre. A projekt végrehajtása adott időkereten és belül és meghatározott anyagi erőforrások felhasználása mentén történik meg.

A projekt szakaszai: előkészítés, tervezés, végrehajtás zárás és értékelés.

A projektekre általánosan jellemző:

- a feladat végrehajtásáról a menedzsment dönt;
- a feladat adott cél elérésére szolgál;
- a feladat végrehajtása időben korlátozott;
- a rendelkezésre álló erőforrások korlátozottak;
- a feladat végrehajtására projektszervezetet hoznak létre

4.2.2 Projektmenedzsment

A projektmenedzsment az erőforrások szervezésével és irányításával foglalkozó diszciplína, amelynek célja, hogy a projekt céljait a rendelkezésre álló erőforrások segítségével a rendelkezésre álló időn belül elérjük. Információs rendszerek tervezésénél ez az információs rendszer létrehozását jelenti.

19. ábra: A projektszervezet felépítése

4.2.3 A projektet megvalósító szervezet struktúrája

Projektmunka során az egyik legfontosabb feladat a projektmenedzsment hatáskörének meghatározása. A projektet megvalósító szervezetben meg kell határozni, hogy ki milyen hatáskörrel, döntési jogosultsággal rendelkezik, definiálni kell az alá- és fölérendeltségi viszonyokat, a beszámolási kötelezettségeket, a felelősségi köröket stb.

- A megbízó

A megbízó rendszerint a projektmenedzsment tagja, a projekt megvalósításának kezdeményezője. Biztosítja a projekt számára a célok megvalósításához szükséges erőforrásokat, működési feltételeket.

- Projektmenedzsment

A projektmenedzsment a projekt legfőbb döntéshozó, ellenőrző szerve. Hatásköre kiterjed minden, a projekt sikeres végrehajtását befolyásoló kérdésre, tényezőre pl. az erőforrások optimális felhasználása, határidők betartása stb.

- Projektmenedzser

A projektmenedzser a projekt irányításának napi feladatait látja el, rendszerint ő felel a projekt megfelelő színvonalú végrehajtásáért.

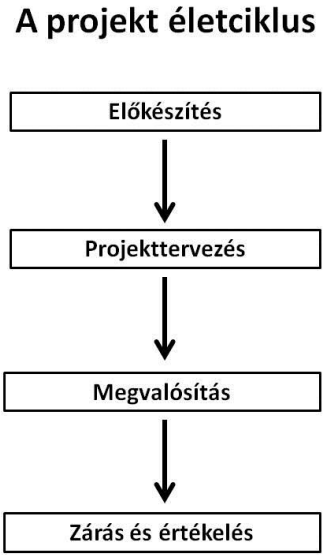
- A projektcsoport

A projektcsoport a projekt céljainak elérése érdekében, a projektmenedzsment által jóváhagyott és irányított csoport.

4.2.4 A projekt megvalósításának szakaszai

A projekt megvalósítása során az első lépés a projekt-előkészítés és a -tervezés, amelynek eredményeként a projekt célja (és a cél eléréséhez szükséges idő és erőforrás) egyre pontosabban látható.

A projekt előkészítése során meghatározásra kerülnek a projekt megvalósításának peremfeltételei, amelyek az imént említett projektcélon és erőforrásokon kívül tartalmazzák a projekt megvalósításában résztvevők névsorát, a projekt hatásmechanizmusát és a cél megvalósításának minőségi feltételeit.



20. ábra: A projektmegvalósítás szakaszai

Az előkészítési szakasz során hivatalos dokumentáció rendszerint nem készül, ekkor még csak nagy vonalakban kerül meghatározásra a projektötlet. A projektötlet megszületése után készül el az első dokumentum, az ún. projektindító dokumentum, amelynek alapján döntés születik a részletes projekttervezés megkezdéséről.

A projekttervezés szakaszában kerül részletesen megtervezésre a projekt megvalósításának módja. Ekkor kerül sor a projekt megvalósításáért felelős szervezet felállítására; a működési szabályzat kialakítására, az elvégzendő feladatok hierarchikus rendszerének meghatározására és a logisztikai tervezésre (időterv, energia- és információ és nyersanyagáramlás megtervezése).

A projektterv és a projektszervezet működési szabályzatának elfogadását követően döntés születhet a projekt megvalósításának elindításáról.

A projektmegvalósítás szakasza a projekt konkrét céljának realizálását jelenti. A projekt tervezése és a megvalósítás rendszerint kisebb-nagyobb mértékben eltér egymástól, hiszen a tervezés papíron történik a megvalósítás pedig a való világ folyton változó környezetében. A lehetőségekhez képest a minél pontosabb megvalósítást a projekt előrehaladásának ellenőrzésére szolgáló mérföldkövek és a monitoring rendszer segíti.

4.2.5 A projekt megvalósításának jellemzői

A projekt megvalósítását rendszerként megközelítve a teljes projektfolyamatot részrendszerekre oszthatjuk fel, ahol a részrendszerek hierarchikus felépítésűek és az egymással kapcsolatban álló rendszerkomponensek kimenetei bemenetként szolgálnak a következő rendszerkomponens számára. A részrendszerek sorrendje rendszerint kötött, akkor léphetünk a következő részrendszerre, ha az előző rendszerkomponensben meghatározott feladatok elvégzésre kerültek. A kötött sorrend ellenére a projekt megvalósításának folyamata nem lineáris, rendszeresen felül kell vizsgálni a részrendszereket, hogy minden követelmény teljesítésre került-e, és esetleg szükséges-e változtatni a követelményeken. Rendszerint ezt a szemlélet követjük a teljes projektfolyamat egészére is.

4.2.6 A projekt megvalósításának első szakasza

A projekt megvalósításának első szakasza az előkészítési szakasz, amely egy jól körvonalazható igény megfogalmazásával kezdődik. Ezt követi a megoldásanalízis, amely a megvalósíthatósággal és a megvalósítás alternatíváival foglalkozik. Ennek a szakasznak a kimenete a többféle megoldást tartalmazó megvalósíthatósági dokumentum. A későbbiekben az alternatívákból a projektmenedzsment kiválasztja az optimális változatot, meghatározza projekt sikeres végrehajtását garantáló mérföldköveket és közelítő pontossággal megtervezi a szükséges erőforrásokat, majd elkészíti a projekt megvalósításához szükséges szervezet tervét. A szakasz végeredményeként megszületik projektalapító dokumentum és a megvalósíthatósági tanulmány alapján döntés születik a részletes tervezés megkezdéséről.

4.2.7 A projekt céljának megfogalmazása, a megvalósíthatóság vizsgálata, a megvalósíthatósági tanulmány elkészítése

Az előző szakaszban felmerülő igény kielégítésére a projektmenedzsment megfogalmazza azt a célt, amelynek az elérésére a projekt, mint eszköz felhasználható. A cél elérése érdekében szükség van egy projektmenedzser kijelölésére, aki a projekt tevőleges megvalósításában rendszerint nem vesz részt, azonban koordinálja a megvalósítás folyamatát.

A projekt megvalósíthatóságának a vizsgálata során a projektmenedzsment eldönti, hogy az elérendő cél milyen alternatívák mentén valósíthatóak meg és a megvalósítására elegendőek-e a rendelkezésre álló erőforrások. Egy adott cél elérésére több reálisan végrehajtható alternatíva is kidolgozható, amelyeknek

részletes kidolgozásának eredménye a megvalósíthatósági tanulmányok létrejötte.

4.2.8 A projekt céljainak hierarchiája

Ahogyan az előzőekben már láttuk, a projektet egy jól megfogalmazható igény kelti életre és az igény kielégítése a projekt végső célja. Azonban a végső cél eléréséhez számtalan részcélon keresztül vezet az út. ezek a részcélok az információs rendszerekben hierarchikus rendben épülnek egymásra, azaz az egyik részcél megvalósulása feltétele a másik részcél megvalósulásának. Az információs rendszerek megvalósítását célul kitűző projekteknek nagyon fontos annak a meghatározása is, hogy a kész információs rendszernek milyen tulajdonságokkal nem kell majd rendelkeznie, mert ez vezet el ahhoz, hogy a tervezés során felmerült célok közül néhányat kizárjunk a megvalósítási folyamatból.

4.2.9 A projekt hatókörének meghatározása

A projekt hatókörének meghatározása a célrendszer alapján történik, azaz leírja, hogy mi tartozik a projekt hatáskörébe és mi nem. A projekt hatóköre a projekt által meghatározott célok eléréséhez elvégzendő feladatok és eredmények pontos meghatározását tartalmazza.

4.2.10 Időterv és erőforrásbecslés

Az időterv és erőforrásbecslés során meghatározásra kerül a projekt kivitelezéséhez szükséges idő és erőforrások meghatározása. Mivel ezek a tervek becslésen alapulnak, ezért a projekt megvalósítása során gyakran szembesülünk azzal a problémával, hogy a terveinket módosítani kell (ez szinte minden esetben azt jelenti, hogy pozitív irányban kell eltérnünk a becsült adatoktól, azaz a projekt megvalósítása több időt és nagyobb összeget fog felemészteni, mint ahogyan azt terveztük). Az erőforrások alatt itt egyaránt gondolunk anyagi és humán erőforrásra. A humán erőforrás becslése azért is nagyon fontos, mert az alapján állíthatjuk majd fel a tervezési szakaszban a projekt kivitelezéséhez szükséges csoportot.

4.2.11 A projektcsoport felállítása

A projekt célrendszerének meghatározása után azonosítjuk a célok eléréséhez szükséges tevékenységeket, majd a tevékenységeket szerepkörökhöz rendeljük hozzá. A szerepkörök alapján meghatározzuk a szükséges kompetenciákat és a kompetenciákhoz hozzárendeljük a megfelelő

személyeket. Ha valamelyik kompetenciához nem áll házon belül rendelkezésre megfelelő humán erőforrás, akkor gondoskodnunk kell annak megszerzéséről (fejvadász cégek). A projektcsoportban a személyek kiválasztásán túl meg kell határozni a szerepkörök hierarchiáját, az elvárásokat, a döntési és felelősségi jogköröket, nem is beszélve a az adott szerepkörhöz tartozó anyagi ellenszolgáltatásokról.

4.2.12 A feladatok meghatározása

A projekt célrendszerének meghatározása kétségtelenül elengedhetetlen a projekt globális tervezése szempontjából, azonban ezek a célok rendszerint túlságosan nagy léptékűek ahhoz, hogy projekt megvalósításának tervezése során ezekkel dolgozzunk. Éppen ezért azonosítanunk kell a célok eléréséhez szükséges feladatokat, amelyeket a célok eléréséhez szükséges lépések részletezésével érhetünk el. Természetesen a részletezettség mértékét a célok komplexitása és az ésszerűség határozza meg. Az is nyilvánvaló, hogy azokat a részcélokat, amelyek más részcélok megvalósulására épülnek, sokkal nehezebb megfelelő részletezettséggel feladatokra bontani, mint azokat, amelyeket a projekt kezdetén kell megvalósítani. Ennek következményeként a feladatmátrixot időről-időre felül kell vizsgálnunk.

4.2.13 A feladatmátrix megtervezése

A feladatmátrix megtervezése során az előzőekben meghatározott feladatok sorrendiségét a logikai és technológiai összefüggések alapján fel kell állítani, majd a feladatokhoz hozzá kell rendelni az elvégzéshez szüksége időt és a megvalósításhoz szükséges anyagi és emberi erőforrásokat. Meg kell vizsgálni, hogy mely folyamatok követelik meg a szigorú sorrendiséget és mely feladatok végezhetőek párhuzamosan. A tevékenység eredményeképpen rendelkezésre áll egy időterv, amely tartalmazza az összes elvégzendő feladatot és a szükséges erőforrásokat is.

4.2.14 Kockázatok kezelésének tervezése

A projekttervezés szempontjából kockázatnak minősül minden olyan tényező, amely a projekt megvalósulását veszélyeztetheti. A kockázati tényezők elemzése viszonylag tág határok között biztosítja a projekt megvalósítását. A kockázati tényezők között meg kell különböztetnünk külső és belső tényezőket. Az előbbi a projekt megvalósításának környezetét befolyásolja (politikai rendszerek, szélsőséges időjárási viszonyok stb.) A belső kockázati tényezők közül kiemelt szerepe van a pénzügyi (pl. a becsült anyagi erőforrás nem elegendő a

projekt megvalósításához), a technológiai (a projekt megvalósításához tervezett technológia alkalmatlan a kitűzött célok elérésére) és a humán erőforráshoz (szakértelem hiánya, emberi mulasztásból adódó károkozás stb.) köthető kockázatoknak.

A kockázatkezelés tervezése során megpróbálják minél pontosabban azonosítani a lehetséges kockázati tényezőket és olyan indikátorokat létrehozni, amelyek jelezhetik valamelyik kockázati tényező térnyerését.

4.2.15 A kommunikáció tervezése

A projekt eredményes megvalósításához elengedhetetlen az információ zavarmentes áramlásának biztosítása. Ez azt jelenti, hogy a projekt humán erőforrására vonatkoztatva elemezni kell, hogy kinek, mikor, milyen információra van szüksége és ennek megfelelően gondoskodni kell a szükséges információ megfelelő helyre is időben történő eljuttatásáról. Az információ tartalmilag magába foglalja az adott részfeladat elvégzéséhez szükséges ismereteken túl a többi részfeladat készültségi állapotát és minden olyan változást, amely kihatással lehet a projekt sikeres kivitelezésére.

4.2.16 A logisztikai feladatok tervezése

A logisztikai feladatok tervezésének célja, hogy az információhoz hasonlóan minden nyersanyag illetve technológiai komponens a megfelelő időben jusson el a megfelelő helyre. Információs rendszerek tervezésénél ez nem csupán a nyersanyagok beszállításának tervezését jelenti, hanem azt is meghatározza, hogy melyek lesznek azok a rendszerkomponensek amelyeket a projektcsoport fejleszt ki, és melyek lesznek azok, amelyeket a hatékonyság növelése érdekében (idő, erőforrás, vagy szakértelem hiányában) készen (off the shelf) vásárolnak meg.

A gyakorlatban a logisztikai folyamatok tervezésénél a szükséges nyersanyag vagy a technológiai komponens azonosítása után megvizsgálják, hogy mely források kerülhetnek szóba az igények kielégítésére. A következő lépés az árajánlatok bekérése, majd az árajánlatok komplex elemzése (ár, minőség, megbízhatóság, idő, garancia stb.) után történik majd a beszállítók kiválasztása és a szerződéskötés.

4.2.17 Az előkészítési szakasz lezárása

Amikor minden tervezési dokumentum rendelkezésre áll, véget ér az előkészítési szakasz. A dokumentumokat összefoglaló néven projektindító dokumentációnak nevezzük, és tartalmazza:

- A projekt céljának megfogalmazását,
- A megvalósíthatóság vizsgálatát,
- A megvalósíthatósági tanulmányt
- A projekt céljainak hierarchiáját
- A projekt hatókörének meghatározását
- Időterv és erőforrásbecslést
- A projektcsoport felállítását
- A feladatok meghatározását
- A feladatmátrix megtervezését
- Kockázatok kezelésének tervezését
- A kommunikáció tervezését
- A logisztikai feladatok tervezését

4.2.18 A végrehajtási szakasz

A következő szakasz a projekt megvalósítására irányul. Minden projekt megvalósításának első lépése a projektindító értekezlet. Az értekezletet a projektmenedzser vezeti és azok kapnak rá meghívást, akik a projektcsoport felállításának szakaszában kiválasztásra kerültek. A projektmenedzser ismerteti a projekt célját, bemutatja egymásnak a projekt megvalósításában résztvevőket. A projektmegbeszélések az indítóértekezlet után is rendszeresek, céljuk a projekt előrehaladásának bemutatása és a felmerült problémák megoldása, illetve a kommunikáció elősegítése. Az értekezletekről jegyzőkönyvek készülnek, amelyeket a résztvevőkön kívül megkap a projektmenedzsment minden tagja.

A változások kezelése

7. A projektek tervezése íróasztalon történik, ezért a megvalósítás szinte minden esetben eltér a tervtől. A változás adódhat a projektterv pontatlanságából (a projektterv számos eleme becsléseken alapul), de adódhat a projekttervezési folyamattól független tényezőből is (pl. árak emelkedése, szabványos megváltozása), de akár a megrendelő is módosíthatja a projekt megvalósítása közben az igényeit.
8. A változások menedzselése nagyon fontos, mert minden változás kihatással lehet a projekt egészére. A változások hatását meg kell tervezni a projekt egészére vonatkozóan és jóvá kell hagynia a projektmenedzsmentnek. A jóváhagyott változtatásokról minden érintett személyt írásban értesíteni kell.

A projektkontrolling

A kontrolling egyik célja, hogy információt gyűjtsön a projekt előrehaladásáról. Erre rendszerint olyan indikátorokat használnak, amelyek megmutatják, hogy a projekt a projektterv szerinti módon a kitűzött cél elérése felé halad-e? Az indikátorok elemzése után meg kell találni, hogy mi lehet az oka a tervtől való eltérésnek, és mely eszközök segítségével lehetséges visszatérni a tervhez. Miután megtalálták és alkalmazták a megfelelő eszközöket, meg kell vizsgálni, hogy valóban elértük-e kívánt hatást, majd a teljes folyamatot dokumentálni kell.

Kockázatok elkerülése

A projekt tervezése során meghatározott kockázati tényezők negatív hatásának megelőzése, kivédése. A projekttervezés szempontjából kockázatnak minősül minden olyan tényező, amely a projekt megvalósulását veszélyeztetheti. A kockázati tényezők elemzése viszont tág határok között biztosíthatja a projekt megvalósítását. A kockázatkezelés tervezése során megpróbálják minél pontosabban azonosítani a lehetséges kockázati tényezőket és olyan indikátorokat létrehozni, amelyek jelezhetik valamelyik kockázati tényező térnyerését.

A kommunikáció működtetése

A projekt eredményes megvalósításához elengedhetetlen az információ zavarmentes áramlásának biztosítása, azaz a kommunikáció. A kommunikáció értelmezhető a projekten belül, a projekt megvalósítói között és a projekten kívül, azaz a projekt megvalósítói és a külső környezet között.

Logisztikai folyamatok működtetése

A logisztikai feladatok működtetésének elsődleges célja, hogy az információhoz hasonlóan minden nyersanyag illetve technológiai komponens a megfelelő időben jusson el a megfelelő helyre. Információs rendszerek tervezésénél ez nem csupán a nyersanyagok, hanem a technológiai komponensek beszállítását is jelenti.

A projekt megvalósítása

A projekt optimális esetben a megvalósítási tervnek megfelelően és a megrendelő igényeinek megfelelően készül el a projekt. Ez rendszerint csak abban az esetben valósul meg, ha a projekt megvalósítása során folyamatos a kommunikáció a projekt megrendelője és a projekt kivitelezői között. Információs rendszerek tervezésénél akkor nyilvánítjuk a projektet késznek, amikor a rendszer átadásra és beüzemelésre kerül. Nagyon fontos megemlíteni, hogy a rend-

szer átadása csak az információs rendszer működtetéséhez elengedhetetlen dokumentációval együtt tekinthető teljesnek.

4.2.19 A projekt zárása

A projekt utolsó szakasza a projekt zárása. A projekt megvalósítása során rengeteg tapasztalat és ismeret halmozódott fel, amelynek dokumentálása nem csak azért fontos, hogy elemezni tudjuk, hogy mely lépések végrehajtása történt zökkenőmentesen, és melyek megvalósítása okozott problémát (és ennek mi lehetett az oka), hanem azért is, mert a legtöbb projekt megvalósítása felfogható tanulási folyamatként is, amely a kollektív szakmai ismereteket gyarapítja egy adott szervezeten belül.

A projekt értékelése többféle szempont alapján történhet, a gyakorlatban az egyik leggyakrabban alkalmazott értékelési metódus Peter Hobbs 2000-ben megjelent Project Management című könyvében olvasható, az evaluation criteria címszó alatt:

Az időgazdálkodás értékelése:

- A projekttervben meghatározott időkeretek között valósult-e meg a projekt?
- Mely területek megvalósítás volt időigényesebb?
- Milyen következtetéseket vonhatunk le a projekt időmenedzsmentjére vonatkozóan?

A költségtervezés értékelése:

- A projekttervben meghatározott költségkeretek között valósult-e meg a projekt?
- Mely projekttevékenységek igényeltek több, és melyek kevesebb anyagi ráfordítást?
- Mi volt az oka annak, ha el kellet térni az eredeti költségtervtől?

A projekt eredményeinek értékelése:

- Mennyire volt elégedett a projekt megrendelője a projekt végeredményével?
- Hogyan lehetett volna a projekttel szembeni elvárásokat precízebben meghatározni?

A projekt értékelése a megvalósításában közreműködők vonatkozásában

- A munkamegosztás megfelelő volt-e a projekt megvalósítói között?
- A projekt megvalósítói megfelelően értelmezték a szerepüket?
- Hatékony volt-e a projekt megvalósítóinak teljesítményértékelése?

A projekt értékelése a kommunikáció szempontjából

- A projekt megvalósítói megfelelő ismeretekkel rendelkeztek a munkamenettel kapcsolatban?
- A projekt megvalósítói időben megkapták a szükséges információt?
- A projekt megvalósítói hamar jelezték a felmerülő problémákat?
- A projekt megvalósítói közül mindenkit bevontak a kommunikációs folyamatba?
- Hogyan lehetne hatékonyabbá tenni a projekt megvalósítói közötti kommunikációt?

A projekt értékelése az alkalmazott módszerek szempontjából

- Hatékonyak voltak a projektspecifikációs és tervezési módszerek?
- Hogyan valósult meg a folyamatok és a változtatások ellenőrzése?
- Került-e új módszer bevezetésre a projekt megvalósítása kapcsán?

4.2.20 A projekt átadását követő tevékenységek meghatározása

A tevékenység célja a projekt megvalósítását követő feladatok meghatározása. Az információs rendszerek tervezésénél elengedhetetlen a az új rendszer finomhangolása, karbantartása, amelynek módjáról és feltételeiről meg kell állapodnia a projekt megrendelőjének és a kivitelezőknek. Különösen igaz ez azokra az információs rendszerekre, amelyeknél az új rendszer nem váltja le azonnal a régi rendszert, hanem egy ideig (adatok konverziójának szükségessége, a munkaerő betanítása stb. miatt) a két rendszernek párhuzamosan kell működnie.

4.2.21 A projektzáró értekezlet

A projektzáró értekezlet célja, hogy a projekt eredményeit és tanulságait megismertessük a projekt megvalósítóival és megköszönjük a résztvevők munkáját. Az eseményről jegyzőkönyv készül, ekkor a résztvevőknek lehetőségük

van véleményüket, tapasztalataikat megosztani a többiekkel. Ennek akkor van különösen nagy jelentősége, ha a projekt célkitűzései közül nem sikerült mindent maradéktalanul megvalósítani.

