12. fejezet

A programtervezés alapvető módozatai

Egyszerű programjaink készítése során nem okozott különösebb gondot, hogy a programok szerkezetét, a szükséges függvényeket, eljárásokat, adatszerkezeteket megtervezzük: ezek ilyen egyszerű esetekben természetesen „adódtak”, logikusnak tűntek. A feladat struktúrája természetes módon megjelent a program szerkezetében is, mind az adatszerkezetek, mind a vezérlési szerkezetek szintjén.

Bonyolultabb esetekben azonban a tervezési folyamatot célszerű tudatos módon, szisztematikusan végezni. A gyakorlatbak két alapvető megközelítést, gondolkodási módot találunk: az alulról fölfelé tervezést és a felülről lefelé tervezést.

A továbbiakban áttekintjük e két alapvető módszert. Azt azonban már most meg kell jegyezni, hogy sem egyik, sem másik módszerről nem lehet állítani, hogy jobb a másiknál: jelentős szemléletbeli különbség van a két megközelítés között. Így egyes gyakorlati esetekben hol egyik, hol a másik megközelítés bizonyulhat hatékonyabbnak. De egy valós mérnöki feladat megoldása közben valószínűleg egy kombinált módszert fogunk alkalmazni: a tervezés során váltogatni fogjuk a két módszert.

12.1. Felülről lefelé tervezés

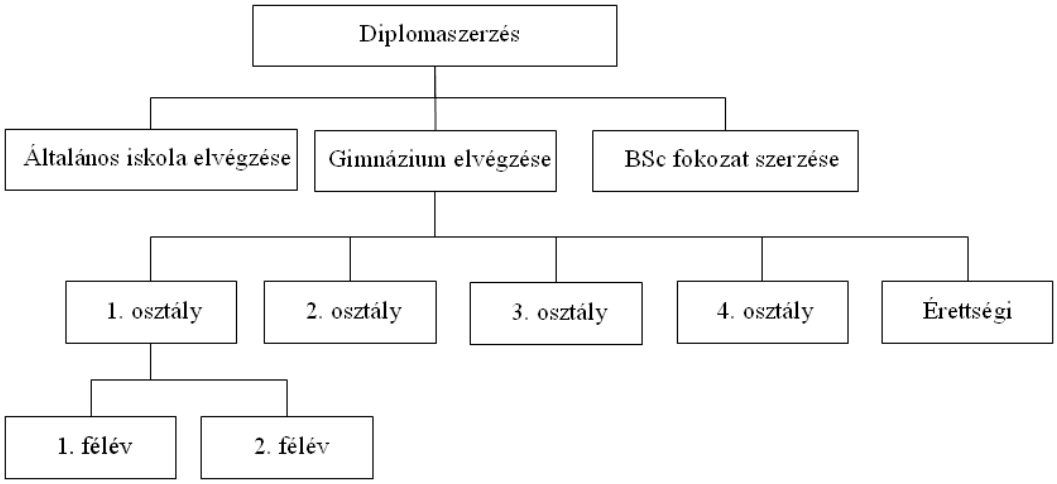
A felülről lefelé (top-down) tervezés alapelve az analízis: a célt elemezve azt kisebb logikai egységekre bonthatjuk, majd ezeket ismét tovább analizálva még kisebb logikai egységeket kapunk. Az analízis-részekre bontás folyamatot addig folytatjuk, míg már tovább nem bontható, elemi tevékenységekig jutunk.

Az felülről lefelé tervezés lépései a következők:

1. A feladatot (célt) egyetlen tevékenységként értelmezzük.
2. Ha van összetett tevékenységünk, akkor azt belátható számú, jól definiált feladatkörű tevékenységre bontjuk.
3. A tevékenységek sorrendjét az összetett tevékenységen belül meghatározzuk.
4. Ismétlés a 2. ponttól mindaddig, amíg csak elemi tevékenységek maradnak.