4.2.22 A projektzáró jelentés

A projektzáró jelentés célja, hogy összefoglalja a projekt megvalósítása során végzett tevékenységeket, bemutassa az elkészült projektet és kitérjen azokra a változtatásokra is, amelyekre a projekt sikere érdekében szükség volt. A zárójelentés elkészítését nagyban segítik a rendszeres projektmegbeszélések jegyzőkönyvei, a projekt előrehaladását monitorozó dokumentumok és a projektzáró értekezlet jegyzőkönyve. A zárójelentés részét képezheti az elkészült projekt átadásának dokumentumai és a kész projekt dokumentációjának rövidített változata.

4.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

4.3.1 Összefoglalás

A negyedik fejezet célja az volt, hogy a projektmenedzsment legfontosabb ismereteit összefoglalja a hallgatók számára. A lecke kapcsán szó volt többek között a projektszervezet felépítéséről, a projekt életciklusról, a projekt szakaszairól. A lecke második részében megtárgyaltuk a projektfolyamat sajátosságait, ejtettünk szót a projekt-előkészítés szakaszairól, a projekt céljának megfogalmazásáról, a megvalósíthatóság vizsgálatáról és a megvalósíthatósági tanulmány elkészítéséről.

A lecke harmadik részében szó volt a célrendszer és sikerkritériumok meghatározásáról, az erőforrásbecslésről és a projekttervezés szakaszairól.

4.3.2 Önellenőrző kérdések

1. Ismertesse a projektmenedzsment lényegi elemeit!
2. Mit tud a projektszervezet felépítéséről?
3. Sorolja fel a projektéletciklus és a projekt szakaszait
4. Ismertesse a projektfolyamat sajátosságait!
5. Ismertesse a projekttervezés szakaszait!

5. LECKE: A RENDSZERFEJLESZTÉSI PROJEKT TERVEZÉSÉNEK KEZDETI LÉPÉSEI

5.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Az ötödik lecke célja, hogy a hallgatókat megismertessük rendszerkövetelmények tervezésének alapvető ismereteivel. Ennek során górcső alá kerül az információs rendszer financiális hasznának vizsgálata, a követelmények feltárása és elemzése, illetve a követelmények felderítése. Beszélünk továbbá arról is, hogy a folyamat végén meg kell vizsgálnunk, hogy a megrendelő által kívánt rendszert definiáltuk-e a folyamat során, mely vizsgálatot röviden validálásnak nevezzük.

5.2 TANANYAG

- A rendszerkövetelmények tervezése
- Az információs rendszer hasznának vizsgálata
- A követelmények feltárása és elemzése
- Követelmények felderítése
- Forgatókönyvek
- Interjúk
- Annak a vizsgálata, hogy a megrendelő által kívánt rendszert definiáltuk-e
- A követelmények felülvizsgálata

5.2.1 A rendszerkövetelmények tervezése

A rendszerkövetelmények tervezési folyamatának célja a rendszerkövetelmények jellemzőit leíró dokumentum létrehozása. A teljes folyamat három fontos követelménytervezési részfolyamatot foglal magába:

- az információs rendszer hasznának vizsgálata,
- a követelmények feltárása és elemzése
- annak a vizsgálata, hogy a követelmények a megrendelő által kívánt rendszert definiálják-e?

5.2.2 Az információs rendszer hasznának vizsgálata

Az információs rendszer financiális hasznát vizsgáló dokumentumot szokták megvalósíthatósági tanulmánynak is nevezni. A megvalósíthatósági tanulmány bemenetéül az üzleti követelmények kezdeti változata szolgál, a rendszer körvonalazott leírása, illetve az, hogyan támogatja majd a rendszer az üzleti folyamatokat. A megvalósíthatósági tanulmány eredményeit ajánlott egy jelentésben összefoglalni, amely javaslatot tesz arra, hogy érdemes-e folytatni a munkát a követelménytervezéssel és a rendszerfejlesztési folyamattal.

21. ábra: A megvalósíthatósági tanulmány elemei

A megvalósíthatósági tanulmány rövid, tömör tanulmány, amely számos kérdést próbál megválaszolni:

Támogatja-e a rendszer a vállalat általános célkitűzéseit?

Megvalósítható-e a rendszer a jelenlegi technológiával adott költségen belül és adott ütemezés szerint?

Integrálható-e a rendszer más, már használatban lévő rendszerekkel?

Az a kérdés, hogy a rendszer hozzájárul-e az üzleti célokhoz vagy sem, igen kritikus. Ha a rendszer nem támogatja a célokat, akkor nincs valódi üzleti értéke.

22. ábra: A megvalósíthatósági tanulmány kérdései

A megvalósíthatósági tanulmány elkészítése magában foglalja az információk felmérését, az információk összegyűjtését és a jelentés megírását. Az információk felmérésének fázisa azonosítja azokat az információkat, amelyek a három fent megjelölt kérdés megválaszolásához szükségesek.

A megvalósíthatósági tanulmányban konzultálhatunk annak az osztálynak a vezetőivel, ahol a rendszert használni fogják, azokkal a rendszerfejlesztőkkel, akik már ismerik az olyan típusú rendszereket, amilyet itt is terveznek használni, továbbá a technológiai szakértőkkel és a rendszer végfelhasználóival.

Az információ megszerzése után megírjuk a megvalósíthatósági tanulmányi jelentést. Ennek már ajánlást is tanácsos tartalmaznia arról, hogy a rendszer fejlesztését érdemes-e folytatni vagy sem. A jelentésben javasolhatunk változtatásokat a termék felhasználási területén, a költségvetésben és a rendszer ütemezésében, emellett ajánlhatunk további magas szintű követelményeket.

5.2.3 A követelmények feltárása és elemzése

A követelménytervezési folyamat következő szakasza a követelmények feltárása és elemzése. E tevékenység során a szoftvertervezők együtt dolgoznak a megrendelőkkel és a rendszer végfelhasználóival azért, hogy felderítsék az alkalmazási szakterületről, hogy milyen szolgáltatásokat kellene biztosítania a rendszernek, mekkora a rendszer szükséges teljesítménye, milyen hardverre vonatkozó megszorítások vannak és így tovább.

A követelmények feltárása és elemzése különféle embereket érinthet a szervezeten belül. A projekt által érintett személy kifejezésen olyan személyt

vagy csoportot értünk, akiket a rendszer közvetve vagy közvetlenül érint. A projekt által érintett személyek közé számítanak a végfelhasználók, akik érintkeznek a rendszerrel, és mindenki más is a szervezeten belül, akire a telepítés hatással lehet. Kulcsfigurái lehetnek még a rendszernek az egyéb, kapcsolódó rendszereket fejlesztő vagy karbantartó mérnökök, üzleti vezetők, szakterületi szakértők, kereskedelmi unió képviselői és így tovább.

A feltárás és a projekt által érintett személyek követelményeinek megértése több szempontból is bonyolult:

1. A kulcsfigurái gyakran nem tudják valójában, mit várnak el a számítógépes rendszertől, a legáltalánosabb fogalmakat kivéve. Nehezen fejezik ki, hogy mit akarnak a rendszertől; valóságtól elrugaszkodott kívánságaik lehetnek, mivel nincsenek tisztában a kéréseik költségeivel.
2. A projekt által érintett személyek természetes módon, saját fogalmaik segítségével fejezik ki követelményeiket, saját munkájuk ismereteit használva. A megrendelő szakterületén szerzett tapasztalatokkal nem rendelkező követelménytervezőknek meg kell érteniük a követelményeket.
3. Az egyes projekt által érintett személyeknek különböző követelményeik vannak, amelyeket különböző módon fejeznek ki. A követelménytervezőknek a követelmények összes lehetséges forrását fel kell kutatniuk, és fel kell fedezniük a közös dolgokat és ellentmondásokat.
4. A rendszerkövetelményeket politikai tényezők is befolyásolhatják. Például a vezetők speciális rendszerkövetelményeket igényelhetnek, amelyek lehetővé tennék a szervezeten belüli befolyásuk növelését.
5. A gazdasági és üzleti környezet, amelyben az elemzés történik, dinamikus. Változása elkerülhetetlen az elemzési folyamat során. Ezáltal bizonyos követelmények fontossága is változhat. Új projekt által érintett személyektől új követelmények is származhatnak, amelyeket még nem vitattak meg.

5.2.4 Követelmények felderítése

A követelmények felderítése az a folyamat, amely során a tervezett vagy meglévő rendszerekről információt gyűjtünk, majd ebből kiszűrjük a felhasználói és rendszerkövetelményeket. A követelmények felderítésének fázisa során információforrásként szolgál a dokumentáció, a rendszer kulcsfigurái és hasonló rendszerek specifikációi. A projekt által érintett személyekkel interjúk és megfigyelések révén tarthatunk kapcsolatot, és a követelmények felderítéséhez forgatókönyveket és prototípusokat használhatunk.

5.2.5 Forgatókönyvek

Az emberek általában könnyebben kezelik a valós életbeli problémákat, mint az absztrakt leírásokat. Képesek megérteni egy forgatókönyvet arról, hogy hogyan érintkezhetnek a szoftverrendszerrel, és képesek kritikát megfogalmazni az ilyen forgatókönyvekről. A követelménytervezők fel tudják használni az ilyen viták során nyert információt az aktuális rendszerkövetelmények megformálásához.

A forgatókönyvek különösen hasznosak akkor, ha további részletekkel szeretnénk kiegészíteni a követelmények körvonalazott leírását. Ezek interakciósorozatok leírásai. Minden forgatókönyv egy vagy több lehetséges interakciót takar. Számos forgatókönyvtípust kifejlesztettek már, ezek mind különböző részletezettségű, különböző típusú információkat nyújtanak a rendszerről.

A forgatókönyv az interakció körvonalazásával kezdődik, a feltárás alatt további részletekkel bővítjük, hogy végül az interakció teljes leírása megszülessen. Legáltalánosabb formájában a forgatókönyv a következőket tartalmazhatja:

1. annak leírása, mit vár a rendszer illetve a felhasználó a forgatókönyv kezdetén;
2. a forgatókönyvbeli események normális menetének leírását;
3. leírást arról, mi romolhat el, és ezt hogyan kezeli a rendszer;
4. információkat egyéb tevékenységekről, amelyek ugyanabban az időben mehetnek végbe;
5. a rendszer állapotának leírását a forgatókönyv befejeződésekor.

A forgatókönyv-alapú feltárás elvégezhető informális módon. Ekkor a követelménytervezők a projekt által érintett személyekkel együtt dolgoznak a forgatókönyvek azonosításán és a forgatókönyv részleteinek megragadásán. A forgatókönyvek megfogalmazhatók szövegesen, kiegészíthetők diagramokkal, képernyőképekkel és így tovább. Másik lehetőségként alkalmazhatunk strukturáltabb megközelítést jelentő esemény-forgatókönyveket vagy használati esettanulmányokat.

5.2.6 Interjúk

A rendszer kulcsfiguráival készített interjúk a legtöbb követelménytervezési folyamatnak részei. Ezekben az interjúkban a követelménytervezési csoport kérdéseket tesz fel a kulcsfiguráknak az általuk használt, illetve a fejlesztendő rendszerről. Az ezekre adott válaszokból követelményeket állítanak elő. Kétféle interjú van:

Zárt interjú, ahol a kulcsfigura egy előre definiált kérdéshalmazt válaszol meg.

Nyílt interjú, ahol nincs előre megadott kérdéssorozat. A követelménytervezési csoport problémák széles körét vizsgálja át a kulcsfigurákkal, így alaposabb ismereteket szereznek az igényeikről.

Az interjúk segítségével átfogó képet kapunk arról, mi az egyes kulcsfigurák munkája, hogyan érintkeznek a rendszerrel, és milyen nehézségekkel szembesülnek a jelenlegi rendszernél. Az interjúk nem igazán jók az alkalmazás szakterületi követelményeinek megértésére.

Interjúkkal nem lehet hatékonyan feltárni a vállalati követelményeket és megszorításokat, ugyanis egy vállalaton belül a kulcsfigurák között szövevényes hatalmi és befolyási viszonyok vannak. A nyilvános vállalati struktúrák ritkán felelnek meg annak, ahogy az adott vállalatnál a döntéshozatal valójában végbemegy-e, azonban az interjúalanyok sokszor nem kívánják elárulni a tényleges (nem a papír szerinti) struktúrát egy idegennek. Az emberek általában vonakodnak a követelményeket érintő politikai és vállalati kérdések tárgyalásától. A hatékony interjúkészítőket két dolog jellemzi:

Nyitottak, kerülik az előítéleteket a követelményekről, és hajlandók odafigyelni a kulcsfigurákra. Ha a kulcsfigura meglepő követelménnyel áll elő, hajlandók megváltoztatni elképzeléseiket a rendszerről.

A beszélgetést kérdésfeltevéssel vagy követelményjavaslattal indítják, vagy felkínálják, hogy dolgozzanak együtt a rendszer egy prototípusán. A „mondjon, amit akar" típusú kérdések nem kecsegtetnek hasznos információval. A legtöbb ember sokkal könnyebben beszél konkrét dolgokról, mint általánosságban.

Az interjúkból nyert információk kiegészítik a dokumentumokból, felhasználók megfigyeléséből stb. kapott információkat. Néha – a dokumentumokból nyert információktól eltekintve – az interjúk jelentik az egyetlen információforrást a rendszerkövetelményekről. Ugyanakkor az interjút csak önmagában alkalmazva könnyen kimaradhatnak lényeges információk is, ezért azt más követelmény-feltárási technikák mellett szabad csak végezni.

5.2.7 Annak a vizsgálata, hogy a megrendelő által kívánt rendszert definiáltuk-e

Azt az eljárást, amikor megvizsgáljuk, hogy a megrendelő által kívánt rendszert dediniáltuk-e, röviden a követelmények validálásának nevezzük. A követelmények validálása azzal foglalkozik, hogy a követelmények valóban azt a rendszert definiálják, amit a megrendelő akar. A követelmények validálása át-

fedi az elemzést, mivel célja, hogy megtalálja a követelményekkel kapcsolatos problémákat. A követelmények validálása azért lényeges, mert a követelménydokumentumbeli hibák jelentős átdolgozási költségekhez vezethetnek, ha azokat a fejlesztés alatt vagy már a rendszer beüzemelése után vesszük észre. A rendszer követelményproblémából eredő megváltoztatásának költsége sokkal magasabb, mint a tervnek vagy kódolási hibáknak a kijavításáé. Ennek az az oka, hogy a követelmények megváltozása általában azzal jár, hogy a rendszertervet és az implementációt is meg kell változtatni, a rendszert pedig újra tesztelni kell.

A követelményvalidálási folyamat során ajánlott ellenőrzéseket végeznünk a dokumentumbeli követelményeken. Ezek az ellenőrzések magukban foglalják a következőket:

1. Validitás-ellenőrzések. A felhasználó azt gondolhatja, hogy egy rendszernek meghatározott funkciókat kell ellátnia. Ugyanakkor, a további gondolkodás és elemzés újabb vagy más szükséges funkciókat azonosíthat. A rendszereknek sokféle kulcsfigurája van, eltérő igényekkel, és a követelmények bármilyen csoportja elkerülhetetlenül a kulcsfigurák közösségének egészével kötött kompromisszum lesz.
2. Ellentmondás-mentességi ellenőrzések. A dokumentumban szereplő követelmények nem mondhatnak ellent egymásnak. Azaz, ne legyenek ellentmondó megszorításai vagy leírásai ugyanannak a rendszerfunkciónak.
3. Teljesség-ellenőrzések. A követelménydokumentumnak javasolt tartalmaznia mindazon követelményt, amely a rendszer felhasználói által kért összes funkciót és megszorítást definiálja.
4. Megvalósíthatósági ellenőrzések. Létező technológiák ismereteit felhasználva ellenőriznünk kell a követelményeket, hogy megbizonyosodjunk afelől, hogy azok tényleg megvalósíthatók-e. Ezeknek az ellenőrzéseknek ki kell terjedniük a költségvetésre és a rendszerfejlesztés ütemtervére is.
5. Verifikálhatóság. Hogy csökkentsük a megrendelő és a vállalkozó közötti viták lehetőségét, mindig ellenőrizhető módon kell rögzíteni a rendszerkövetelményeket. Ez azt jelenti, hogy meg kell tudnunk írni egy olyan követelményhalmazt, amellyel bizonyítani lehet, hogy az átadott rendszer teljesíti az összes előírt követelményt.

A követelmények validálására számos technika létezik, melyek együtt vagy egyedileg használhatók:

1. Követelmények felülvizsgálata. A követelményeket módszeresen elemzi felülvizsgálók egy csoportja. Ezt a folyamatot a következő fejezetben tárgyaljuk.
2. Prototípus-készítés. Ebben a validálási folyamatban a rendszer egy végrehajtható modelljét mutatjuk be a végfelhasználóknak és a megrendelőknek, így tapasztalatokat szerezhetnek a modellel kapcsolatban, hogy lássák, vajon kielégíti-e a valós igényeket.
3. Teszteset generálása. A követelményeknek tesztelhetőeknek kell lenniük. Amennyiben a követelmények tesztjeit a validálási folyamat részeként tervezték kivitelezni, az gyakran követelményi problémákat fed fel. Ha a tesztet nehéz vagy lehetetlen megtervezni, az általában azt jelzi, hogy a követelmények nehezen implementálhatók, és tanácsos azokat újra átgondolni. Az extrém programozás szerves része, hogy azelőtt kell teszteket készíteni a felhasználói követelményekből, mielőtt bármilyen kód meg lenne írva.

A követelményvalidálás nehézségeit nem szabad alábecsülnünk. Igen nehéz azt megmutatni, hogy a követelmények egy halmaza valóban kielégíti a felhasználó igényeit. A felhasználóknak a működő rendszerről kell képet alkotniuk, és azt kell elképzelniük, hogyan illeszkedik majd a rendszer az ő munkájukba. Gyakorlott számítógépes szakemberek számára is igen kemény feladat egy ilyen absztrakt elemzés elvégzése, a rendszer felhasználói számára pedig még keményebb. Következésképpen, a követelmények validálása ritkán derít fel minden követelményekkel kapcsolatos problémát. Elkerülhetetlenek a követelménydokumentum elfogadása utáni változtatások a hiányosságok és félreértések kijavítására.

5.2.8 A követelmények felülvizsgálata

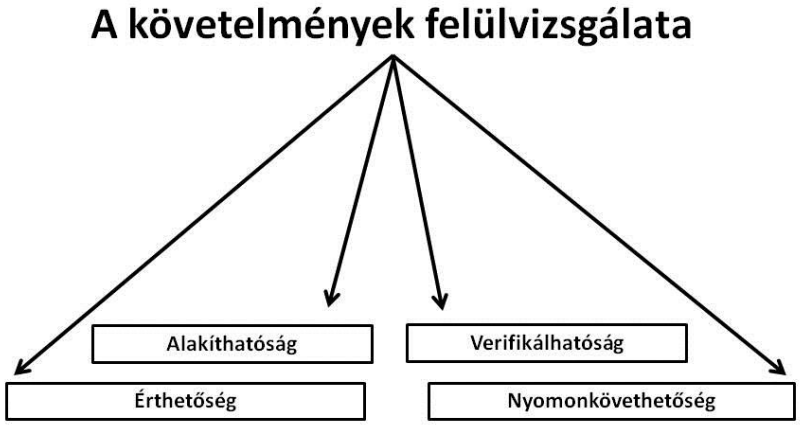
A követelmények felülvizsgálata manuális folyamat, amelynek során mind a megrendelő, mind a partnervállalatok részéről ellenőrzik a követelménydokumentumot a rendellenességek és a hiányosságok megtalálása céljából. A felülvizsgálati folyamat végezhető ugyanúgy, akár a programok felülvizsgálata. Emellett megszervezhető olyan széleskörű tevékenységként is, amely során különböző emberek a dokumentum különböző részeit ellenőrzik.

A követelmények felülvizsgálata történhet informálisan vagy formálisan. Az informális felülvizsgálat egyszerűen csak a vállalkozókat érinti, akik a követelményeket annyi kulcsfigurával vitatják meg, ahánnyal csak lehetséges. Meglepő, hogy a rendszer fejlesztői és felhasználói közötti kommunikáció milyen sokszor fejeződik be a feltárás után, és nincs semmiféle megerősítés arról, hogy a dokumentált követelmények valóban azok, amelyeket a kulcsfigurák mondtak

és akartak. Számos probléma deríthető fel egyszerűen úgy, hogy beszélgetünk a rendszerről a kulcsfigurákkal, még mielőtt elköteleznénk magunkat a formális felülvizsgálat irányában.

A formális követelmény-felülvizsgálatban a fejlesztő csoportnak „végig kell vezetnie" az ügyfelet a rendszerkövetelményeken, elmagyarázva minden egyes követelmény következményeit. A felülvizsgálati csoportnak ajánlott minden követelményt ellenőriznie ellentmondás-mentességi szempontból, és javallott a követelményeket mint egészet is ellenőriznie teljességi szempontból. A felülvizsgálók szintén ellenőrizhetik a következőket:

- Verifikálhatóság. Tesztelhető-e a rendszer valósághűen a jelenlegi formájában?
- Érthetőség. Helyesen értették-e meg a követelményt a rendszer vásárlói és végfelhasználói?
- Nyomon követhetőség. Világosan van-e megfogalmazva a követelmény eredete? Lehet, hogy vissza kell mennünk a követelmény forrásáig ahhoz, hogy megbecsülhessük egy változtatás hatását. A nyomon követhetőség azért fontos, mert lehetővé teszi, hogy megbecsüljük, milyen hatással van egy változtatás a rendszer többi részére.
- Alakíthatóság. Alakítható-e a követelmény? Azaz megváltoztatható-e a követelmény anélkül, hogy ez a többi rendszerkövetelményre nagymértékű hatással lenne?



23. ábra: Követelmények felülvizsgálata

A követelményekben lévő ütközésekre, ellentmondásokra, hibákra és hiányosságokra tanácsos a felülvizsgálóknak rámutatniuk, és azokat a felülvizsgála-

ti jelentésben formálisan feljegyezniük. Azután már a felhasználók, a megrendelők és a rendszerfejlesztők feladata, hogy tárgyalnak-e az így azonosított problémák megoldásáról.

5.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

5.3.1 Összefoglalás

Az ötödik lecke célja az volt, hogy a hallgatókat megismertessük rendszerkövetelmények tervezésének alapvető ismereteivel. Ennek során górcső alá került az információs rendszer financiális hasznának vizsgálata, a követelmények feltárása és elemzése illetve a követelmények felderítése. Beszéltünk továbbá arról is, hogy a folyamat végén meg kell vizsgálnunk, hogy a megrendelő által kívánt rendszert definiáltuk-e a folyamat során.

5.3.2 Önellenőrző kérdések

1. Foglalja össze rendszerkövetelmények tervezésének lépéseit
2. Mit jelent az információs rendszer financiális hasznának vizsgálata?
3. Ismertesse a követelmények feltárásának és elemzésének menetét!
4. Mit nevezünk validálásnak?

6. LECKE: A RENDSZERÖSSZETEVŐK MEGHATÁROZÁSA ÉS IMPLEMENTÁCIÓ

6.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A hatodik lecke célja, hogy megismertesse a hallgatókat a szoftvertervezés és implementáció legfontosabb lépéseivel. Szó lesz az architekturális tervezéshez kapcsolódóan az architekturális modellekről, a modularizálás és moduláris tervezés folyamatáról. A lecke második részében az alrendszerek modulokra bontása kapcsán megismerkedünk az objektumorientált és adatfolyam modell felbontással, annak előnyeivel és hátrányaival. A lecke végén a vezérlési stílusokról, ezen belül a központosított vezérlésről és az eseményvezérelt rendszerekről beszélünk.

6.2 TANANYAG

- Szoftvertervezés és implementáció
- Az architekturális tervezés
- Architekturális modellek
- Modularizálás, moduláris tervezés
- Az alrendszerek modulokra bontása
- Objektumorientált felbontás
- Az objektumorientált megközelítés előnyei
- Az objektumorientált megközelítés hátrányai
- Az adatfolyam modell
- Az adatfolyam modell előnyei
- Az adatfolyam modell hátrányai
- Vezérlési stílusok
- Központosított vezérlés
- Eseményvezérelt rendszerek

6.2.1 Szoftvertervezés és implementáció

A rendszerkövetelmények meghatározásának eredményeképpen rendelkezésre áll az a dokumentáció, amely alapján világossá válik, hogy a rendszer milyen összetevőkből épül fel, és hozzá lehet kezdeni a rendszer megvalósításához. A szoftvertervezés és implementáció a szoftverfejlesztési folyamatnak a rendszerspecifikáció alapján, futtatható rendszerré történő konvertálása. Ez a folyamat magába foglalja a szoftvertervezést és a szoftverfejlesztést, illetve bizonyos esetekben tartalmazhatja a specifikáció finomítását is. A szoftver tervezése magába foglalja a szoftver struktúrájának és az adatoknak a meghatározását, valamint a komponensek közötti interfészek és bizonyos esetekben a használt algoritmusok megadását is.

A tervezés iteratív módon történik több verzión keresztül. A tervezési folyamat számos különféle absztrakciós szinten lévő rendszermodell kifejlesztését is tartalmazhatja, és a tervezési folyamat szakaszai átfedhetik egymást. A tervezési folyamat egyik legfontosabb eleme architekturális tervezés.

6.2.2 Az architekturális tervezés

Az architekturális tervezés során elő kell állítani a rendszerszerkezetet, amely megfelel a rendszerkövetelményeknek. Egy szoftverrendszer architektúrája egy bizonyos architekturális modellen vagy stíluson alapulhat. Az architekturális stílus a rendszer szervezésének egy mintája, mint amilyen például a kliens-szerver vagy a rétegzett architektúra. Ezen stílusoknak, alkalmazási korlátaiknak, erősségeiknek és gyenge pontjaiknak az ismerete roppant fontos. A legtöbb nagy rendszer architektúrája azonban nem egyetlen stílus alapján épül fel, a rendszer különböző részeit különböző architekturális stílus segítségével lehet megtervezni. Bizonyos esetekben a teljes rendszerarchitektúra összetett lehet, amelyet különböző architekturális stílusok kombinálásával kapunk.

Az architekturális tervezés során ki kell választanunk a követelményeknek kielégítését lehetővé tévő legmegfelelőbb szerkezetet. A rendszer egységeinek modulokra bontásához meg kell határoznunk az alrendszerek komponensekre vagy modulokra bontásának stratégiáját. A használható megközelítések különféle architektúrákat tesznek lehetővé. Végül a vezérlés modellezése során azt kell eldöntenünk, hogy hogyan vezéreljük az alrendszerek végrehajtását, végül ki kell alakítani a létrejött rendszer részei közötti vezérlőkapcsolatok általános modelljét.

Egy architekturális tervet nehéz értékelni, mivel az architektúra igazi tesztjét a telepítés után tudjuk elvégezni, vizsgálva a funkcionális és nemfunkcionális követelményeknek történő megfelelést. Bizonyos esetekben azonban értékel-

hetjük a tervünket referenciamodellekkel vagy általános architekturális modellekkel történő összehasonlítás segítségével.

Az architekturális tervezés folyamatának eredménye egy architekturális tervezési dokumentum. Ez a rendszer grafikus reprezentációit tartalmazza, a hozzájuk kapcsolódó leíró szöveggel együtt. Tartalmaznia kell annak leírását, hogy a rendszert hogyan lehet alrendszerekre bontani, az egyes alrendszerek miképpen oszthatók modulokra, és ezekhez milyen megközelítést alkalmazunk. A rendszer grafikus modelljei az architektúra különböző nézőpontjait tükrözik.

6.2.3 Architekturális modellek

A kialakítandó architekturális modellek többek között az alábbiak lehetnek:

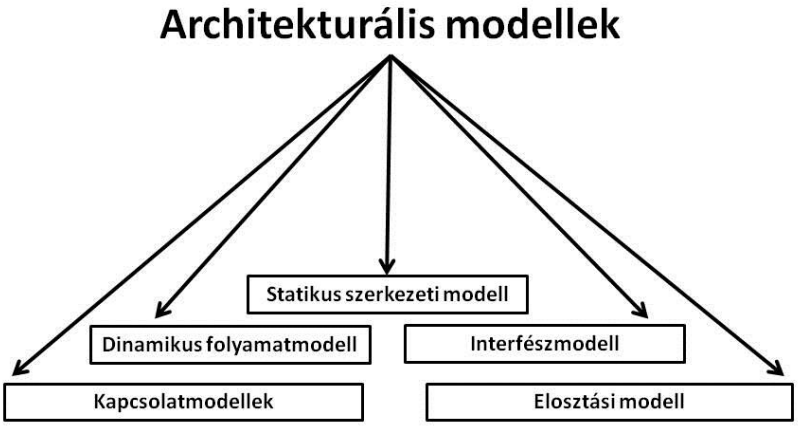
A **statikus szerkezeti modell** azt mutatja meg, hogyan lehet az alrendszereket és komponenseket különálló egységként fejleszteni.

A **dinamikus folyamatmodell** azt ábrázolja, hogy a rendszer hogyan szervezhető futási idejű folyamatokba. Ez különbözhet a statikus modelltől.

Az **interfészmodell** az alrendszerek által publikus interfészeiken keresztül nyújtott szolgáltatásait írja le.

A **kapcsolatmodellek** az alrendszerek közötti kapcsolatokat (például adatáramlás) mutatják be.

Az **elosztási modell** azt adja meg, hogy az egyes alrendszereket hogyan kell elosztani a számítógépek között.

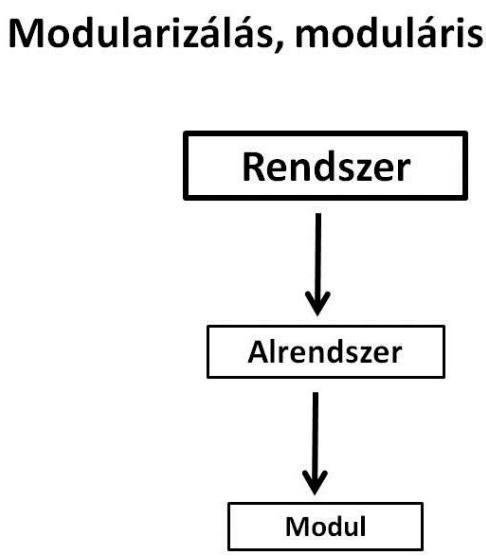


24. ábra: Architekturális modellek

A rendszer-architektúrák leírására számos kutató az architektúraleíró nyelvek (architectural description languages, ADL) használatát javasolja. Az ADL-ek alapelemei komponensek és összekötők (konnektorok), valamint szabályokat és irányelveket tartalmaznak a jól formált architektúrák építésére vonatkozóan. A specializált nyelvekhez hasonlóan azonban az ADL-eket csak szakértők értik, a szakterület és az alkalmazás szakemberei nem, ami bonyolulttá teszi elemzésüket. A gyakorlatban architekturális leíráshoz éppen ezért az olyan informális modelleket és jelölésrendszereket fogják használni, mint amilyen az UML.

6.2.4 Modularizálás, moduláris tervezés

A modularizálás fogalma alatt a rendszer olyan, kisebb részekre (modulokra) való tagolását értjük, amelynél a modulok közötti kapcsolatok is megfogalmazottak a modulok egy későbbi, többszörös alkalmazásának érdekében.



25. ábra: Modularizálás, moduláris tervezés

Az információs rendszerekben az alrendszerek és modulok nem különböztethetők meg egyértelműen, de az érdemes azokat a következő módon elképzelni:

Az **alrendszer** egy olyan önálló rendszer, amely működése nem függ más alrendszerek szolgáltatásaitól. Az alrendszerek modulokból épülnek fel, és az egyéb alrendszerekkel interfészeken keresztül kommunikálnak.

A **modul** olyan rendszerkomponens, amely más modulok számára szolgáltatásokat biztosít. Általában nem tekintjük önálló rendszernek. A modulok rendszerint egyszerűbb rendszerkomponensekből épülnek fel.

6.2.5 Az alrendszerek modulokra bontása

Az alrendszerek modulokra bontása során két fő stratégia ismeretes:

- Objektumorientált felbontással a rendszert egymással kommunikáló objektumok halmazára bontjuk fel.
- Adatfolyam modell használatával a rendszert bemenő adatokat elfogadó és azokat kimenő adatokká alakító funkcionális modulokra bontjuk.

Az objektumorientált megközelítésben a modulok egyéni állapottal, valamint az ezeken az állapotokon értelmezett műveletekkel rendelkező objektumok. A csővezetékelvű modellben a modulok funkcionális transzformációk. A modulok mindkét esetben szekvenciális komponensként vagy folyamatként valósíthatók meg.

Lehetőség szerint nem szabad elhamarkodottan döntenünk a rendszer konkurenciájáról. A konkurens rendszertervezés elkerülésének előnye, hogy a szekvenciális programok könnyebben tervezhetők, implementálhatok, ellenőrizhetők és tesztelhetők, mint a párhuzamos rendszerek. A folyamatok közötti időbeli függőség nehezen formalizálható, vezérelhető és ellenőrizhető. A legjobb megoldás az, hogy először modulokra kell bontani a rendszereket, majd az implementáció során kell eldönteni, hogy azokat szekvenciálisan vagy párhuzamosan kell-e végrehajtani.

6.2.6 Objektumorientált felbontás

Egy objektumorientált architekturális modell a rendszert lazán kapcsolódó, jól definiált interfészekkel rendelkező objektumok halmazára tagolja. Az objektumok a többi objektum által biztosított szolgáltatásokat hívják.

Az objektumorientált felbontás objektumosztályokkal, azok attribútumaival és műveleteivel foglalkozik. Implementációkor az objektumok ezekből az osztályokból jönnek létre, és az objektum műveleteinek koordinálásához valamilyen vezérlési modellt alkalmaznak.

6.2.7 Az objektumorientált megközelítés előnyei

Az objektumorientált megközelítés előnyei jól ismertek. Mivel az objektumok lazán kapcsolódnak, az objektumok implementációja változtatható anélkül, hogy az hatással lenne más objektumokra. Az objektumok gyakran a valós világ

egyedeinek reprezentációi, így a rendszer szerkezete könnyen megérthető. Mivel a valós világ egyedei különböző rendszerekben is létezhetnek, az objektumok újrafelhasználhatók. Az architekturális komponensek közvetlen implementációjának biztosítására objektumorientált programozási nyelveket fejlesztettek ki.

6.2.8 Az objektumorientált megközelítés hátrányai

A szolgáltatások használata érdekében az objektumoknak explicit módon kell hivatkoznia a többi objektum nevére és interfészére. Ha a rendszer tervezett változtatásához az interfész megváltoztatására van szükség, akkor a változtatást minden, a megváltozott objektumot használó helyen át kell vezetni. Míg az objektumok tisztán leképezhetők egyszerű valós világbeli egyedekre, az összetettebb egyedeket gyakran nehezen lehet objektumként reprezentálni

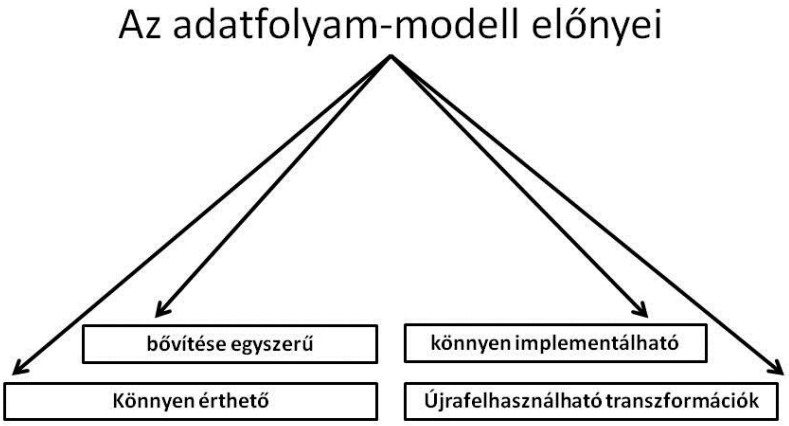
6.2.9 Az adatfolyam modell

Az adatfolyam modell használata esetén funkcionális transzformációk dolgozzák fel bemenetüket és hoznak létre kimeneteket, közöttük áramlanak az adatok és sorban előrehaladva kerülnek átalakításra. Minden feldolgozási lépés transzformációként valósul meg. A bemeneti adatok ezeken a transzformációkon mennek keresztül, míg végül átalakulnak kimeneti adatokká. A transzformációk mind szekvenciálisan, mind pedig párhuzamosan végrehajthatók. Az adatok egyesével vagy kötegelve is feldolgozhatók.

Az adatfolyam modell különböző változatait azóta alkalmazzák, mióta a számítógépeket először használták automatikus adatfeldolgozásra. Ha a transzformáció szekvenciális, az adatfeldolgozás pedig kötegelt, akkor ezt az architekturális modellt kötegelt szekvenciális modellnek nevezzük. Az adatfeldolgozó rendszerek általában nagyszámú bemeneti rekordból egyszerű számításokkal sok kimeneti jelentést hoznak létre.

6.2.10 Az adatfolyam-modell előnyei

- Támogatja a transzformációk újrafelhasználhatóságát.
- Könnyen érthető abban az értelemben, hogy sok ember a saját munkáját a bemenetek és kimenetek feldolgozásaként értelmezi.
- A rendszer új transzformációk hozzáadásával történő bővítése általában egyszerű.
- Mind konkurens, mind pedig szekvenciális környezetben könnyen implementálható.



26. ábra: Az adatfolyam-modell előnyei

6.2.11 Az adatfolyam modell hátrányai

A modell alapvető hátránya az összes transzformáció által felismerhető közös adatátviteli formátum igényéből származik. Az egyes transzformációknak vagy egyeztetniük kell egymással az átadandó adatok formátumát, vagy a kommunikációban részt vevő adatok számára egy szabványos formátumot kell kialakítani. Az utóbbi csak önálló és újrafelhasználható transzformációk esetén használható.

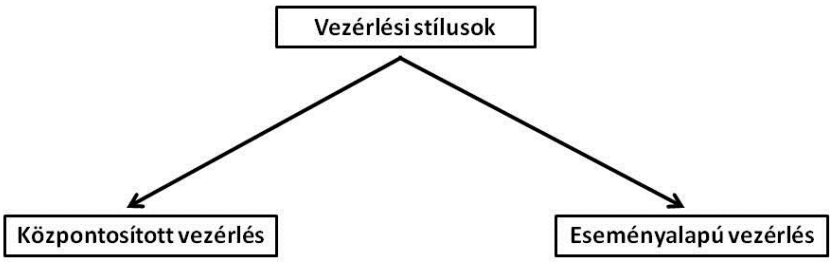
Az interaktív rendszereket nehéz adatfolyam modell alapján megírni a feldolgozandó adatáram iránti szükséglet miatt. Míg az egyszerű szöveges be-, illetve kimenet modellezhető ily módon, a grafikus felhasználói felületek sokkal összetettebb be- és kimeneti formátumokat és olyan, eseményeken alapuló vezérlést tartalmaznak, mint amilyen az egérkattintás és a menükiválasztás. Ezt nehéz az adatfolyam modellel kompatibilis formára hozni.

6.2.12 Vezérlési stílusok

Egy rendszer strukturális modelljei azzal foglalkoznak, hogy a rendszer hogyan bontható fel alrendszerekre. Ahhoz, hogy az alrendszerek rendszerként működjenek, úgy kell őket vezérelni, hogy szolgáltatásaik a megfelelő helyre a megfelelő időben eljussanak. A strukturális modellek nem tartalmaznak vezérlési információkat.

Ehelyett a rendszer kiépítőjének az alrendszereket az alkalmazott strukturális modellt kiegészítő valamilyen vezérlési modellnek megfelelően kell szer-

veznie. Az architekturális szinten elhelyezkedő vezérlési modellek az alrendszerek közötti vezérlési folyamatokkal foglalkoznak.



27. ábra: Vezérlési stílusok

A szoftverrendszerekben két általános vezérlési stílust alkalmaznak:

Központosított vezérlés:

A vezérlés teljes felelősségét egyetlen alrendszer látja el, amely beindítja és leállítja a többi alrendszert. A vezérlést át is háríthatja egy másik alrendszerre, de elvárás, hogy a vezérlés felelőssége visszatérjen hozzá.

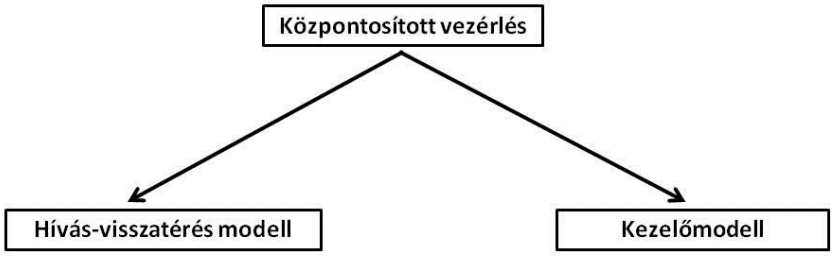
Eseményalapú vezérlés:

Ahelyett, hogy a vezérlési információ egyetlen alrendszerbe lenne beágyazva, minden alrendszer válaszolhat egy külsőleg létrejött eseményre. Ezek az események vagy más alrendszerekből, vagy a rendszer környezetéből származhatnak.

A vezérlési stílusok kiegészítik a strukturális stílusokat. Minden korábban bemutatott strukturális stílust meg lehet valósítani központosított és eseményalapú vezérléssel egyaránt.

6.2.13 Központosított vezérlés

A központosított vezérlési modellben van egy kitüntetett alrendszer, a rendszervezérlő, amely a többi alrendszer végrehajtásáért felelős. A központosított vezérlési modellek két csoportba oszthatók attól függően, hogy a vezérelt alrendszerek szekvenciálisan vagy párhuzamosan hajtódnak-e végre.



28. ábra: Központosított vezérlés

1. A hívás-visszatérés modell:

Ez a megszokott fentről le alprogram modellje, ahol a vezérlés az alprogram-hierarchia csúcsán kezdődik, és alprogramhívások segítségével jut el a fa alsóbb szintjeire. Ez a modell csak szekvenciális rendszerek esetén alkalmazható.

2. A kezelőmodell:

Konkurens rendszerekre alkalmazható. Ebben egy kijelölt rendszerkezelő rendszerkomponens irányítja a többi rendszerfolyamat indítását, leállítását, valamint koordinálja azokat. A folyamat olyan alrendszer vagy modul, amely végrehajtható más folyamatokkal párhuzamosan. Ez a fajta modell használható szekvenciális rendszerek esetén is, ahol egy kezelőrutin, állapotváltozók értékeitől függően, meghív egyes alrendszereket. Ezt általában többirányú elágaztató utasítással valósítják meg.

A hívás-visszatérés modell modulszinten használható a tevékenységek és objektumok vezérlésére. Sok objektumorientált rendszerben az objektumok műveletei (metódusok) eljárásként vagy függvényként vannak implementálva. Például amikor egy Java-objektum igénybe szeretné venni egy másik objektum szolgáltatásait, azt a megfelelő módszer meghívásával teszi.

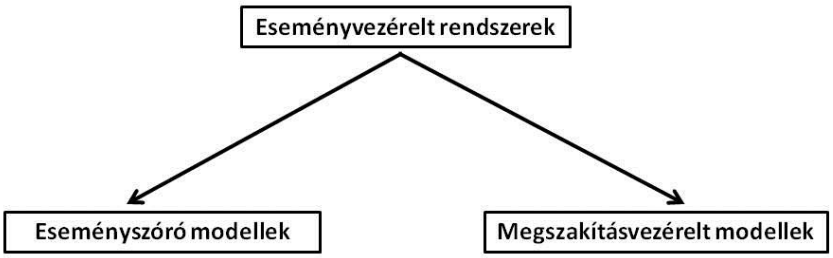
A modell merev és korlátozott természete egyben előny és hátrány is. Erőssége a vezérlési folyam elemzésének és bizonyos bemenet esetén a rendszer válasza kiszámíthatóságának viszonylagos egyszerűsége. Hátránya, hogy a normálistól eltérő működés esetén használata kényelmetlen.

A kezelőmodellt gyakran használják olyan „gyengén" valós idejű rendszerekben, ahol nincsenek nagyon szűk időkorlátok. A központi vezérlő irányítja az érzékelőkkel és működtetőkkel kapcsolatban álló folyamatok végrehajtását.

A rendszert vezérlő folyamat a rendszer állapotváltozói alapján dönti el, hogy az egyes folyamatokat mikor kell elindítani, illetve leállítani. Ellenőrzi, hogy a többi folyamat hozott-e létre feldolgozandó vagy feldolgozásra továbbküldendő információt. A vezérlő általában ciklikusan ellenőrzi az érzékelőket és folyamatokat, hogy bekövetkezett-e valamilyen esemény vagy állapotváltozás. Éppen ezért ezt a modellt eseményciklus-modellnek is szokás nevezni.

6.2.14 Eseményvezérelt rendszerek

Központosított vezérlési modellekben a vezérlési döntéseket általában a rendszer néhány állapotváltozójának az értéke határozza meg. Ezzel szemben az eseményvezérelt vezérlési modelleket külsőleg létrehozott események irányítják. Ebben a környezetben az esemény fogalma alatt nem csupán egy bináris jelet kell érteni, hanem lehet egy értéktartományt befutó jel vagy egy menüből kapott parancs is. Egy esemény és egy egyszerű bemenet között az a különbség, hogy az esemény időzítése az őt kezelő folyamat fennhatóságán kívül helyezkedik el.



29. ábra: Eseményvezérelt rendszerek

Ebben a szakaszban két eseményvezérelt vezérlési modellt mutatunk be:

Eseményszóró modellek. Ezekben a modellekben egy esemény minden alrendszerhez eljut, és bármelyik, az esemény kezelésére programozott alrendszer reagálhat rá.

Megszakításvezérelt modellek. Ezeket kizárólag olyan valós idejű rendszerekben használják, ahol egy megszakításkezelő észleli a külső megszakításokat. Ezek aztán valamely más komponenshez kerülnek feldolgozásra.

Az eseményszóró modellek igen hatékonyan integrálják az egy hálózat különböző számítógépei között osztott alrendszereket. A megszakításvezérelt modelleket szigorú ütemezési követelményekkel rendelkező valós idejű rendszerekben használják.

Az eseményszóró megközelítésnek előnye, hogy evolúciója viszonylag egyszerű. Események bizonyos osztályának kezelését végző új alrendszerek eseményeik eseménykezelőnél történő regisztrálásával integrálhatók. Bármely alrendszer aktiválhat bármely egyéb alrendszert anélkül, hogy tudná annak nevét vagy helyét. Az alrendszerek osztott gépeken implementálhatók. Ez az osztottság a többi alrendszerre nézve átlátszó.

A modell hátránya, hogy az alrendszerek nem tudják, hogy az eseményeket kezeli-e valamilyen eszköz, és ha igen, mikor. Amikor egy alrendszer létrehoz egy eseményt, nem tudja, hogy mely alrendszerek jelezték érdeklődésüket az adott eseményre. Lehetséges, hogy különböző alrendszerek ugyanarra az eseményre regisztrálták magukat. Ez konfliktushoz vezethet, amikor az eseménykezelés eredményei hozzáférhetővé válnak.

A külsőleg létrejött események nagyon gyors kezelését igénylő valós idejű rendszereknek eseményvezéreltnek kell lenniük. Például egy autó biztonsági rendszereit vezérlő valós idejű rendszernek fel kell ismernie egy ütközést, és fel kell fújnia a légzsákot, mielőtt a sofőr feje a kormánykeréknek ütközik. Ahhoz, hogy biztosítva legyen az eseményekre történő ilyen gyors válaszadás, megszakításalapú vezérlést kell alkalmazni.

A megszakításalapú vezérlés modelljében adott ismert számú megszakítástípus, valamint mindegyik típus számára egy kezelő. Minden megszakítástípus rendelkezik egy memóriaterülettel, ahol kezelőjének címe helyezkedik el. Amikor egy bizonyos típusú megszakítás érkezik, egy hardverkapcsoló a vezérlést azonnal a kezelőhöz továbbítja. Ez a megszakításkezelő ezután a megszakítással jelzett eseményre adandó válasz gyanánt egyéb folyamatokat indíthat el, illetve állíthat le.

Ez a modell csak olyan bonyolult valós idejű rendszerekben használható, ahol néhány eseményre azonnali válasz szükséges. Kombinálható is a központosított kezelő modellel: a központi kezelő a rendszer normál működését kezeli, míg váratlan események esetén a megszakításalapú vezérlés alkalmazható.

6.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

6.3.1 Összefoglalás

A hatodik lecke célja az volt, hogy megismertesse a hallgatókat a szoftvertervezés és implementáció legfontosabb lépéseivel. Szó esett az architekturális tervezéshez kapcsolódóan az architekturális modellekről, a modularizálás és moduláris tervezés folyamatáról. A lecke második részében az alrendszerek modulokra bontása kapcsán megismerkedtünk az objektumorientált és adatfo-

lyam modell felbontással, annak előnyeivel és hátrányaival. A lecke végén a vezérlési stílusokról, ezen belül a központosított vezérlésről és az eseményvezérelt rendszerekről beszéltünk.

6.3.2 Önellenőrző kérdések

1. Ismertesse az architekturális tervezés lényegét!
2. Magyarázza el a modularizálás funkcióját!
3. Ismertesse objektumorientált megközelítés előnyeit!
4. Ismertesse az objektumorientált megközelítés hátrányait!
5. Ismertesse az adatfolyam modell előnyeit!
6. Ismertesse az adatfolyam modell hátrányait!
7. Ismertesse vezérlési stílusokat!

7. LECKE: ADATBÁZIS-TERVEZÉS

7.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A hetedik lecke célja az, hogy megismertessük a hallgatókat az adatbázis-tervezés legfontosabb lépéseivel. A lecke során beszélünk az adatbázis-kezelés alapjairól, melynek kapcsán többek között szóba kerül az adatbázisok felépítése, a tulajdonságtípusok, a kapcsolatok és a kapcsolatok tipizálása illetve a kapcsolattípusok. A lecke második részében szót ejtünk az adatmodellek jellemzőiről, közelebbről megvizsgáljuk a relációs adatmodell tulajdonságait. A lecke végén pedig az adatbázisrendszerek tervezési lépéseiről ejtünk szót.

7.2 TANANYAG

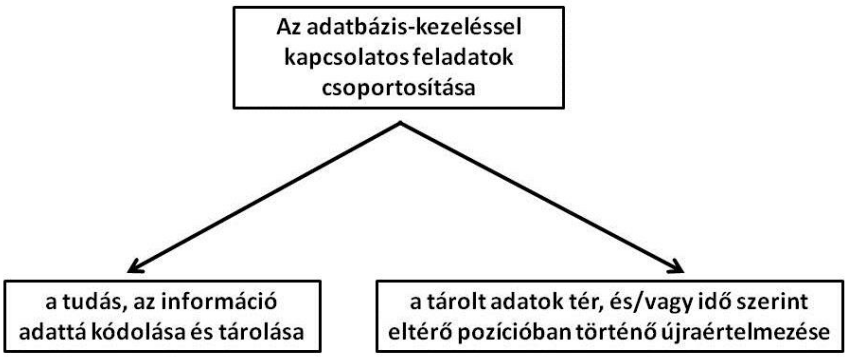
- Az adatbázis-kezelés alapjai
- Az adatbázisok felépítése
- Egyed és tulajdonság
- Tulajdonságtípus
- Egyedtípus
- Kapcsolatok
- A kapcsolatok tipizálása, kapcsolattípusok
- Az adatmodell
- A relációs adatmodell
- Speciális mezők a relációs adatmodellben
- Az adatbázis
- Az adatbázis-rendszerek tervezési lépései

7.2.1 Az adatbázis-kezelés alapjai[1]

Tudásközpontú világunk egyik fő jellemzője az információ, a tudás felértékelődése. E világban kiemelt szerepet kap a megszerzett tudás informatikai eszközökkel történő rögzítése, tárolása és a tárolt tudás hatékony újrafelhasználása. A tudás rögzítése azonban egyáltalán nem kézenfekvő dolog, hiszen tudjuk, hogy a rendelkezésünkre álló információ csak adattá kódolva válik átadhatóvá, tárolhatóvá, kezelhetővé. Éppen ebben rejlik az adatok tárolásának és

[1] Szabó Bálint: Az adatbázis-kezelés alapjai című könyve alapján, a szerző engedélyével.

későbbi információvá értelmezésének, azaz az adatbázis-kezelésnek a nehézsége, egyben jelentősége.



30. ábra: Az adatbázis-kezeléssel kapcsolatos feladatok csoportosítása

Az adatbázis-kezeléssel kapcsolatos feladatok két fontos csoportba sorolhatók:

- a tudás, az információ adattá kódolása és tárolása, illetve
- a tárolt adatok tér, és/vagy idő szerint eltérő pozícióban történő újraértelmezése.

A tudás tárolásakor:

- ismereteinket adattá kódoljuk,
- az adatokat informatikai eszközökkel,
- jól meghatározott szerkezetben,
- hatékonyan újraértelmezhető formában tároljuk.
- Az adatok értelmezésekor:
- az adatbázisban rögzített adatokat kezeljük,
- velük műveleteket végzünk,
- illetve értelmezzük azokat,
- összefüggéseket igyekszünk feltárni,
- törekszünk megszerezni az adatok által hordozott információt.

Az adatbázis-kezelés kétpólusú tevékenység. Feladatait egyfelől a tudás adattá kódolása, az adatok informatikai eszközökkel, jól meghatározott szerkezetben, hatékonyan értelmezhető formában történő tárolása, másfelől a tárolt adatok rendezése, kezelése, és újraértelmezése alkotja.

7.2.2 Az adatbázisok felépítése

Az adatbázis-kezelésben munkánk hatékonyságát eredendően meghatározza az adatbázis szerkezete. Ez dönti el ugyanis, hogy miről, mit, és milyen összefüggésekben tárolunk el. E szerkezet megértéséhez az alábbi kérdésekre kell válaszokat keresnünk:

- Hogyan épül fel egy adatbázis szerkezete?
- Milyen szerkezeti elemek alkotják?
- Mitől lehet ez a szerkezet jobb, vagy rosszabb?

A következőkben az adatbázist alkotó szerkezeti elemeket ismerjük meg. Arról beszélünk, miket is tárolunk el egy adatbázisban, mit jelent az egyed, egyedtípus, tulajdonság, tulajdonságtípus, kapcsolat és kapcsolattípus.

7.2.3 Egyed és tulajdonság

Az adat meghatározásában található „objektum” kifejezést, az adatbázis-kezelésben az egyed szóval helyettesítjük. Az adatbázisokban pedig adatokat, azaz egyedek tulajdonságait tároljuk.

7.2.4 Tulajdonságtípus

Amikor adatbázist készítünk, általában sok-sok egyed számos különböző tulajdonságát tároljuk el. Az egyedek tulajdonságait csoportosíthatjuk, tipizálhatjuk aszerint, hogy milyen jellemzőt írnak le. Az azonos jellemzőket meghatározó tulajdonságok halmazát tehát tulajdonságtípusnak nevezzük. A tulajdonságtípusokat elnevezzük, ezzel a névvel utalunk arra, hogy milyen jellemzőt írnak le a tulajdonságtípusba tartozó tulajdonságok. A név, a foglalkozás, a születési idő, személyiigazolvány-szám mind tulajdonságtípusok.

7.2.5 Egyedtípus

Nemcsak a tulajdonságok, de az egyedek is halmazokba sorolhatók, mégpedig aszerint, hogy milyen tulajdonságtípusok jellemzik őket. Azok az egyedek, amelyek mindegyike rendelkezik névvel, foglalkozással, születési idővel, és személyiigazolvány-számmal, ugyanabba az egyedhalmazba, más néven egyedtípusba tartoznak. Az azonos tulajdonságtípusokkal jellemezhető egyedek halmazát egyedtípusnak nevezzük.

7.2.6 Kapcsolatok

Egy adatbázisban általában több egyedtípust is tárolunk. Az adatbázis egyedtípusai nem függetlenek egymástól, közöttük általában valamilyen viszony, kapcsolat van. Az egyedtípusok közötti kapcsolatokból szintén nagyon hasznos és fontos információk nyerhetők. Az egyedtípusok közötti viszonyt kapcsolatnak nevezzük. Egy kapcsolat mindig két egyedtípus közötti viszonyt ír le. Az adatbázisokban egyedtípusokba csoportosított egyedek tulajdonságtípusokba sorolt tulajdonságait, valamint az adatbázis egyedtípusai közötti kapcsolatokat tároljuk.

7.2.7 A kapcsolatok tipizálása, kapcsolattípusok

Az egyedekhez és tulajdonságokhoz hasonlóan, a kapcsolatokat tipizálhatjuk, csoportosíthatjuk. Két egyedtípus ('A' és 'B') kapcsolatát aszerint sorolhatjuk típusokba, hogy az 'A' egyedtípus egyedei a 'B' egyedtípus hány egyedével lehetnek kapcsolatban, és a 'B' egyedtípus egyes egyedei az 'A' egyedtípusban hány egyedhez kapcsolódhatnak.

- Egy az egyhez kapcsolat (1:1)

Az egy az egyhez kapcsolat esetén az 'A' egyedtípus bármelyik egyede a 'B' egyedtípus egyetlen egyedéhez kapcsolódhat, és a 'B' egyedtípus bármely egyede is csak egyedhez kapcsolódhat az 'A' egyedtípusban. A kapcsolat jelölése: 1:1.

- Egy a többhöz (1:N) kapcsolat

Az egy a többhöz kapcsolat esetén az 'A' egyedtípus bármely egyede a 'B' egyedtípus több egyedéhez is kapcsolódhat, azonban a 'B' egyedtípus egyedei az 'A' egyedtípusban csak egy egyedhez kapcsolódhatnak. Jelölése: 1:N.

- Több a többhöz kapcsolat

A több a többhöz kapcsolatban az 'A' egyedtípus egyedei a 'B' egyedtípus több egyedéhez kapcsolódhatnak, és ez fordítva is igaz. A 'B' egyedtípus egyedei az 'A' egyedtípus több egyedéhez kapcsolódhatnak.

7.2.8 Az adatmodell

Amikor adatbázist készítünk, feladatunk valójában nem más, minthogy az egyedeket megfelelő egyedtípusokba, tulajdonságaikat pedig tulajdonságtípusokba sorolva tároljuk. Ezt követően valamilyen módon rögzítenünk kell az egyedtípusok közötti kapcsolatokat is. Mindezt természetesen úgy kell elvégez-

nünk, hogy a keletkezett adathalmaz a lehető legkisebb helyet foglalja el, ugyanakkor gyorsan, egyszerűen és hatékonyan legyen kezelhető.

Az egyedek, egyedtípusok, tulajdonságok, tulajdonságtípusok, kapcsolatok, kapcsolattípusok tárolásának módját az adatmodell határozza meg. Különféle adatmodellek állnak rendelkezésre, és mindegyik más-más lehetőségeket kínál az adatok tárolására.

Az adatmodell egy szabályrendszer, amely egy adatszerkezet felépítésének elveit határozza meg. Pontosan rögzíti, hogyan tárolhatók az egyedek, egyedtípusok, tulajdonságok és tulajdonságtípusok, valamint az egyedtípusok közötti kapcsolatok.

Az alkalmazott adatmodellen múlik az is, hogy milyen műveletek lesznek elvégezhetők az adatbázisban tárolt adatokkal.

7.2.9 A relációs adatmodell

Az adatbázis-kezelésben többféle adatmodellt ismerünk. Ilyen a hierarchikus, a hálós, az objektumorientált és a relációs adatmodell. Tananyagunkban a relációs adatmodellt mutatjuk be. Ez a legegyszerűbb, legkönnyebben kezelhető, éppen ezért legelterjedtebb adatmodell. Tanulmányozásakor talán legfontosabb, hogy az egyedtípusok tárolására használt struktúra, a tábla szerkezetét megértsük!

Tábla, mező, rekord, mezőérték

A relációs adatmodellben egy egyedtípus minden adata egy egyszerű táblában tárolódik. A tábla egy mátrix, amelyet sorok és oszlopok alkotnak. Minden oszlop egy tulajdonságtípusnak felel meg. A sorokban az egyedek helyezkednek el, a sorok celláiban pedig a tulajdonságok kapnak helyet.

A relációs adatmodellben az egyedtípus helyett a tábla megnevezést használjuk. A tulajdonságtípus fogalomnak a mező felel meg. Az egyedeket rekordoknak nevezzük. A tulajdonságot mezőértéknek hívjuk.

A relációs adatmodell szabályai

A relációs adatmodell tábláinak felépítését az alábbi szabályok határozzák meg:

- Egy táblán belül nem lehet két teljesen egyforma rekord. Azaz táblán belül minden rekordnak – valamiben – különböznie kell a többitől.
- Minden rekordban azonosnak kell lennie a mezők sorrendjének.
- Minden rekordban azonos számú mezőnek kell lennie.
- A táblában nem lehetnek többértékű mezők.

*Rekordok különbözősége: a*zt jelenti, hogy mindenképpen kell, hogy legyen olyan mező (esetleg mezőkombináció), amelynek értéke minden rekordban egyedi.

Mezők sorrendje: a mezők sorrendjének azonosnak kell lennie.

*Mezők száma: e*z a szabály nem engedi meg, hogy az egyes rekordokban több, míg másokban kevesebb mezőt hozzunk létre.

*Többértékű mezők: a*z olyan mezőket, amelyek az egyes rekordokban több értéket is tartalmazhatnak, többértékű mezőknek nevezzük.

A relációs adatmodell nem engedi meg a többértékű mezők használatát! Ez azt jelenti, hogy a személyek tábla sajnos az előbbi és az utóbbi formában sem használható.

7.2.10 Speciális mezők a relációs adatmodellben

Bár a relációs adatmodell szabályai nem szólnak részletesen a speciális mezőkről, az azonosító, az összetett kulcs, a többértékű mező, az ismétlődő értékű mező, és az idegen kulcs fogalmának megismerése a későbbi tananyag megértésének fontos feltétele.

Azonosító vagy kulcs

A relációs adatmodell szerint készült táblákban kell, hogy legyen olyan mező, amelynek értéke minden rekordban más és más, két rekordban sohasem egyforma, így alkalmas arra, hogy a rekordokat egyértelműen megkülönböztesse egymástól.

Az azonosító, vagy más néven kulcs egy egyedtípus olyan tulajdonságtípusa, amelynek értékei egyértelműen megkülönböztetik egymástól az egyedtípus egyedeit. Adatbázisok készítésekor általában kerüljük az egyedtípus valódi tulajdonságtípusainak azonosítóként történő használatát. Gyakoribb, hogy az egyedtípushoz hozzárendelünk egy új mezőt, amelyben minden rekord egy sorszámot kap. Ez a sorszám különbözteti majd meg az egyedtípus rekordjait. Az adatbázis kezelőjének kell gondoskodni arról, hogy ez a sorszám valóban különböző legyen az egyes rekordok esetén.

Előfordulhat, hogy egy táblában több olyan mező is van, amely alkalmas az azonosító szerepének betöltésére. Az ilyen mezőket alternáló kulcsoknak vagy kulcsjelölteknek nevezik. Azt a mezőt, amelyet kiválasztunk és kulcsként használunk, elsődleges kulcsnak is hívják.

*Összetett kulcs: e*lőfordulhat, hogy olyan táblát hozunk létre, amelyben egyetlen olyan mező sincsen, amelyet azonosítóként használhatnánk. Ilyen

esetben kaphat szerepet az összetett kulcs. Az összetett kulcs egy tábla kettő vagy több mezőjéből alkotott mezőkombináció, amelyet a rekordok azonosítására használunk. Az összetett kulcsot tartalmazó tábla rekordjaiban az összetett kulcs minden elemében kell legyen érték!

*Idegen kulcs: a*z adatbázisokban nem csupán az egyedtípusokat, hanem azok közti kapcsolatokat is el kell tárolnunk. A táblák közötti kapcsolatok tárolására használjuk az idegen kulcsokat. Idegen kulcsnak nevezzük egy tábla olyan mezőjét, amelynek értékei egy másik tábla rekordjait azonosítják. Az idegen kulcs segítségével a táblák közötti kapcsolatokat tudjuk eltárolni a relációs adatmodellnek megfelelő adatbázisokban.

A kapcsolatok ilyen tárolása miatt a relációs adatmodellben két tábla között közvetlenül csak 1:1, vagy 1:N kapcsolat lehetséges. Az N:M kapcsolat kialakításához kisebb „trükkre” lesz szükség.

*Többértékű mező: e*gyáltalán nem ritka jelenség, hogy egy egyedtípus valamelyik tulajdonságtípusa több értéket is felvehetne az egyes egyedek esetében. A többértékű mező egy egyedtípus olyan tulajdonságtípusa, amely az egyes rekordokban több értéket is tartalmazhat. Nagyon fontos tudnunk, hogy a relációs adatmodell nem engedi meg a többértékű mezők használatát. A relációs adatmodellnek adatbázisok kialakításakor mindenképpen meg kell szűntetnünk a többértékű mezőket!

Ismétlődő értékű mező: a többértékű mezővel szemben az ismétlődő értékű mezőt nem tiltja a relációs adatmodell. A későbbiekben azonban látni fogjuk, hogy az ilyen mezők fölöslegesen megnövelik az adatbázis mértet, és általában kezelési nehézségeket is okoznak. Egy mező akkor ismétlődő értékű, ha ugyanaz az a mezőérték több rekordban is előfordul.

7.2.11 Az adatbázis

Eddig többször használtunk már az adatmodell és az adatbázis fogalmat. Előbbi – mint tudjuk – egy adathalmaz tárolásának elvi szerkezetét adja meg. Az adatbázis az adatmodell szabályainak megfelelő tényleges adathalmaz. Ha úgy tetszik, az adatmodell a szabály, az adatbázis a szabály megvalósítása.

Az adatbázis olyan ténylegesen tárolt adathalmaz, amelyet egy adatmodell szabályainak megfelelően alakítottak ki. Az adatbázisban egyedtípusokba sorolt egyedek tulajdonságtípusokba sorolt tulajdonságait, valamint az egyedtípusok egyedei közötti kapcsolatokat tároljuk.

Minden adatbázis valamilyen adatmodell alkalmazása. Éppen ezért az adatbázisok jellemezőjeként szokás megemlíteni, hogy milyen adatmodell alap-

ján készültek. Az általunk készített adatbázisok mind a relációs adatmodell alapján készülnek majd, tehát relációs adatbázisok lesznek.

A hálós, a hierarchikus, illetve az objektumorientált adatmodelleket ritkábban használják. Alkalmazásukkal hierarchikus adatbázisok, hálós adatbázisok, illetve objektumorientált adatbázisok készíthetők.

7.2.12 Az adatbázisrendszerek tervezési lépései

Az adatbázisrendszerek tervezésénél a korábban már tárgyalt általános szoftverfejlesztési irányelvekből érdemes kiindulni. A tervezés legfontosabb lépései ennek megfelelően:

- követelményanalízis: a vizsgált problématerület elemzése, a megoldandó feladatok, a követelmények meghatározása;
- rendszerelemzés: a problématerület modellezése, a belső struktúrák és a működés feltárása több, különböző szemszögből és különböző részletességgel;
- rendszertervezés: az elkészítendő szoftver belső struktúrájának, működésének több, különböző szemszögből történő feltárása, különböző részletesség mellett;
- kódolás: az elkészült modell leírás átkonvertálása a számítógép által érthető formára, valamilyen programozási nyelvet felhasználva;
- tesztelés: az elkészült kód hibamentességének ellenőrzése;
- karbantartás: folyamatos ellenőrzés, módosítások, hibakijavítások sorozata.

Természetesen itt nem szabad elfelejteni, hogy a tervezés folyamata nem egy egyszerű szekvencia, mivel bizonyos elemek többször is megismétlődnek, így a tervezés ciklikus folyamattá alakul át, és a fenti felsorolás ennek a ciklusnak az elemeit sorolja fel egy kvázilogikai sorrendben. Ha az egyes tevékenységek absztrakciós szintjeit vizsgáljuk, akkor látható, hogy a piramisban fentről lefelé haladva egyre csökken a leírásmód elvontsága és egyre nagyobb szerepet kap a konkrétabb megfogalmazás.

A szoftvertermék leírása a fejlesztés során fokozatosan alakul át az absztrakt, esetleg pármondatos köznapi leírásból a futtatható több ezer soros gépikód-sorozatig. Ez az átalakulás a modellek sorozatán keresztül valósul meg. Előbb absztraktabb, később konkrétabb, részletekkel gazdagított modellek jelennek meg. A DBS rendszer sem tér el ettől a fejlesztési metodikától, sajátossága, egyedisége leginkább a felhasznált modellekben rejlik. A DBS rendszerek jellegzetessége, hogy kiemelt helyre, elsődlegesen az adatrendszerre koncent-

rál. Különösen fontos tevékenység az adatbázis megtervezése, hiszen a DBS központjában az adatbázis áll, és ennek hatékonysága, korrektsége az összes alkalmazás teljesítményére kihat. Az adatbázis tervezése kettős célt követ: egyrészt ki kell elégíteni a felhasználók információigényét, másrészt ügyelni kell az információfeldolgozás hatékonyságára is.

Az adatbázis tervezésekor is több modellen keresztül jutunk el a fizikai adatbázishoz, hiszen induláskor rendszerint csak igen informális ismereteink vannak a feladatról, a modellezett világról. Ezzel szemben az elkészített adatbázisrendszer igen szigorú, formális nyelvvel írható le. Több modell létezik, melyek különböző absztrakciós szinteken írják le a megvalósítandó adatbázist. Mivel ezek mindegyike az adatbázis leírására szolgál, mindegyikre használható az adatmodell kifejezés.

Az adatmodelleket alapvetően két nagy csoportba szokás osztályozni. Az egyik adatmodell-típus a szemantikai adatmodell (SDM) vagy koncepcionális (conceptual) adatmodell, mely elvontabb szinten, részletek nélkül, emberközelien írja le az adatszerkezetet. Az SDM-ek alkalmasak az adatbázis lényegének a kiemelésére, a szerkezet megértésére. Egy adott adatbázis SDM-e ugyanaz marad akkor is, ha esetleg módosul a kiválasztott DBMS, hiszen a modellezett valóság is változatlan marad, és az SDM valóságközeli modell-leírás.

A másik csoport a konkrétabb, DBMS közeli adatmodellek köre, melyekből már megemlítettük a relációs, a hálós, a hierarchikus és az OO-modelleket. Ezek a modellek rugalmasságuk ellenére mégiscsak egyfajta korlátozást, keretet szabnak, melybe bele kell gyömöszölni a valóságot. A valóság tökéletes leírásához azonban a meglévőnél sokkal gazdagabb lehetőségekre lenne szükség, mint amit a hagyományos DBMS adatmodellek támogatnak. Mivel ezek a modellek túlságosan távol vannak a modellezett valóság közvetlen leírásától, ezért rendszerint felhasználják a szemantikai adatmodelleket is a tervezésnél, első lépcsőként, és az elkészült SDM-et konvertálják át DBMS-adatmodellre.

A modellek szerepének kiemelésével újrafogalmazhatjuk az adatbázis tervezésének főbb lépéseit:

- Igényfelmérés és analízis.
- Koncepcionális adatbázismodell elkészítése.
- DBMS-rendszer kiválasztása.
- A fogalmi modell átkonvertálása adatbázis adatmodellre.
- A fizikai adatmodell tervezése.
- Adatbázis implementálása.

7.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

7.3.1 Összefoglalás

A hetedik lecke célja az volt, hogy megismertessük a hallgatókat az adatbázis-tervezés legfontosabb lépéseivel. A lecke során beszéltünk az adatbázis-kezelés alapjairól, melynek kapcsán többek között szóba került az adatbázisok felépítése, a tulajdonságtípusok, a kapcsolatok és a kapcsolatok tipizálása illetve a kapcsolattípusok. A lecke második részében szót ejtettünk az adatmodellek jellemzőiről, közelebbről megvizsgáltuk a relációs adatmodell tulajdonságait. A lecke végén pedig az adatbázisrendszerek tervezési lépéseiről ejtünk szót.

7.3.2 Önellenőrző kérdések

1. Ismertesse az adatbázis-kezelés alapjait!
2. Ismertesse az adatbázisok felépítését!
3. Ismertesse a tulajdonságtípusokat!
4. Ismertesse az egyedtípusokat!
5. Mutassa be a kapcsolatok tipizálását, kapcsolattípusokat!
6. Mit tud az adatmodellekről?
7. Mutassa be a relációs adatmodellt!

8. LECKE: ADATBEVITEL ÉS ŰRLAPTERVEZÉS

8.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

Az előző fejezetben tárgyalt adatbázisokat adatokkal kell feltöltenünk, hogy be tudják tölteni funkciójukat. Az adatbázisok adatai többféle forrásból származhatnak: bizonyos esetekben az adatok egy csoportja már rendelkezésre áll, vagy automatikusan kinyerhető elektronikus formában a rendelkezésre álló adattárolókból, de számos esetben az információ hordozói emberek, akiknek az adattároló „eszközeihez" nem férünk hozzá közvetlenül. Ebben az esetben megfelelően konstruált kérdéseket kell feltennünk, hogy a kívánt adatokhoz és információhoz hozzáférhessünk. Ennek a folyamatnak a jól bevált eszközei az űrlapok. A nyolcadik lecke célja, hogy megismertesse a hallgatókat az űrlapokhoz kapcsolódó alapfogalmakkal, így lesz szó az űrlapok jellemzőiről (azonosíthatóság, kitöltést segítő információk, áttekinthető felületek, egyértelmű tagolás és definiálás, egységes forma, következetesség, pontosság, a redundáns adatbekérés elkerülése és feldolgozhatóság), a lecke végén pedig szót ejtünk az űrlapokon alkalmazott adattípusokról.

8.2 TANANYAG

- Adatbázisok feltöltése
- Az űrlapokkal kapcsolatos alapfogalmak
- Az űrlapok jellemzői
- Azonosíthatóság
- A kitöltést segítő információk
- Áttekinthető felületek
- Egyértelmű tagolás és definiálás
- Egységes forma
- Következetesség, pontosság
- A redundáns adatbekérés elkerülése
- Feldolgozhatóság
- Az űrlapokon alkalmazott adattípusok

8.2.1 Adatbázisok feltöltése

Az előző fejezetben tárgyalt adatbázisokat adatokkal kell feltöltenünk, hogy be tudják tölteni funkciójukat. Az adatbázisok adatai többféle forrásból származhatnak: bizonyos esetekben az adatok egy csoportja már rendelkezésre áll, vagy automatikusan kinyerhető elektronikus formában a rendelkezésre álló adattárolókból, de számos esetben az információ hordozói emberek, akiknek az adattároló „eszközeihez" nem férünk hozzá közvetlenül. Ebben az esetben megfelelően konstruált kérdéseket kell feltennünk, hogy a kívánt adatokhoz és információhoz hozzáférhessünk. Ennek a folyamatnak a jól bevált eszközei az űrlapok.

8.2.2 Az űrlapokkal kapcsolatos alapfogalmak

Az űrlapokkal kapcsolatos fogalmak néhány eleme (adat, adatmező stb.) mát korábban is szerepelt a tananyagban, de ebben a kontextusban némileg eltér az értelmezés.

Az űrlap: űrlapnak nevezzük azokat a papíralapú vagy dokumentumokat, amelyek előre meghatározott adatok összegyűjtésére, illetve közlésére szolgálnak.

Adat: az adat az előre meghatározott kérdésre az adatszolgáltató által adott válasz.

Adatmező: az adatmező egy előre megformázott terület, amely az adat rögzítésére szolgál.

Karakter: a karakter a nyelvi sajátosságoktól függetlenül bármely betű, számjegy, írásjel, illetve az azokat elválasztó szóköz.

Kibocsátó: a kibocsátó az űrlap tartalmi és formai tervezését ellátó szervezet.

Befogadó: az űrlapokon megjelenített adatokat feldolgozó szervezet.

Adatszolgáltató: az adatokat szolgáltató személy.

Elektronikus űrlapok

A nyomtatott űrlapok elektronikus, vagy hibrid változatai. Típusai:

- Elektronikusan tárolt és továbbított (letölthető), de alapvetően papíralapúra tervezett űrlap (kitöltés és továbbítás)
- Elektronikusan tárolt és továbbított (letölthető), elektronikusan kitölthető, de papíralapúan továbbítható

- Elektronikusan tárolt és továbbított (letölthető, vagy online kitölthető) és elektronikusan kitölthető és továbbítható űrlap
- Elektronikusan tárolt és online kitölthető űrlap.

Ezen belül a feldolgozás szempontjából két alcsoportot különböztethetünk meg:

- a bevitt adatok feldolgozásához emberi közreműködésre van szükség
- a bevitt adatok közvetlenül alkalmasak gépi értelmezésre

8.2.3 Az űrlapok jellemzői

Az űrlapok létrehozásánál célszerű szem előtt tartani néhány elvet, amely az űrlapok hatékonyságát növelheti:

- Azonosíthatóság
- A kitöltést segítő információk
- Áttekinthető felületek
- Egyértelmű tagolás és definiálás
- Egységes forma követése, betartása
- Következetesség, pontosság
- A redundáns adatbekérés elkerülése
- Feldolgozhatóság

31. ábra: Űrlapok jellemzői

8.2.4 Azonosíthatóság

Az űrlapot a kibocsátónak, a kitöltőnek és a befogadónak egyaránt könnyen kell tudnia azonosítani. Ennek alapvető feltétele az űrlap beazonosíthatóságának egységes formája a következő információk megjelenítése az űrlap első oldalának fejrészében:

Azonosító karaktersor – az űrlap nyilvántartási kódja

Kibocsátó megnevezése – intézmény (esetleg az azon belüli szervezeti egység)

Az űrlap megnevezése – közérthető megnevezés

Hatályosság kezdete – mikortól alkalmazandó az űrlap

8.2.5 A kitöltést segítő információk

Ahhoz, hogy egy adatszolgáltatás a lehető leghatékonyabban történhessen és a kitöltő adatszolgáltatását megkönnyítsük, a kitöltőt már az űrlap első oldalán tájékoztatni kell a szükséges információkról, információhordozókról, így a kitöltés alatt nem kell a figyelmét kereséssel elterelnie.

8.2.6 Áttekinthető felületek

A kitöltőre az űrlap felülete, megjelenése által gyakorolt első benyomás önmagában is befolyásolhatja a hibalehetőségeket. Amennyiben egy áttekinthető (könnyen felismerhető, könnyen értelmezhető) felülettel találkozik a kitöltő, akkor a figyelme jobban összpontosulhat a kitöltésre, míg egy első látásra bonyolultnak, áttekinthetetlennek tűnő űrlap fokozza a hibák előfordulásának esélyét. A felületeket úgy kell kialakítani, hogy azok vezessék a szemet, informáljanak, és ne ijesszék meg a kitöltőt.

8.2.7 Egyértelmű tagolás és definiálás

A kitöltést könnyíti, egyben a hibalehetőségeket is csökkenti, ha a különféle – logikai, tartalmi egységet képező – adatokra vonatkozó mezőket az űrlap kitöltőjének tagoltan kell kitöltenie.

Az űrlap felületét úgy kell tagolni, hogy egyértelmű és az adatok tartalmából következően logikailag összefüggő csoportokba rendezze a kérdéseket, illetve adatokat.

Az adatok definícióit szintén egységes, logikus módon kell megadni lehetőleg úgy, hogy az ne szoruljon további magyarázatra. Vannak olyan általános

használt definíciók, amelyeket feltételek nélkül kell alkalmazni, mivel azok például okiratokban is úgy szerepelnek.

8.2.8 Egységes forma

Az egységes forma az adatszolgáltatásban mind az űrlap szerkesztője, mind az adatszolgáltató, mind pedig a befogadó és a feldolgozó számára az adatok áttekinthetőségét könnyíti meg, ezért az önmagában is növeli az adatfeldolgozás hatékonyságát.

Az egységes forma megvalósításához kérdéscsoportokba rendezett, azonos módon összeállított adatcsoportokat kell alkalmazni olyan témákban, amelyek több űrlapon azonos, vagy közel azonos adatokat kezelnek.

8.2.9 Következetesség, pontosság

A következetesség nem csupán egy adott űrlapon belül elvárás. Minden űrlapon azonos elnevezéseket, azonos csoportokat kell használni az azonos adatokra és adatcsoportokra, az ezekre vonatkozó szabályoktól egy űrlapszerkesztő sem térhet el. A következetes felépítés lehetővé teszi, hogy az egyedi megoldásokat igénylő, valamint az új űrlapok is illeszkedjenek a rendszerbe, megtartva annak áttekinthetőségét és egyértelműségét.

E követelménynek eleget tesz például egy úgynevezett XML-sémadefiníció alkalmazása, amely megszabja az űrlapokon szereplő adatmezők jellemzőit.

8.2.10 A redundáns adatbekérés elkerülése

A jó űrlap egy adatot csak egyszer kér be akkor is, ha egy másik kontextusban ismételten szükség van rá.

Elektronikus űrlapkitöltő felületeknél megvalósítható, hogy amennyiben egy adatot már megadtunk, úgy azt a rendszer automatikusan kitöltse egy ismételt bekérés esetében. Ennek egy kifinomult változata, amikor a kitöltő a gyakran ismételt adatokat (pl. az azonosítás adatait) a szükséges adatbiztonsági szempontok betartása mellett eltárolhatja. Ezeket az adatokat ettől kezdve nem szükséges manuálisan kitölteni, hanem elegendő megmutatni annak tárolási helyét és a rendszer annak tartalmából minden létező, szükséges adatot átemel az aktuális űrlap megfelelő helyeire.

8.2.11 Feldolgozhatóság

Két fontos tény is alátámasztja az eddig leírtak fontosságát, amely tények az űrlapok feldolgozásakor kerülnek előtérbe, és amelyek az egységesség fontosságát emelik ki.

Az azonos adatok és adatcsoportok egységes megjelenítése és használata a feldolgozást nagyban megkönnyíti, mivel a feldolgozók számára is könnyebbséget jelent az azonos felületek áttekintése, lényegesen hamarabb és pontosabban deríthetőek fel kitöltési hiányosságok és hibák. Az egységesség ezen túlmenően megteremti az automatizmusok kiépítésének lehetőségét.

A másik tény az űrlapok elektronikus űrlapkitöltő felületekre történő adaptálásakor kerül előtérbe. A korábban már említett okok miatt az űrlapok belátható időn belül három formában lesznek jelen hétköznapjainkban:

- Nyomtatott űrlap
- Elektronizált űrlap
- Elektronikus adatszolgáltató felület

Ez azt is jelenti, hogy a befogadóhoz mindhárom adatszolgáltatás eredménye vegyesen jut el. Amennyiben mindhárom forma minden tekintetben azonos, úgy a feldolgozás egységessé tehető.

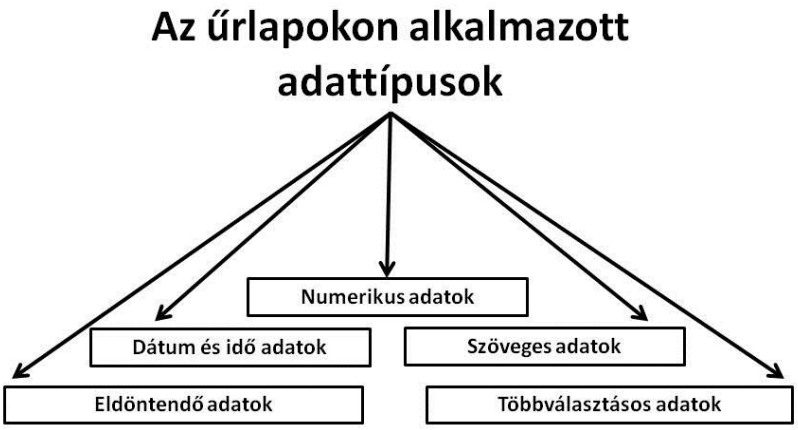
8.2.12 Az űrlapokon alkalmazott adattípusok

Az űrlaptervezéshez ismerni kell az űrlapokon megjeleníthető általános adattípusokat és azok alkalmazási lehetőségeit, valamint korlátait. Az adattípusok meghatározása eltér az adatbázisoknál megszokottaktól, aminek az az oka, hogy az űrlapok tervezői rendszerint nem informatikai szakemberek.

- Numerikus adatok: csak számokat, aritmetikai és elválasztó jeleket tartalmazhatnak; (pénzösszeg, mennyiség, irányítószám, stb.).
- Dátum és idő adatok: karakteres adatok – azok az adatok, amelyek hossza behatárolható és bármilyen írásjelet, valamint szóközt tartalmazhatnak (pl. vezetéknév, keresztnév, település, stb.).
- Szöveges adatok: a karakteres adatok kiterjesztett változata, amelynél a hossz szöveges leírásokat tesznek lehetővé (pl. indoklások, előre nem definiálható adatok).
- Eldöntendő adat: egyválasztásos kijelölés, kettő vagy több lehetséges válasz közül; ennek altípusa az igen-nem adat, amelynek lehetséges értékei az igaz vagy a hamis, továbbá az osztályozó adatok, amelyek egy

skála lehetséges értékei közül egyet engednek kiválasztani (a kitöltő neme, családi állapota, stb.)

- Többválasztásos adat: egy adott felsorolásból több válasz is megjelölhető; (vállalkozás tevékenységei, beszélt nyelvek, név előtagok)
- Nem adattípus, de az adatok fontos jellemzője a kötött formátum. Egyes adatok tartalma szükségképpen meghatározza azok összetételét és tagolását.



32. ábra: Űrlapokon alkalmazott adattípusok

8.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

8.3.1 Összefoglalás

Ahogyan a bevezetőben már szót ejtettünk róla, az előző fejezetben tárgyalt adatbázisokat adatokkal kell feltöltenünk, hogy be tudják tölteni funkciójukat, amelyre az egyik leggyakrabban használt eszköz az űrlap. A nyolcadik lecke célja az volt, hogy megismertesse a hallgatókat az űrlapokhoz kapcsolódó alapfogalmakkal, így esett szó az űrlapok jellemzőiről (azonosíthatóság, kitöltést segítő információk, áttekinthető felületek, egyértelmű tagolás és definiálás, egységes forma, következetesség, pontosság, a redundáns adatbekérés elkerülése és feldolgozhatóság), a lecke végén pedig szót ejtettünk az űrlapokon alkalmazott adattípusokról.

8.3.2 Önellenőrző kérdések

- Sorolja fel az űrlapok jellemzőit!

1. Mit jelent az űrlapok jellemzői között az azonosíthatóság?
2. Mit jelent az űrlapok jellemzői között a kitöltést segítő információk megadása?
3. Mit jelent az űrlapok jellemzői között az áttekinthető felületek alkalmazása?
4. Mit jelent az űrlapok jellemzői között az egyértelmű tagolásra és definiálásra törekvés?
5. Mit jelent az űrlapok jellemzői között az egységes forma?
6. Mit jelent az űrlapok jellemzői között a következetesség és a pontosság?
7. Mit jelent az űrlapok jellemzői között a feldolgozhatóság?

9. LECKE: INTERFÉSZ- ÉS DIALÓGUSTERVEZÉS

9.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A kilencedik fejezet célja, hogy megismertessük a hallgatókat a felhasználói interfész tervezésének alapelveivel. Ennek során beszélünk a felhasználói felület egységességéről, szót ejtünk a felhasználói kompetenciákról, felhasználói felület egységességéről, és megvizsgálunk olyan, az interfésztervezés szempontjából fontos fogalmakat, mint a konzisztens működési sémák kialakítása és az adekvát információmegjelenítés. A lecke második felében beszélünk a felhasználók támogatásáról és a színhasználat kérdéseiről.

9.2 TANANYAG

- A felhasználói felület tervezése
- A felhasználói felület egységessége
- Vegyük figyelembe a felhasználó kompetenciáit
- Konzisztens működési sémák kialakítására
- A véletlen felhasználói hibák minimalizálása
- A felhasználó megfelelő informálása
- A felhasználók széles körének támogatása
- Az adekvát információmegjelenítés
- Az adekvát színhasználat

9.2.1 A felhasználó felület

Az információs rendszerek felhasználói a felhasználó felület, más kifejezéssel a felhasználói interfészen keresztül kommunikálnak a rendszerrel. A felhasználó interfész grafikus felülete „eltakarja” a rendszer működésének összetett algoritmusait a felhasználók szeme elől. Éppen ezért a felhasználó felületek megfelelő megtervezése kritikus fontosságú a rendszer üzemletetése szempontjából, hiszen bármennyire is kiválóan működő rendszert tervezünk és fejlesztünk ki, ha a felhasználók nem látják át a rendszer kezelőfelületét, nem tudják majd azt hatékonyan használni. A rendszerfejlesztő mérnökök gyakran elkövetik azt a hibát, hogy túlbecsülik a rendszerfelhasználók informatikai kompetenciáit: a vállalatok hétköznapjaiban rendszerint nem mérnökök, hanem

középszintű számítógép-felhasználói ismeretekkel rendelkező munkavállalók üzemeltetik a rendszereket. A bonyolult felépítésű, nehezen átlátható szerkezetű felhasználói felületek frusztrálják a felhasználót, aki a rendszer valós minőségi mutatóitól függetlenül negatív véleménnyel lesz a teljes rendszerről.

9.2.2 A felhasználói felület tervezése

Az informatikai rendszerek interfésztervezésénél rendszerint megkülönböztetünk hardver- és szoftvertervezési feladatokat. A hardverinterfészek tervezésének szakirodalma bőséges információforrásként szolgál a tervezőmérnökök számára, számtalan ergonómiával, egészséges munkakörnyezettel foglalkozó szakkönyvet találhatunk. A szoftverek felhasználói interfészének a szakirodalma sokkal szerényebb, a szoftvertervező mérnökök elsősorban a szoftverfejlesztésre klasszikus területeire koncentrálnak, a felhasználó felületek tervezése – érdemtelenül – sokkal kisebb megbecsülésnek örvend. A következőkben tekintsük át a szoftverinterfészek tervezésének legfontosabb szempontjait.

9.2.3 Törekedjünk a felhasználói felület egységességére

A felhasználói felület egységessége azt jelenti, hogy a rendszer kommunikációs felületének funkcionális elemei (adatok beviteli mezők, adatok bevitelnek jóváhagyása, a szoftver struktúrájában a szintek közötti mozgás, adatbeviteli mezőkből kilépés, parancsvégrehajtás kiadása/törlése stb.) a képernyőn mindig ugyanabba a pozícióba kerüljenek. Ezáltal a felhasználó könnyebben elsajátítja a rendszer kezelését és gyorsabban eligazodik a képernyőn megjelenő információk között.

9.2.4 Vegyük figyelembe a felhasználó kompetenciáit

A felhasználói felületek tervezésénél vegyük figyelembe, hogy felhasználók a használat során teljesen más kontextusban kerülnek kapcsolatba a rendszerrel, mint a tervezők. A rendszer tervezésénél használt elnevezések, változók, rendszerkomponensek nevei semmitmondóak a felhasználó számára. A tervezők szempontjából a fejlesztési szakaszban kétségtelenül könnyebben azonosíthatóak az egyes folyamatok, ha a rendszertervben szereplő elnevezéssel jelennek meg a képernyőn, de a felhasználó csak a saját munkaterületének a működését látja át, csak annak a zsargonját ismeri, ezért a fejlesztőknek ezt figyelembe véve kell a képernyőn szereplő elemeket elnevezniük.

9.2.5 Törekedni kell a konzisztens működési sémák kialakítására

A felhasználók, amikor még tapasztalatlanok a rendszer használatában, a rendszer funkcionalitásán kívül (lehessen arra használni, amire tervezték) még nincsenek elvárásaik a rendszer működésével kapcsolatban. A rendszeres használat során azonban egyre jobban megismerik a rendszert, kognitív térképet alakítanak ki magukban arról, hogy mit hol találnak meg és egy adott akcióra milyen választ kaphatnak rendszertől, ezáltal gyorsabban és hatékonyabban végzik a munkájukat. A megszerzett tapasztalatokat általánosítják és elvárják, hogy a rendszer minden szituációban a már megismert sémáknak megfelelően működjön, ellenkező esetben a felhasználó elbizonytalanodik és visszasüllyed a kezdő felhasználók szintjére. Éppen ezért törekedni kell arra, hogy a sematizálható működési elemek minden esetben nagyon hasonlóan épüljenek fel és fussanak le.

9.2.6 Törekvés a véletlen felhasználói hibák minimalizálására

Az információs rendszerek felhasználó felületének tervezésekor törekednünk kell arra, hogy a felhasználók által véletlenül elkövetett hibák lehetőségét minimálisra szorítsuk vissza. A felhasználók a rendszer használata közben biztosan hibákat fognak véteni, aminek az oka lehet a fáradtság, dekoncentráció stb. Az egyik leggyakoribb hiba az elgépelés (pl. adatbevitelnél), ezért a rendszerfejlesztőknek törekedniük kell arra, hogy az ilyen jellegű hibalehetőségeket csökkentsék (pl. adatok begépelésére alkalmas mező helyett legördülő listából kell kiválasztani a beviteli értékeket).

A listából választás azonban nem minden esetben megoldható (pl. széles értéktartomány, tizedes pontosságú értékek stb.) ezért azt is lehetővé kell tenni, hogy a felhasználók kijavítsák a hibás tevékenységet. Ennek az egyik módja az, ha a rendszer figyelmezteti a felhasználót a hibás bevitelre (a rendszer megjelöli a nem megfelelő értékű vagy formátumú adatokat) illetve a rendszer megerősítést kér a destruktív műveletek végrehajtása előtt.

9.2.7 A felhasználó megfelelő informálása

A jól megtervezett felhasználó felület optimális esetben olyan egyszerűen épül fel, hogy a használata magától értetődik. Azonban az összetett információs rendszerek sokrétű feladatokat látnak el, amelyeket rendszerint nem könnyű használni. Ráadásul felhasználóként változhat, hogy valaki mit ítél jól strukturált, könnyen átlátható felületnek és mit bonyolultnak. Éppen ezért a felhaszná-

ló felület elengedhetetlen részét képezik a súgók. A korábbi rendszereknél a súgó egy témakörök szerint felépített, illusztrált szöveges dokumentumot jelentett, amely minden szükséges információt tartalmazott az adott rendszerről. Ezek használata nem minden esetben könnyítette meg a felhasználók dolgát, mert az információ megtalálását gyakran hosszas keresgélés előzte meg.

A felhasználók szempontjából olykor hasznosabb a kevesebb információt tartalmazó, tömörebb súgó, amit a felhasználó gyorsabban elolvas. Különösen igaz ez akkor, ha ehhez az információhoz azonnal hozzáfér. Éppen ezért, a jól tervezett rendszerekben a hagyományos, minden részletre kiterjedő súgó mellett készítenek egy kontextusfüggő változatot, amelyet integrálnak a felhasználó interfészbe, lehetővé téve, hogy a felhasználó azonnali segítséget kapjon, ha szüksége lenne rá.

9.2.8 Törekvés a felhasználók széles körének támogatására

Az információs rendszereket sokféle felhasználó használhatja, optimális esetben ezek mindegyikét támogatnia kell a rendszernek, ami azt jelenti, hogy a felhasználó interfész többféle interakciót támogasson és megjelenése testreszabható legyen.

A kevésbé tapasztalt felhasználók rendszerint az egeret használják a navigációra (ezt közvetlen manipulációs interakciónak nevezzük, ilyen pl. az adatbeviteli mezők és a menüelemek között az egérrel történő navigálás) a rendszer felületén, ezért lehetőleg megfelelően nagy méretű, egymáshoz nem túl közeli interakciós felületelemeket tervezzünk. Ugyanakkor a tapasztalt felhasználók rendszerint nem használnak egeret, hanem előnyben részesítik a billentyűkódokat (általánosan alkalmazott pl. az adatbeviteli mezőkön előrelépésre a Tab billentyű) és ennek megfelelően a felületnek ezt a típusú felhasználói interakciót is támogatnia kell.

Egyre fontosabb szempont az informatikai rendszerek tervezésénél, hogy a megváltozott munkaképességű munkavállalók is használhassák a rendszert, ezért optimális esetben a felhasználó felület megjelenése testreszabható, hogy a gyengénlátó munkavállalók is használhassák.

9.2.9 Törekedjünk az adekvát információmegjelenítésre

Az információs rendszerek felhasználó felületén gyakran kell a felhasználó számára információt megjeleníteni. Ez történhet direkt módon (pl. a számok

megjelenítésével) vagy az információ vizualizálásával. Sok tényező befolyásolja, hogy mikor melyik megjelenítés módot használjuk:

- Numerikus értékek esetén felhasználóak a konkrét, pontos értékekre van szüksége egy adott jellemzőre vonatkoztatva, vagy több érték egymáshoz való viszonyához?
- Milyen gyorsan kell követni az értékek változását és tartoznak-e az adott jellemzőhöz kritikus értékek, amelyekről a felhasználót azonnal értesíteni kell és ebben az esetben a felhasználóak be kell-e avatkoznia a rendszer működésébe közvetlenül?
- Szöveges információ esetén mérlegelnünk kell, hogy statikus vagy dinamikus a megjelenítendő információ és ez utóbbi esetben mily gyors a változás sebessége? Szüksége van-e felhasználónak a korábbi szövegelemek megjelenítésére vagy azok törölhetőek?

9.2.10 Törekedjünk adekvát színhasználatra

Az információs rendszerek felhasználói felületének a tervezésénél nagy gondot kell fordítanunk a színek használatára. A színek szerepe az információ megértésének s feldolgozásának a segítésében kiemelten fontos. Az előző pontban már beszéltünk arról, hogy esetenként a felhasználót informálni kell arról, hogy az információs rendszerben megjelenített mennyiségek az elfogadható tartományból kilépnek és a kritikus tartományba lépnek át. Ennek az információnak a kiértékelése gyorsan kell, hogy megtörténjen és ezekben az esetekben nem is a pontos érték a lényeges, hanem a kritikus érték átlépésének a ténye. Ezekben az esetekben kihasználhatjuk a színekhez társított általános jelentéstartalmakat, azaz a normál értéktartományt jelölhetjük pl. zölddel, a kritikus értéktartomány elérését pedig pirossal, ami a konkrét értékektől függetlenül lehetővé teszi az azonnali beavatkozás szükségességének megítélését. Hasonló megfontolásokból a felhasználó által kezdeményezet, potenciálisan hibát okozó tevékenységekre (pl. hibás adatbevitel) is a piros színnel hívja fel a figyelmet a rendszer.

A színek szerepet kaphatnak az összetartozó értékek csoportosításában is. Számos esetben előfordul, hogy az értékeket táblázatos formában kell megjeleníteni. Ebben az esetben a felhasználónak nehézséget okozhat a – különösen nagy méretű táblázatok esetén - táblázat sorainak követése, amely megfelelő színhasználattal jelentősen csökkenthető. Abban az esetben is segíthetjük az összetartozó értékek beazonosítását színek segítségével, ha ezek az értékek a képernyő más-más területén kell, hogy megjelenjenek.

A színek megfelelő használata tehát növelheti a rendszer információmegjelenítésének a hatékonyságát. Érdemes azonban a színeket megfontoltan használnunk, hiszen a nagyon sok színt tartalmazó képernyő zavaró lehet. Bánjunk különösen óvatosan a színek villogtatásával, mert ez azon túl, hogy zavaró, az arra hajlamos felhasználókban akár rohamokat is előidézhet. A színeket csak az információ jelentésének kiemelésére, nyomatékosítására használjuk, mert pl. a színvak illetve színtévesztő felhasználók nem képesek a színekhez társított jelentések értelmezésére.

9.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

9.3.1 Összefoglalás

A kilencedik fejezet célja az volt, hogy megismertessük a hallgatókat a felhasználói interfész tervezésének alapelveivel. Ennek során beszéltünk a felhasználói felület egységességéről, szót ejtettünk a felhasználói kompetenciákról, felhasználói felület egységességéről és megvizsgáltunk olyan, az interfésztervezés szempontjából fontos fogalmakat, mint a konzisztens működési sémák kialakítása és az adekvát információmegjelenítés. A lecke második felében beszéltünk a felhasználók támogatásáról és a színhasználat kérdéseiről.

9.3.2 Önellenőrző kérdések

1. Ismertesse a felhasználói felületek tervezésének alapelveit!
2. Mit jelent a felhasználói felület egységessége?
3. Mit jelent a konzisztens működési séma?
4. Mi jellemzi az információmegjelenítést?
5. Mit tart fontosnak a színek használatával kapcsolatosan?

10. LECKE: RENDSZERIMPLEMENTÁLÁS ÉS TELEPÍTÉS

10.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A tízedik lecke feladata, hogy a hallgatókat megismertesse a rendszerimplementáció és a rendszertelepítés alapvető jellemzőivel. Ennek keretein belül szó lesz a gyors szoftverfejlesztés és az inkrementális fejlesztés előnyeiről és hátrányairól, majd a hibrid fejlesztési módszerekről. A lecke második részében a szoftvertesztelés lépéseit vesszük górcső alá, melynek során megvizsgáljuk a rendszertesztelést, az integrációs tesztelést, a funkcionális teszteket, a teljesítménytesztelést és a követelményalapú tesztelést.

10.2 TANANYAG

- Implementáció
- A gyors szoftverfejlesztés jellemzői
- Az inkrementális fejlesztés előnyei
- Az inkrementális fejlesztés hátrányai
- Hibrid fejlesztési módszerek
- Szoftvertesztelés
- Rendszertesztelés
- Integrációs tesztelés
- Az integráció típusai
- Funkcionális tesztek
- Teljesítménytesztelés
- A stressztesztelés funkciói
- Követelményalapú tesztelés

10.2.1 Implementáció

A rendszerimplementálás alapjait már ismertettük a 2. leckében, most egy kicsit jobban, a gyakorlat aspektusából közelítjük meg a problémát. Az informatikai eszközök lehetővé tették az üzleti élet globalizációját, amelynek köszönhetően megszűntek a térbeli korlátok a vállalatok között. Az üzleti folyamatok felgyorsultak és az ebben a környezetben csak az tud sikeres maradni, aki gyor-

san reagál a változásokra, kihasználja a piac nyújtotta lehetőségeket. Különösen ez az informatikai üzletágra. A szoftverek szállítása elektronikus úton történik, ezért a megrendelő szempontjából teljesen mindegy, hogy a néhány méterre, vagy több ezer kilométerre található annak a cégnek az irodája, amelyik a vállalt által igényelt szoftvert elkészíti.

A szoftverek minden információs rendszernek az alapját képezik, ezért a rendszerfejlesztés üzleti tendenciái szorosan összekapcsolódnak a szoftverfejlesztéssel. A gyors fejlesztési trend miatt nagyon fontos, hogy az egyre nagyobb számban igényelt szoftverek minél hamarabb elkészüljenek és átadásra kerüljenek. Ez a tendencia nem kedvez az alaposan átgondolt fejlesztési módszereknek, hiszen kis túlzással azt állíthatjuk, hogy a globalizált üzleti világ folyamatai olyan gyorsan változnak, hogy mire a szoftver specifikációja a hagyományos módon tervezve megszületne, már érvényét veszítené.

A fenti jelenségek miatt egyre inkább előtérbe kerül az ún. gyors szoftverfejlesztés. A gyors szoftverfejlesztési módszerek segítségével viszonylag rövid idő alatt lehet használható funkcionalitású szoftvereket előállítani. A gyors szoftverfejlesztési módszereknél a szoftverfejlesztés lépései rendszerint nem egymás után, lineárisan következnek, hanem gyakran párhuzamosan is folynak. A gyors szoftverfejlesztési módszerek nem egységesek, de számos jellemzőben hasonlóságot mutatnak.

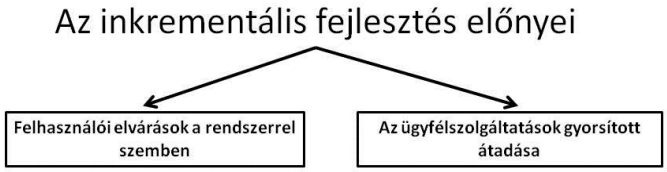
10.2.2 A gyors szoftverfejlesztés jellemzői

A gyors szoftverfejlesztés egyik legfontosabb jellemzője, hogy a rendszerfejlesztési folyamatok nem lineárisan futnak le, hanem a specifikáció, a tervezés és az implementálás párhuzamosan folyik. A rendszer specifikációját nem dolgozzák részletesen és a tervezési folyamat dokumentációja csak a legfontosabb adatokat tartalmazza. A rendszer tervezése kis lépésekben történik, a rendszer megrendelői és a rendszer fejlesztői nagyon szorosan együtt kell, hogy működjenek, hiszen nincsen részletesen kidolgozott specifikáció és rendszerterv, amit a rendszer megrendelői jóváhagytak volna.

10.2.3 Az inkrementális szoftverfejlesztés jellemzői

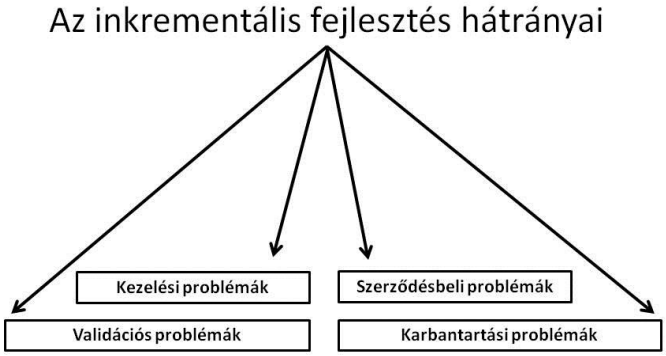
Ahogyan az előző részben már beszéltünk róla, az inkrementális fejlesztés egyik legfontosabb jellemzője, hogy a fejlesztés kis lépésekben történik, és a szoftver átadása sem egyetlen alkalom. A rendszer megrendelői nagy vonalakban vázolják, hogy milyen funkciókat várnak el a rendszertől, amit a fejlesztők igyekeznek a legfontosabb funkciókat minél hamarabb megvalósítani. A módszer előnye a gyorsaságon túl az, hogy a megrendelők látják a rendszer gyakorlati megvalósítását és a funkciók esetleges módosítását szinte azonnal kezde-

ményezni tudják. A rendszer átadása így a már implementált funkciók szerint történik. A rendszer felhasználóinak az átadás után azonnal lehetőségük nyílik az átadott rendszer véleményezésére és kérhetik az elvárásaiknak nem megfelelően működő funkciók átalakítását.



33. ábra: Az inkrementális fejlesztési mód előnyei

Ez a rendszerfejlesztési mód az előnyök mellet természetesen hátrányos tulajdonságokkal is rendelkezik. Az egyik hátrány, hogy azoknál a vállalatoknál ahol az ügymenetet konzervatív módon kezelik, nem tudnak mit kezdeni ezzel a fejlesztési szemléletmóddal, hiszen ezeknél a cégeknél a szokásos eljárás az, hogy bürokratikus szervezetek döntenek minden lépésről, és pontosan meghatározott szerződésekben szabályozzák az elvárásokat a fejlesztő céggel szemben, amibe rendszerint nem férnek bele az inkrementális fejlesztés bizonytalansági tényezői. A másik hátrány, hogy ezek a cégek nincsenek hozzászokva, hogy a fejlesztési folyamatban aktívan részt vegyenek. Ők rendszerint megrendelnek egy pontosan meghatározott szolgáltatást, és amikor elkészült, akkor átveszik. Ráadásul a szerződéskötéskor aláírt specifikációktól biztosan különbözne az átadott rendszer, ami szintén problémát jelentene a teljesítési igazolás kiadásánál.



34. ábra: Az inkrementális fejlesztés hátrányai

A hátrányok között kell említeni azt is, hogy az inkrementális szoftverfejlesztés dokumentációja rendszerint hiányos. Ennek az az oka, hogy a fejlesztési folyamat és a funkciók változása olyan gyors, hogy mire elkészülne a dokumentáció, már el is avulna. A szoftver minden változatához dokumentációt készíteni nagyon költséges és időigényes lenne.

A dokumentáció hiánya a szoftver karbantartását is nehézkessé teszi. Egy összetett rendszer működésének a megértése – ha valaki nem volt a fejlesztőcsoport tagja - akkor még jól dokumentált fejlesztési folyamat mellett sem egyszerű feladat, a dokumentáció hiánya nélkül pedig szinte lehetetlen.

10.2.4 Hibrid fejlesztési módszerek

Az inkrementális szoftvertesztelés nem alkalmazható minden esetben. Ez a szoftverfejlesztési eljárás az időhiány és a dokumentáció hiányosságai miatt megköveteli, hogy egy kézben legyen a fejlesztés. Éppen ezért azoknál a nagyméretű, összetett rendszereknél, ahol számtalan komponenst kell fejleszteni, ez a metódus nem jöhet számításba.

Abban az esetben sem fejleszthető inkrementálisan a rendszer, ha a kifejlesztendő rendszer egy nagyméretű rendszer egyik köztes alrendszere, amely adatokat kap és továbbít a többi alrendszer felé. A rendszerrel szemben támasztott követelmények ilyenkor annyira szigorúak, hogy az inkrementális fejlesztés nem vezetne eredményre.

Ezekben az esetekben alkalmazzuk a hibrid fejlesztési módszert. A hibrid fejlesztési folyamatot azért nevezzük hibridnek, mert elkészítjük a rendszer prototípusát, amit hagyományos módon fejlesztünk, ami segíthet abban, hogy megvizsgáljuk a rendszerspecifikáció és rendszerterv több változatát (ami az inkrementális rendszerfejlesztési metódusra jellemző). ez a folyamat segít abban, hogy megértsük, pontosan mit is vár el a rendszer megrendelője a rendszertől, és ez milyen viszonyban van a rendszerspecifikációval.

Fontos megjegyezni, hogy ebben az esetben a prototípus elkészítése nem arra szolgál, hogy annak a működését folyamatosan javítva végül átadjuk a megrendelőnek. A prototípus szerepe itt csak a rendszerterv és a rendszerspecifikáció tisztázása. Sejthető, hogy ez a folyamat azokra a specifikációs elemekre fókuszál, amelyek nem egyértelműek (pl. a rendszer megrendelője nem tudja egészen pontosan megfogalmazni, hogy mire is lenne szüksége), illetve a kifejlesztendő rendszerfunkció új, a fejlesztők által eddig nem implementált rendszerkomponens. A korábban már implementált rendszertulajdonságok általában nem is kerülnek bele a prototípusba.

Ez a szemlélet kihatással van a szoftverrel szemben támasztott minőségi követelményekre is. Mivel a prototípusnak nem kell majd valós körülmények között működnie, ezért a stabilitás és a megbízható működés szempontjából is jelentős engedményeket tehetünk.

Ez a megközelítés különbözik az inkrementális fejlesztési metódustól, hiszen ott az a cél, hogy a rendszer megrendelője a lehető leghamarabb megkapja a megrendelt rendszert, ami ugyan koránt sincs tökéletesen kész állapotban (de azért stabilnak és megbízhatóak kell lennie). Ebből a szempontból akár ezt a rendszert is nevezhetnénk prototípusnak, de itt azokat a funkciókat implementáljuk elsőként, amelyek a legfontosabbak és a legegyszerűbben elkészíthetőek és amint ezek elkészülnek át is adjuk a rendszert a megrendelőnek és később dogozzuk ki a nem teljesen érthető elvárásokat és a kevésbé fontos rendszerkomponenseket (a sorrend tehát fordított a hibrid fejlesztési módszertanhoz képest). Az is jelentős különbség, hogy a hibrid fejlesztési eljárásnál sosem adjuk át a prototípust a megrendelőnek.

10.2.5 A szoftverek tesztelése

A szoftvertesztelés során megvizsgáljuk, hogy a szoftver rendelkezik-e azokkal a tulajdonságokkal, amelyet a szoftver megrendelői és a szoftver kifejlesztői elvárnak, azaz a szoftver valós körülmények között is alkalmas lesz az üzemszerű és megbízható működésre. A tesztelési folyamat a szoftver minden komponensének a működését megvizsgálja (az összes menüpont összes eleme, bementek és kimenetek vizsgálata, a rendszer viselkedésének vizsgálata helyes és hibás bemeneti adatok bevitele esetén, a stabilitás vizsgálata stb.)

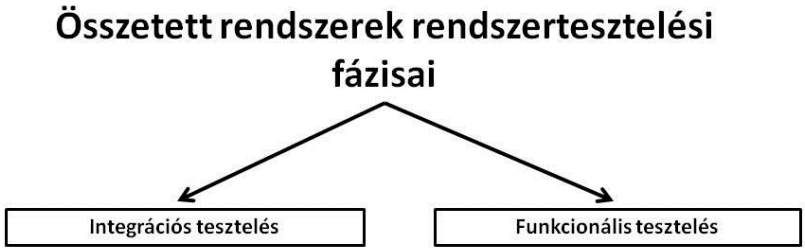
Azt, hogy ki és hogyan végzi majd a tesztelést, az rendszerint már a tervezési fázisban meghatározásra kerül. Nagy és összetett rendszereknél általában a rendszerkomponensek fejlesztőit bízzák meg a komponensek tesztelésével, ami érthető, hiszen ők látják át a legjobban az adott komponens működését és sok energiát és időt emésztene fel, ha ezt a feladatot olyan személy végezné el, aki nem vett részt a fejlesztésben (nem is beszélve arról, hogy az esetleges hibákat úgyis a rendszerkomponens fejlesztőinek kell kijavítaniuk). A rendszer egészének működését viszont rendszerit tapasztalt tesztelőcsoport végzi, akik a komponensek integrációja után a rendszer egészének tesztélését végzik, különös tekintettel azokra a komponensekre, amelyek fejlesztésére eddig nem volt példa.

10.2.6 Rendszertesztelés

A rendszerek részrendszerekből vagy rendszerkomponensekből épülnek fel, ezért a rendszertesztelésről akkor beszélhetünk, ha legalább két részrend-

szer vagy rendszerkomponens integrálásra és tesztelésre kerül. Összetett funkciókat ellátó, rendszerek esetén a rendszertesztelést rendszerint két fázisra bontjuk:

Az integrációs tesztelés során a tesztelőcsoport elsősorban azt vizsgálja, hogy az integrált komponensek megfelelően működnek-e együtt, azaz nagyon leegyszerűsítve azt ellenőrzik, hogy az egyik rendszerkomponens a megfelelő adatokat a megfelelő időben szolgáltatja-e a másik komponens számára. Ha hibát észlelnek a működésben, akkor elsősorban azt próbálják meghatározni, hogy melyik komponens felelős a hibáért. Ha sikerül beazonosítani a komponenst, akkor azt a hiba dokumentálásával visszaadják a fejlesztőknek javításra. A komponensek többfélék lehetnek. A nagyméretű, összetett működésű rendszerekben rendszerint találunk sajátfejlesztésű és készen beszerzett rendszerkomponenst egyaránt. Az integrációs tesztelés során nem minden esetben vizsgálják a rendszer összes komponensét egyszerre, gyakran csak elkülöníthető komponens-csoportokat tesztelnek.



35. ábra: Összetett rendszerek rendszertesztelési fázisai

A funkcionális tesztelés során a rendszer megrendelőjének szempontjából vizsgáljuk a rendszert, ami azt jelenti, hogy a tesztelőcsoport egyrészt arra koncentrál, hogy az elkészült rendszer mindenben megfelel-e a rendszer megrendelője által megfogalmazott elvárásoknak, másrészt a rendszer stabilitását és robosztusságát is tesztelik. Ennél a tesztelési metódusnál a tesztelőcsoport már nem foglalkozik az egyes komponensek szerepével. Nagyon leegyszerűsítve azt mondhatnánk, hogy ha a komponenseket egy lineáris láncolatnak képzeljük el, akkor a tesztelők csak azt vizsgálják, hogy az első komponens bemenetére adott érték a legutolsó komponens kimenetén megfelelő értéket generál-e, és nem foglalkoznak a lánc többi komponensének a működésével.

A megrendelők bevonása a tesztelési folyamatba attól függ, hogy milyen fejlesztési metódust alkalmazunk. Inkrementális fejlesztés esetén magától értetődően szoros együttműködésre van szükség a rendszer fejlesztői és a rendszer

felhasználó között, így a tesztelések során is természetes, hogy a felhasználók az esetek többségében jelen vannak, hiszen a tesztek eredményeitől is függ a rendszer továbbfejlesztésének a menete. A hagyományos szoftverfejlesztési eljárásoknál viszont csak a tesztelés végső stádiumába vonják be a rendszer megrendelőit, amelyet a rendszer átadása követ.

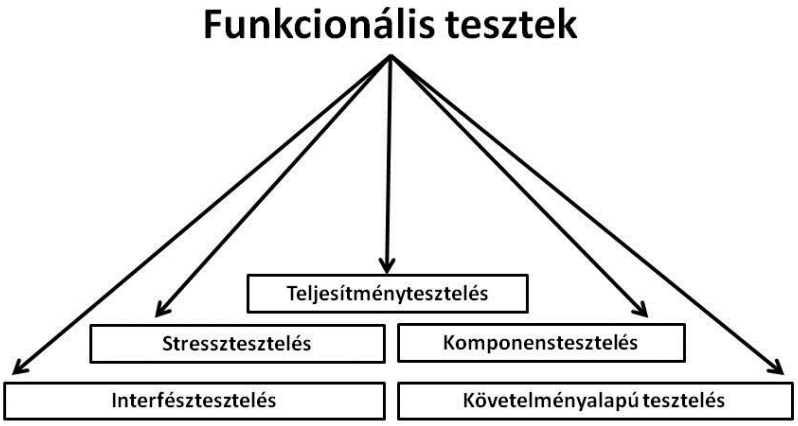
10.2.7 A rendszerintegráció típusai

A rendszerintegráció az jelenti, hogy a rendszer komponenseit azonosítjuk, majd meghatározzuk, hogy mely komponensek működnek szorosan együtt. Az együttműködő komponensekből komponenscsoportokat hozunk létre, majd a komponenscsoportokat integráljuk egységes egésszé.

Attól függően, hogy a rendszer integrációjánál milyen szemléletmódot alkalmazunk, beszélünk fentről lefelé (top-down), vagy lentről felfelé (bottom-up) integrációról. Az előbbi megközelítésnél a rendszer globális működését szem előtt tartva fogunk hozzá a rendszer integrációjához, míg az utóbbinál a komponensek funkciójából kiindulva építjük fel a teljes rendszert. Az elméleti vizsgálódások során ugyan ez a két megközelítési mód jól elkülöníthető, a valóságban azonban szinte minden esetben érvényesülnie kell mindkét szemléletnek.

10.2.8 Funkcionális tesztek

Ahogyan korábban már beszéltünk róla, a funkcionális tesztelés során a rendszer megrendelőjének szempontjából vizsgáljuk a rendszert. Ez azt jelenti, hogy a tesztelőcsoport arra koncentrál, hogy az elkészült rendszer mindenben megfelel-e a rendszer megrendelője által megfogalmazott elvárásoknak. Nem tesztelik tehát az implementációs folyamatot, sem a komponensek működését külön-külön, csak a rendszer egészét, illetve azt, hogy a rendszer a stabilitás, a teljesítmény és robosztusság szempontjából is megfelel-e a követelményeknek és átadható-e a szoftver megrendelőjének. Nagyon leegyszerűsítve azt mondhatnánk, hogy a komponenseket egy lineáris láncolatnak elképzelve a tesztelők csak azt vizsgálják, hogy az első komponens bemenetére adott érték a legutolsó komponens kimenetén megfelelő értéket generál-e, és nem foglalkoznak a lánc többi komponensének a működésével. Ezt a fajta tesztelést ún. fekete doboz tesztelésnek nevezik, mert a tesztelők úgy vizsgálják a rendszert, mintha nem ismernék a belső felépítését: csak a rendszer bemenetei és kimenetei közötti összefüggést vizsgálják.



36. ábra: Funkcionális tesztek típusai

A tesztelés során fontos szempont, hogy lehetőleg minden bemenetet és minden kimentet vizsgáljunk meg, illetve a rendszer bemeneteit ne csak olyan értékekkel vizsgáljuk, amelyek a rendszer által elfogadott határértékeken belül vannak, hanem olyanokkal is, amelyekről tudjuk, hogy hibaüzeneteket fognak generálni.

A tesztelés során az is elvárás, hogy azonos bemenetre a rendszer mindig azonos kimenettel reagáljon, függetlenül attól, hogy hányszor ismételjük meg az eljárást.

10.2.9 Teljesítménytesztelés

A rendszer integrációja után lehetőségünk nyílik a rendszer teljesítményének vizsgálatára. A teljesítménytesztelés azt jelenti, hogy működés közben vizsgáljuk meg a rendszer stabilitását. A tesztelés során megvizsgáljuk, hogy üzemszerű működés közben mekkora terhelés fogja érni a rendszert. A teljesítménytesztelést kivitelezése során az első tesztlépcsőben az üzemszerű terhelésnél valamivel kisebb terhelésnek tesszük ki a rendszert, majd huzamosabb ideig vizsgáljuk a rendszer stabilitását üzemszerű terhelés mellett, végül a terhelést folyamatosan növeljük egészen addig, amíg a rendszer működése instabillá válik.

10.2.10 A stressztesztelés funkciói

Stressztesztelésnek nevezzük azt a tesztelési eljárást, amikor a rendszert jóval az üzemszerű működés feletti szinten terheljük (lásd az előző pontban).

Ezzel a teszteléstípussal egyrészt azt vizsgáljuk, hogy milyen terhelés mellett képes még a rendszer hosszú távon is stabilan működni, másrészt azt, hogy hogyan működik a rendszer olyan esetekben, amelyek tervezés közben nem láthatóak előre.

A terhelés nem csak a rendszer stabilitását vizsgálja, hanem azt is, hogy az extrém terhelés milyen jellegű hibákhoz vezet. Más és más a rendszer megítélése attól függően, hogy milyen hibát eredményez a stressz helyzet, hiszen míg pl. a rendszer lassulása természetes és elfogadható jelenség, addig az adatvesztés, vagy a rendszer egyes funkcióinak működésképtelenné válása már veszélyeket rejt magában.

10.2.11 Követelményalapú tesztelés

A követelményalapú tesztelés azt jelenti, hogy megvizsgáljuk a rendszerrel szemben támasztott, a rendszer specifikációjában meghatározott követelmények teljesítését. Ebből a megfogalmazásból már sejthető, hogy a követelménytesztelés a rendszerspecifikációból indul ki, és a specifikációban megfogalmazottak elemzésével azonosítja azokat a jellemzőket, amelyek vizsgálatára tesztelési eljárásokat lehet létrehozni. A teszteket úgy tervezik meg, hogy a fejlesztési eljárásban nem résztvevők tesztelők számára is egyértelmű legyen, hogy a rendszer az adott követelményt teljesíti vagy nem. Éppen ezért ezt a tesztelési eljárást inkább a validáló, mintsem a hibakereső eljárásokhoz sorolhatjuk.

10.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

10.3.1 Összefoglalás

A tízedik lecke feladata az volt, hogy a hallgatókat megismertesse a rendszerimplementáció és a rendszertelepítés alapvető jellemzőivel. Ennek keretein belül szó volt a gyors szoftverfejlesztés és az inkrementális fejlesztés előnyeiről és hátrányairól, majd a hibrid fejlesztési módszerekről. A lecke második részében a szoftvertesztelés lépéseit vettük górcső alá, melynek során megvizsgáljuk a rendszertesztelést, az integrációs tesztelést, a funkcionális teszteket, a teljesítménytesztelést és a követelményalapú tesztelést.

10.3.2 Önellenőrző kérdések

1. Ismertesse a gyors szoftverfejlesztés jellemzőit!
2. Ismertesse az inkrementális fejlesztés előnyeit!
3. Ismertesse az inkrementális fejlesztés hátrányait!

4. Ismertesse a hibrid fejlesztési módszereket!
5. Ismertesse a szoftvertesztelés módszereit!
6. Ismertesse a rendszertesztelés folyamatát!
7. Ismertesse az integrációs tesztelés jellemzőit!
8. Ismertesse az integráció típusait!
9. Ismertesse a funkcionális tesztek jellemzőit!
10. Ismertesse a teljesítménytesztelés legfontosabb tudnivalóit!
11. Mik a stressztesztelés jellemzői?
12. Mit tud elmondani a követelményalapú tesztelésről?

11. LECKE: RENDSZERKARBANTARTÁS

11.1 CÉLKITŰZÉSEK ÉS KOMPETENCIÁK

A tizenegyedik lecke célja az, hogy megismertesse a hallgatókat a rendszerkarbantartás legfontosabb ismereteivel. Ennek során a hallgatók betekintést nyernek a programevolúció dinamikájába, a szoftverkarbantartáshoz kapcsolódó feladatok megvalósítási folyamataiba és a szoftverkarbantartás típusaiba. A lecke második részében szót ejtünk a karbantartási költségek előrejelzéséről, az evolúciós folyamatok jellemzőiről és a sürgősségi javításokkal kapcsolatos tudnivalókról.

11.2 TANANYAG

Rendszerevolúció az átadás után

- A programevolúció dinamikája
- Rendszerkarbantartás
- A rendszerkarbantartás típusai
- Karbantartási költségek
- A karbantartás tervezése
- Rendszerevolúció
- Soron kívüli karbantartási igények

11.2.1 Rendszerevolúció az átadás után

A rendszer átadása, telepítése után a rendszerfejlesztők feladata még korántsem ért véget. Ennek egyrészt az az oka, hogy a rendszer használatára ki kell képezniük a helyi munkaerőt, másrészt az átadott rendszeren szinte minden esetben kisebb-nagyobb változtatásokat kell eszközölni annak érdekében, hogy a rendszer optimálisan működjön.

A változtatások többfélék lehetnek: a rendszer felhasználói csak használat közben ismerik meg alaposabban az adott rendszert, ezért előfordulhat, hogy a hatékony működés érdekében szeretnék, ha a rendszer funkcionalitása újabb elemekkel is gyarapodna. A változtatásokat indokolhatják olyan kisebb-nagyobb hibák is, amelyek csak a rendszer működtetése közben derülnek ki, és amelyeket a rendszer tervezőinek a rendszer használóival közösen kell kijavítaniuk. A működési környezet előre nem látható változásai is indokolhatják a rendszer egyes elemeinek változtatását, de a biztonsággal és a kompatibilitással kapcso-

latos követelményekben történő változás is szerepet játszhat a változtatások szükségességében.

A rendszert érintő változtatások tehát elkerülhetetlenek a rendszer teljes élettartama alatt, éppen ezért a rendszerfejlesztési tevékenységet sokan a rendszer hasznos élettartamát folyamatosan végigkísérő ciklikus tevékenységnek tekintik, melynek során a rendszerfejlesztés szinte minden állomásán időről-időre végigmennek, és az új követelményeknek illetve a körülményeknek megfelelően végzik el a kisebb-nagyobb módosításokat.

Ez a megközelítési mód akkor alkalmazható, ha a rendszer fejlesztői és működtetői a fejlesztés és az üzemeltetés alatt szoros kapcsolatban vannak egymással (ideális esetben ugyanannak a cégnek a tagjai). A valós körülmények között ez viszonylag ritka, hiszen a legtöbb cég nem engedheti meg magának, hogy olyan feladtok elvégzésére alkalmas embereket foglalkoztassanak (pl. rendszertervező mérnökök), akiknek a szaktudására esetleg évekig nem lesz szükség. Éppen ezért sokkal gyakoribb, hogy ez külső céget bíznak meg a rendszer kifejlesztésével, és az adott cég humán erőforrása csak működteti az újonnan kifejlesztett rendszert, és az igényelt változtatásokat a fejlesztő cég emberei végzik el. Ebben az esetben elengedhetetlen a szoros együttműködés már a fejlesztési fázisban, és a mindenre kiterjedő, precíz rendszerdokumentáció átadása a rendszer telepítése során.

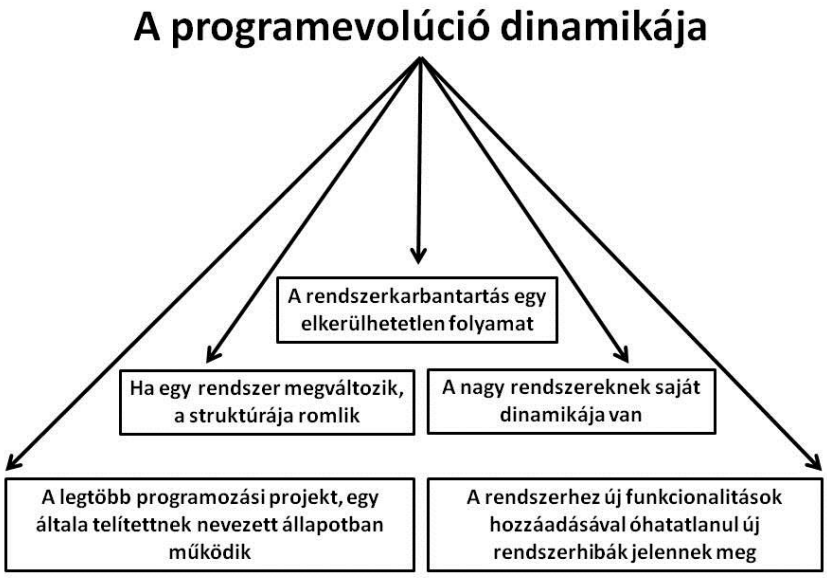
Arra is találhatunk számtalan példát, hogy egy cég megbíz egy fejlesztő céget az informatikai rendszer fejlesztésével, majd a rendszer elkészülte és telepítése után egy másik céggel szerződést köt a rendszer üzemletetésére és karbantartására. A rendszer üzemeltetése és folyamatos fejlesztése szempontjából ez a megoldás a legbonyolultabb, hiszen három szereplővel kell folyamatosan egyeztetni a hatékony működés érdekében.

Még bonyolultabbá teheti a folyamatot, hogy a rendszerfejlesztés területén dolgozó kisebb cégek élettartama viszonylag rövid, cégek egyesülnek, bomlanak fel vagy szűnnek meg. A rendszer átadása utáni félreértések elkerülése érdekében még a fejlesztés megkezdése előtt érdemes tisztázni és szerződésbe foglalni, hogy mely tevékenység tartozik majd a rendszerkarbantartás ingyenes területéhez és mely tevékenységért illeti meg külön díjazás a fejlesztő, karbantartó céget.

11.2.2 A rendszerevolúció dinamikája

A rendszerevolúciós dinamikája a rendszerváltozások tanulmányozása során érthető meg. Számos tanulmány készült ebben a témakörben, az egyik legszélesebb körben elfogadott a Meir Lehman által először 1980-ban publikált (és azóta többször átdolgozott), Programok, életciklusok, és a szoftverevolúció

törvényei című mű, amelyre csak mint Lehman-törvények szoktak hivatkozni. Eredetileg Lehman ezeket a megállapításokat az ún. E-Type programokra tette, de véleményünk szerint ezek a megállapítások általánosan alkalmazhatóak a rendszerfejlesztésben.



37. ábra: A programevolúció dinamikája

Lehman első törvénye a folyamatos változásról szó, ami azt jelenti, hogy a szoftvereket folyamatosan fejleszteni (az elvárásokhoz adaptálni) kell, ellenkező esetben működésük kevésbé lesz kielégítő.

A második törvény állítása szerint, a rendszerek fejlődésével azok bonyolultsága növekszik, amit csak a rendszer karbantartásába befektetett munkával tudunk megelőzni.

A harmadik törvény a rendszer ismeretének a megőrzésével foglalkozik, azaz ahhoz, hogy rendszer a való világ elvárásainak megfelelően fejlődjön, ahhoz nagyon alaposan ismerni kell a rendszer működését és a rendszer fejlesztésének módszertanát (miért az adott módon került kifejlesztésre a rendszer).

A negyedik törvény a rendszerek folyamatos növekedéséről szól, ami azt jelenti, hogy ahhoz, hogy a rendszer alkalmazkodni tudjon a való világ kihívásaihoz, teljes életciklusa során növekednie kell, ami szoros összefüggésben áll a második törvénnyel (a rendszer összetettségének szükségszerű növekedése).

Az ötödik törvény a romló minőség törvénye, ami arra utal, hogy a rosszul értelmezett modifikáció hibás vagy nem kielégítő működéshez vezethet, ami szoros összefüggésben áll a második törvénnyel (a rendszer összetettségének szükségszerű növekedése diszfunkcionalitáshoz vezethet, ha nem törekszünk tudatosan a rendszer integrációjának megőrzésére).

A hatodik törvény szerint a rendszerek működését folyamatosan ellenőrizni kell annak érdekében, hogy az előzőekben felsorolt tényezők negatív hatását csökkentsük.

11.2.3 Rendszerkarbantartás

A karbantartás a hétköznapi életben azt jelenti, hogy tervszerű intézkedésekkel biztosítjuk egy adott berendezés folyamatos és zökkenőmentes működését. Ebben az értelemben rendszerint nem cél a működés megváltoztatása, sőt, a karbantartás célja a működés megváltozásának elkerülése. .A rendszerfejlesztés kontextusában azonban a rendszerkarbantartás azt jelenti, hogy kisebb-nagyobb mértékben megváltoztatjuk a rendszer működését. Ennek az okáról a lecke elején már beszéltünk: „a rendszer felhasználói csak használat közben ismerik meg alaposabban az adott rendszert, ezért előfordulhat, hogy a hatékony működés érdekében szeretnék, ha a rendszer funkcionalitása újabb elemekkel is gyarapodna. A változtatásokat indokolhatják olyan kisebb-nagyobb hibák is, amelyek csak a rendszer működtetése közben derülnek ki, és amelyeket a rendszer tervezőinek a rendszer használóival közösen kell kijavítaniuk. A működési környezet előre nem látható változásai is indokolhatják a rendszer egyes elemeinek változtatását, de a biztonsággal és a kompatibilitással kapcsolatos követelményekben történő változás is szerepet játszhat a változtatások szükségességében”.

A rendszerkarbantartást ennek megfelelően úgy is értelmezhetjük, hogy abban az esetben, ha a rendszer tökéletesen működik, akkor a rendszerkarbantartás célja a jelenlegi állapot fenntartása (valós körülmények között ez nagyon ritka), de sokkal gyakoribb az a megközelítés, hogy a rendszer átadása után a rendszerkarbantartás a rendszer működésének tervszerű megváltoztatása.

38. ábra: A szoftverkarbantartás típusai

A rendszerkarbantartás során sor kerülhet a szoftverkomponensek apróbb működési hibáinak kijavítására, de a karbantartás megoldást jelenthet a tervezési hibáknak a kiküszöbölésére, illetve a rendszer funkcióinak a bővítésére is. A rendszert működésének a megváltoztatását legegyszerűbben a szoftverelemek működésének megváltoztatásával érhetjük el. A nagyobb léptékű változtatáshoz már rendszerit módosítani kell valamelyik rendszerkomponenst, illetve bizonyos esetekben (pl. rendszerfunkciók bővítése) az optimális működés már csak úgy érhető el, ha új rendszerkomponenst fejlesztünk és integráljuk a már meglévő komponensek közé.

11.2.4 A rendszerkarbantartás típusai

- A legegyszerűbb a nem megfelelően működő szoftverelemek hibáinak a kijavítása. Az ilyen típusú hibák rendszerint alacsony anyagi és humánerőforrás ráfordítással kijavíthatóak, hiszen a karbantartás itt nem érinti a többi rendszerkomponenst. Az információs rendszerekben éppen ezért ameddig csak lehet, megpróbálják a szoftverelemek megváltoztatásával kiküszöbölni a tervezés hibákat, hiszen ez a legköltségtakarékosabb megoldás.
- Ezzel szemben a legköltségesebb a rendszerkomponensek hibáinak javítása, vagy az új komponensek tervezése és integrálása, mert ez magával hozhatja a rendszer újratervezésének szükségességét.

- A rendszerkarbantartás harmadik típusába tartoznak azok az esetek, amikor a rendszer környezete változik meg, és ehhez kell adaptálni a rendszer működését. Teljesen természetes, hogy a rendszerek nem elszigetelten, hanem egy adott környezetben kell, hogy működjenek. A rendszer működési környezetébe bele tartoznak a rendszert körülvevő hardver- és szoftvertényezők, de ide tartozhatnak a rendszert körülvevő környezet fizikai jellemzői is, pl. hőmérséklet, páratartalom stb. (pl. ha egy vállalat a telephelyét áteszi egy skandináv országból a trópusokra, nyilvánvalóan újra kell tervezni a teljes rendszer hűtési alrendszerét).

Egy rendszer átadása után a karbantartási típusok között nincs minden esetben éles határvonal. Természetesen rendszer készítőinek és a felhasználóknak is az az érdeke, hogy a hibák a lehető leggyorsabban és a legkisebb anyagi ráfordítás mellett kijavításra kerüljenek, ezért elsőként megpróbálják a szoftverelemek működésének megváltoztatásával orvosolni a hibát.

Időt kell szánni arra is, hogy a rendszer felhasználóit betanítsuk a rendszer használatára. Egy elvileg tökéletesen működő rendszer is hibajelenségeket fog produkálni, ha a felhasználók nem megfelelően használják. Ráadásul ezek a hibák nehezen feltárhatóak, hiszen a hiba nem a rendszerben van. Ritkán az is előfordul, hogy a rendszer felhasználói szándékosan okoznak hibát, illetve akkor is hibára panaszkodnak, ha a rendszer tökéletesen működik. Ennek rendszerint az az oka, hogy az új rendszer megváltoztatta a szakmai hierarchiát az adott környezetben, és ennek a folyamatnak a vesztesei ellenségesen viselkednek az új rendszer bevezetésével kapcsolatosan. Éppen ezért nem szabad elfeledkeznünk arról, hogy a rendszer működési környezetébe bele tartozik a felhasználók szociokulturális környezete is.

Általánosságban elmondhatjuk, hogy a rendszer alapos megtervezése és a folyamatos konzultáció a megrendelővel kifizetődő, mert a már elkészült rendszer újratervezése (tervezési hibák miatt, vagy azért mert nem sikerült teljes mértékben megérteni a megrendelő kívánságait) nagyon költséges és időigényes, nem is beszélve a fejlesztő céget ért presztízsveszteségről.

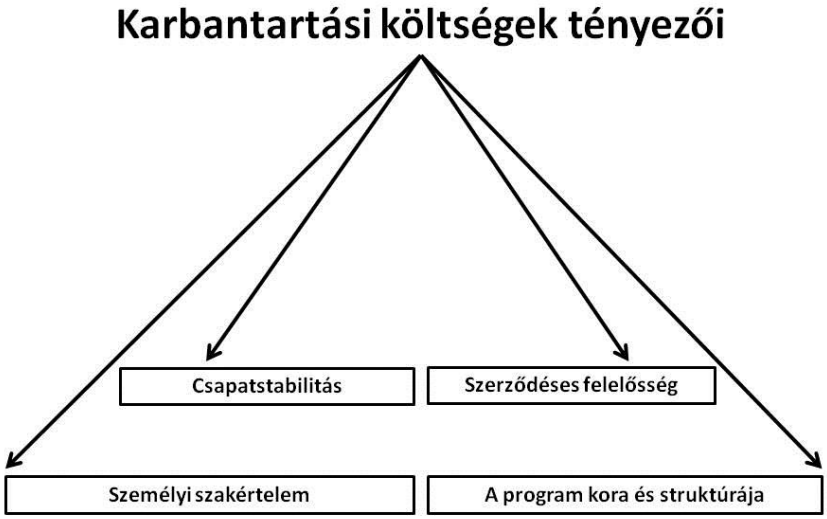
11.2.5 A rendszer karbantartásának költségei

Ahogyan az előző pontban már beszéltünk róla, a már elkészült rendszer újratervezése (tervezési hibák miatt, vagy azért mert nem sikerült teljes mértékben megérteni a megrendelő kívánságait) nagyon költséges. Éppen ezért a rendszerfejlesztési folyamat teljes ideje alatt az alábbiak figyelembevételével kell törekednünk arra, hogy ezt elkerüljük:

- Lehetőség szerint a rendszerfejlesztők végezzék el a karbantartási munkálatokat. Ahogyan a lecke elején már beszéltünk róla, a valós körülmé-

nyek között ez viszonylag ritka, sokkal gyakoribb az, hogy külső céget bíznak meg a rendszer kifejlesztésével, és az adott cég humán erőforrása csak működteti az újonnan kifejlesztett rendszert. Ebben az esetben sok időt és energiát igényel a rendszer működésének megértése, és csak ezután lehet szó a rendszer karbantartásáról.

- A rendszer átadása utáni félreértések elkerülése érdekében még a fejlesztés megkezdése előtt érdemes tisztázni és szerződésbe foglalni, hogy mely tevékenység tartozik majd a rendszerkarbantartás ingyenes területéhez és mely tevékenységért illeti meg külön díjazás a fejlesztő, karbantartó céget. Ha a karbantartást nem a fejlesztő cég végzi, akkor valószínűleg kevesebb energiát fordítanak majd a fejlesztési és működtetési folyamat precíz dokumentálására, ami nehézkessé teheti (esetenként meg is akadályozhatja) a rendszer karbantartását.
- A rendszerek karbantartása rendszerint alacsonyabb presztízsű tevékenység a rendszerfejlesztők körében, mint a rendszerek fejlesztése. Ennek az a következménye, hogy a legképzettebb humán erőforrás foglalkozik a fejlesztési feladatokkal és a kevésbé képzettek a karbantartással. Szakmai szempontból ez a munkamegosztás kifejezetten rossz hatással van a rendszer működtetésére: képzettebb szakemberek által fejlesztett rendszerek működésének a megértése nem könnyű feladat, nem is beszélve a rendszerfunkciók megváltoztatásának feladatáról.
- A rendszer életciklusa és kora jelentős hatással van a rendszer karbantartására. A hasznos életciklusának vége felé a rendszer karbantartására a rendszer üzemletetői nem szívesen szánnak nagyobb összeget. Különösen igaz ez akkor, ha a rendszer kifejezetten régen készült, amikor még nem alkalmazták általánosan a korszerű rendszerfejlesztési metódusokat és a fejlesztési, üzemeltetési dokumentáció elkészítésére és gondozására sem fektettek megfelelően nagy hangsúlyt. Ilyen esetekben előfordulhat, hogy lehetetlen tökéletesen megérteni és az új elvárásokhoz adaptálni a rendszer működését.



39. ábra: A rendszerkarbantartás költségeit meghatározó tényezők

11.2.6 A karbantartás tervezése

A karbantartás tervezése nem könnyű feladat. Mivel a rendszer működésének kisebb-nagyobb mértékben történő megváltoztatása elkerülhetetlen, ezért nem könnyű olyan szerződést kötni a rendszer tervezői és a rendszer felhasználó között, amely minden részletre kiterjedően szabályozza a két fél viszonyát.

A rendszer megrendelője nem akar túl nagy összeget áldozni a rendszer karbantartására, hiszen azért rendelte meg a rendszert, hogy az minden probléma nélkül működjön.

A rendszer tervezői is úgy gondolják, hogy a rendszer átadásával a munka nagy részét elvégezték, a fejlesztő humánerőforrást átcsoportosíthatják újabb rendszerek tervezésére, amely jövedelmezőbb, mint a már elkészült rendszerek karbantartása.

Az előzőekben viszont már láttuk, hogy a rendszer időről-időre történő kisebb-nagyobb fejlesztése elkerülhetetlen az optimális működés szempontjából.

Ráadásul a rendszer fejlesztésének megrendelésekor lehetetlen megjósolni, hogy milyen változtatások lesznek szükségesek a rendszer átadása után. Kompromisszumos megoldásként a felek megegyezhetnek pl. abban, hogy egy átalánydíj fejében elvégzik a rendszer általános karbantartását, ha szükséges, módosítanak egyes szoftverkomponenseket, de új rendszerkomponenseket

illetve rendszerfunkciókat csak külön díjazásért fejlesztenek ki. Sajnos ez a szerződésforma nem jelent megoldást arra a problémára, ha a fejlesztő és a megrendelő másképp értelmezte a fejlesztendő rendszer egyes részeinek a funkcionalitását, ebben az esetben hosszas tárgyalásokra lehet szükség a megegyezéshez.

A tervezési fázisban azt is nagyon nehéz megmondani, hogy az általános karbantartáson túl mikor lesz majd szükség rendszerkomponensek vagy új rendszerfunkciók fejlesztésére, azonban a rendszer és a rendszerkörnyezet kapcsolatának vizsgálata segíthet ennek a megítélésében:

Ahogyan már beszéltünk róla, az információs rendszerek nem elszigetelve működnek egy adott vállalatnál: bemeneti adatokat kapnak a többi rendszertől és kimeneti adatokat szolgáltatnak a többi rendszer számára. Általánosságban elmondhatjuk, hogy minél nagyobb a rendszerkapcsolódási pontok (kimenetek és bemenetek) száma, annál nagyobb eséllyel lesz szükség a karbantartásra.

A másik jellemző, ami a rendszerkörnyezetben hatással lehet a rendszer karbantartására, az a vállalati folyamatok fejlődésének dinamikája: minél dinamikusabban fejlődik a vállalatnál egy-egy folyamat (pl. egy nagyon sikeres termék értékesítési számainak felfutása), annál valószínűbb, hogy a rendszer működésében változásokat kell eszközölni.

11.2.7 Rendszerevolúció

A rendszerevolúció legfontosabb motiválói a rendszer működésének megváltoztatására törekvő igények, amelyek irányulhatnak arra, hogy a hatékonyabb működés érdekében a rendszer funkcionalitása újabb elemekkel gyarapodjon, de a változtatásokat indokolhatják olyan kisebb-nagyobb hibák is, amelyek csak a rendszer működtetése közben derültek ki, és amelyeket a rendszer tervezőinek a rendszer használóival közösen kell kijavítaniuk. A működési környezet előre nem látható változásai is indokolhatják a rendszer egyes elemeinek változtatását, de a biztonsággal és a kompatibilitással kapcsolatos követelményekben történő változás is szerepet játszhat a változtatások szükségességében.

Az evolúciós folyamat első eleme egy elemzési folyamat, amelynek a célja, hogy megvizsgálja, hogy a tervezett változtatás milyen szinten érinti a rendszer egészét, és a változtatás elvégzése milyen anyagi és humán erőforrás ráfordítást és időt igényel. Ennek alapján születik döntés a a változás megvalósításáról, vagy elvetéséről. Itt kell megjegyezni, hogy a változtatásokra irányuló kéréseknek vannak formális (kérelem benyújtása hivatalos formában) és informális (szóbeli) formái. Rendszerváltoztatási a kérelmeket rendszerint nem hajtják azonnal végre, hanem megadott időn belül beérkezett kérelmeket egyszerre

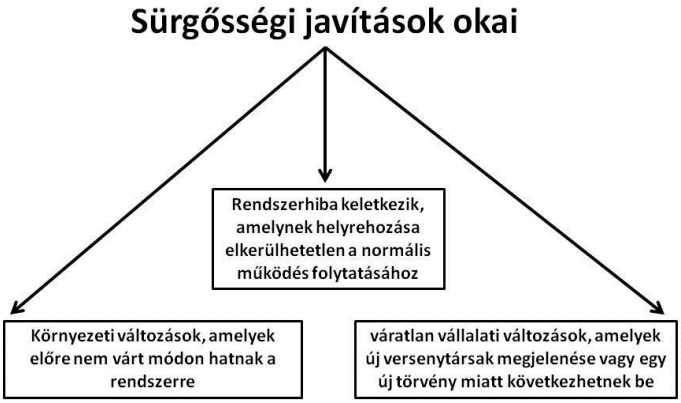
dolgozzák fel, és ekkor születik döntés arról, hogy melyik kerül végrehatásra és melyik nem.

Az elfogadásra került módosítási kérelmeket, implementálják, validálják, majd kiadják a rendszer új verzióját, azaz a fejlesztési folyamatot gyakorlatilag megismétlik. Ez a folyamat gyakran a rendszer teljes élettartama alatt időről-időre megismétlődik. Minden változás, ami a rendszer működését befolyásolja, dokumentálásra kell, hogy kerüljön, ellenkező esetben a ciklikus fejlesztési folyamat miatt nem lehet nyomon követni, hogy az adott komponensnek melyik a legfejlettebb verziója, a rendszer működésének a veszélyeztetéséhez vezethet.

11.2.8 Soron kívüli karbantartási igények

Az előző pontban leírt változtatási kérelmek feldolgozási módja (a kérelmeket rendszerint nem hajtják azonnal végre, hanem megadott időn belül beérkezett kérelmeket egyszerre dolgozzák fel, és ekkor születik döntés arról, hogy melyik kerül végrehatásra és melyik nem) nem minden esetben járható út, mert bizonyos esetekben az azonnali beavatkozás nélkül a rendszer nem képes működni.

Azonnali beavatkozást igényelhet egy olyan rendszerhiba, ami csak a beüzemelést követően jelentkezik és pl. egy kritikus fontosságú rendszerkomponenst érint, amely komponensnek a kimeneti adatait számos más komponens használná fel a rendszerben. Ebben az esetben nincs idő a formális változtatáskérések lefuttatására és a kérés elemzésére (ez utóbbira nincs is szükség, hiszen a rendszer nem működik a javítás nélkül), hanem a lehető leggyorsabban meg kell keresni a hiba okát és helyreállítani a működést.



40. ábra: Soron kívüli javítások okai

Elvileg itt is dokumentálni kellene minden változtatást a rendszeren, de pont az idő hiánya miatt ez nem minden esetben történik meg. Ráadásul a gyors változtatási igény miatt rendszerint idő arra, hogy a rendszerfejlesztők elemezni tudják, hogy a változtatás milyen következményekkel jár a rendszer egészére. Ez a két tényező (a dokumentáció és a rendszerváltoztatás elemzésének a hiánya) jelentősen ronthatja a rendszer integritását és veszélyeztetheti a rendszer működését.

11.3 ÖSSZEFOGLALÁS, KÉRDÉSEK

11.3.1 Összefoglalás

A tizenegyedik lecke célja az volt, hogy megismertesse a hallgatókat a rendszerkarbantartás legfontosabb ismereteivel. Ennek során a hallgatók betekintést nyernek a programevolúció dinamikájába, a szoftverkarbantartáshoz kapcsolódó feladatok megvalósítási folyamataiba és a szoftverkarbantartás típusaiba. A lecke második részében szót ejtettünk a karbantartási költségek előrejelzéséről, az evolúciós folyamatok jellemzőiről és a sürgősségi javításokkal kapcsolatos tudnivalókról.

11.3.2 Önellenőrző kérdések

1. Mit tud a rendszerevolúció dinamikájáról?
2. Sorolja fel a szoftverkarbantartás típusait!
3. Mit tud a karbantartási költségekről?
4. Mit tud az evolúciós folyamatokról?

12. LECKE: ÖSSZEFOGLALÁS

12.1 ÖSSZEFOGLALÁS

A rendszertervezés egy komplex folyamat, amelynek a célja egy adott cél elérése érdekében működő információs rendszer tervezése, létrehozása, telepítése és a működési feltételeinek a biztosítása.

Néhány tíz évvel ezelőtt a rendszertervezés minden szervezetnél egy kifinomult, már-már művészi tevékenység volt, ami hosszú időt és egyedi megoldások kifejlesztését igényelte. Azonban az információs rendszerek iránti igények exponenciális növekedésével a rendszertervezési folyamat egyre inkább tudományosan és gyakorlati tapasztalatok által megalapozott, jól körülhatárolható lépések és eljárások sorozatává szelídült, amelyből néhányat bemutattunk könyvünkben.

A könyv második fejezetében a célunk az volt, hogy a hallgatókat megismertessük a rendszertervezés alapfogalmaival, tanulmányozzák a rendszertervezés rövid történetét, legyenek tisztában az információs rendszerek tervezésének alapvető tudnivalóival és a tervezés ciklusaival. Ismerjék meg a legfontosabb rendszertervezési elveket, rendelkezzenek információval az alábbi metódusokról:

- Vízesés modell
- Evolúciós modell
- Komponensalapú és újrafelhasználható rendszerfejlesztés
- Spirális fejlesztés
- Gyors Alkalmazásfejlesztés (RAD)
- RUP (Rational Unified Process)
- eXtreme Programming
- Agilis szoftverfejlesztés

A harmadik lecke célja az volt, hogy a hallgatók megismerkedjenek a vállalati információs rendszerek szoftvereinek forrásaival. Ennek során szót ejtettünk az outsourcing jellemzőiről, beszélünk arról, hogy milyen előnyei és hátrányai lehetnek annak, ha a szoftverfejlesztést külső IT-cégek segítségéve végezzük. A fejezet második részében megvizsgáltuk az előre elkészített szoftvercsomagok szerepét, és vállalatirányítási információs rendszerek számítási felhőbe történő integrálásának lehetőségeit is. Szót ejtettünk a nyílt forráskódú

szoftverek alkalmazásáról, és a házon belüli szoftverfejlesztés jellemzőiről is. A fejezet végén megvizsgáltuk a szoftvervásárlás legfontosabb szempontjait.

A negyedik fejezet célja az volt, hogy a projektmenedzsment legfontosabb ismereteivel felruházza a hallgatókat. A lecke kapcsán szó volt többek között a projektszervezet felépítéséről, a projekt életciklusról, a projekt szakaszairól. A lecke második részében megtárgyaltuk a projektfolyamat sajátosságai, ejtünk szót a projekt-előkészítés szakaszairól, az üzleti vagy szervezeti cél megfogalmazásáról, a megvalósíthatóság vizsgálatáról és a megvalósíthatósági tanulmány elkészítéséről.

A lecke harmadik részében szó volt a célrendszer és sikerkritériumok meghatározásáról, az erőforrásbecslésről és a projekttervezés szakaszairól.

Az ötödik lecke célja az volt, hogy a hallgatókat megismertessük rendszerkövetelmények tervezésének alapvető ismereteivel. Ennek során górcső alá került az információs rendszer financiális hasznának vizsgálata, a követelmények feltárása és elemzése illetve a követelmények felderítése. Beszéltünk továbbá arról is, hogy a folyamat végén meg kell vizsgálnunk, hogy a megrendelő által kívánt rendszert definiáltuk-e a folyamat során.

A hatodik lecke célja az volt, hogy megismertesse a hallgatókat a szoftvertervezés és -implementáció legfontosabb lépéseivel. Szó esett az architekturális tervezéshez kapcsolódóan az architekturális modellekről, a modularizálás és moduláris tervezés folyamatáról. A lecke második részében az alrendszerek modulokra bontása kapcsán megismerkedtünk az objektumorientált és adatfolyam modell felbontással, annak előnyeivel és hátrányaival. A lecke végén a vezérlési stílusokról, ezen belül a központosított vezérlésről és az eseményvezérelt rendszerekről beszéltünk.

A hetedik lecke célja az volt, hogy megismertessük a hallgatókat az adatbázis-tervezés legfontosabb lépéseivel. A lecke során beszéltünk az adatbáziskezelés alapjairól, melynek kapcsán többek között szóba került az adatbázisok felépítése, a tulajdonságtípusok, a kapcsolatok és a kapcsolatok tipizálása illetve a kapcsolattípusok. A lecke második részében szót ejtettünk az adatmodellek jellemzőiről, közelebbről megvizsgáltuk a relációs adatmodell tulajdonságait. A lecke végén pedig az adatbázisrendszerek tervezési lépéseiről ejtünk szót.

Ahogyan a bevezetőben már szót ejtettünk róla, az előző fejezetben tárgyalt adatbázisokat adatokkal kell feltöltenünk, hogy be tudják tölteni funkciójukat, amelyre az egyik leggyakrabban használt eszköz az űrlap. A nyolcadik lecke célja az volt, hogy megismertesse a hallgatókat az űrlapokhoz kapcsolódó alapfogalmakkal, így esett szó az űrlapok jellemzőiről (azonosíthatóság, kitöltést segítő információk, áttekinthető felületek, egyértelmű tagolás és definiálás,

egységes forma, következetesség, pontosság, a redundáns adatbekérés elkerülése és feldolgozhatóság), a lecke végén pedig szót ejtettünk az űrlapokon alkalmazott adattípusokról.

A kilencedik fejezet célja az volt, hogy megismertessük a hallgatókat a felhasználói interfészek tervezésének alapelveivel. Ennek során beszéltünk a felhasználói jártasság fogalmáról, szót ejtettünk a felhasználói felület konzisztenciájáról, és megvizsgáltunk olyan, az interfésztervezés szempontjából fontos fogalmakat, mint a minimális meglepetés, a visszaállíthatóság, a felhasználó támogatása és a felhasználói sokféleség figyelembe vétele. A lecke második felében beszéltünk a felhasználói interakciókról, az információmegjelenítés és a színhasználat kérdéseiről.

A tizedik lecke feladata az volt, hogy a hallgatókat megismertesse a rendszerimplementáció és a rendszertelepítés alapvető jellemzőivel. Ennek keretein belül szó volt a gyors szoftverfejlesztés és az inkrementális fejlesztés előnyeiről és hátrányairól, majd a hibrid fejlesztési módszerekről. A lecke második részében a szoftvertesztelés lépéseit vettük górcső alá, melynek során megvizsgáltuk a rendszertesztelést, az integrációs tesztelést, a funkcionális teszteket, a teljesítménytesztelést, a komponenstesztelést, az interfésztesztelést és a követelményalapú tesztelést.

A tizenegyedik lecke célja az volt, hogy megismertesse a hallgatókat a rendszerkarbantartás legfontosabb ismereteivel. Ennek során a hallgatók betekintést nyertek a programevolúció dinamikájába, a szoftverkarbantartáshoz kapcsolódó feladatok megvalósítási folyamataiba és a szoftverkarbantartás típusaiba. A lecke második részében szót ejtettünk a karbantartási költségek előrejelzéséről, az evolúciós folyamatok jellemzőiről és a sürgősségi javításokkal kapcsolatos tudnivalókról.

13. KIEGÉSZÍTÉS

13.1 ÁBRAJEGYZÉK