12.1. példa: Hogyan szerezzünk diplomát? (felülről lefelé tervezéssel)

Az első lépés szerint a diplomaszerzés tevékenységből, mint célból indulunk ki. Mivel ez nem elemi tevékenység, így a 2. lépés szerint ezt jól definiált feladatú részegységekre kell bontani. Egy ilyen logikus felbontás lehet, hogy el kell végezni az általános iskolát, a gimnáziumot, majd az egyetemen pl. BSc fokozatot kell szerezni. Ezzel az eredeti feladatot három kisebb feladatra dekomponáltuk. Ezen kisebb tevékenységek viszonya a feladatban egyszerű: a felsorolt sorrendben kell ezeket végrehajtani. Amennyiben ezen tevékenységek sem tekinthetők elemi tevékenységnek (figyelem: ez a végrehajtótól függ), akkor ezeket is tovább analizálhatjuk és bonthatjuk további részfeladatokra. Például a gimnázium elvégzése a négy év sikeres elvégzését és az érettségi letételét jelentheti. Az egyes tanévek további két félévre bonthatók, stb. A feladat végrehajtását Jackson-ábrán is leírhatjuk, ahogy az ábra mutatja. A felbontást tetszőleges finomságig folytathatjuk mindaddig, amíg a számunkra értelmezhető elemi szintig el nem jutunk. Hasonlóan az általános iskolai, valamint az egyetemi tevékenységek is tovább bonthatók.



12.1. ábra. A diploma megszerzésének folyamata felülről lefelé tervezéssel. Az ábrán még nincs minden tevékenység kifejtve.

Az felülről lefelé tervezés előnyei: a felülről lefelé tervezés szép megoldásokat eredményez, hiszen az adott feladathoz jól illeszkedő struktúrájú megoldást kapunk. A komplex feladattól egyre egyszerűbb részfeladatokon át jutunk el a megoldásig, minden szinten kezelhető méretű problémát oldunk meg. A közbülső lépések során definiált részfeladatok önálló szoftver-egységek (pl. függvények) lesznek, melyek elemi és később definiálandó összetett tevékenységeket tartalmaznak. A tervezés során definiált egységek kapcsolatai, interfészei jól definiáltak, ezért ezek önállóan is kezelhetők, párhuzamosan is fejleszthetők. A kódolás tehát könnyű és áttekinthető, az eredmény jól strukturált, logikus felépítésű lesz.

Az felülről lefelé tervezés hátrányai: A fejlesztés során felülről lefelé haladva egységeink definíciói olyan alegységekre hivatkoznak, melyek egyelőre csak funkcionálisan ismertek, ezek definíciója csak később készül el. Így egy magas szinten megtervezett és létrehozott egység nem tesztelhető – hiszen elemei még hiányoznak. Ezért a felülről lefelé tervezéssel készült szoftverek tesztelése csak a tervezési és fejlesztési folyamat legvégén kezdődhet el.

12.2. Alulról felfelé tervezés

Az alulról fölfelé tervezés (angol nyelvű szakirodalomból gyakran bottom-up tervezésnek is nevezik) alapelve a szintézis, már meglévő elemekből történő építkezés. Az alulról felfelé építkezés feltétele, hogy az elemi tevékenységeket ismerjük, ezeket kezdetben meghatározzuk. Ezen elemi tevékenységeket fogjuk logikusan rendezni, csoportosítani mindaddig, míg a tevékenységeknek egy megfelelő sorrendjét és hierarchiáját ki nem alakítjuk, amely hierarchia csúcsán a feladat megoldása áll.

Az alulról felfelé tervezés lépései a következők:

1. Az elemi tevékenységek meghatározása.
2. A tevékenységek függőségeinek feltárása.
3. A függőségek és a tevékenységek kapcsolódásainak figyelembe vételével a tevékenységek sorrendjének átrendezése úgy, hogy
   a. az új sorrend vegye figyelembe a függőségi viszonyokat és
   b. az új sorrendben a hasonló jellegű feladatok egymás mellé kerüljenek.
4. Az előbbi pontban hasonló jellegűnek ítélt tevékenységek egy összetett tevékenységbe zárása.
5. Ismétlés a 2. ponttól mindaddig, amíg egyetlen (összetett) tevékenység marad: ez a feladat megoldása

Az elemi tevékenységek meghatározása során nem feltétlenül a programnyelv elemi utasításaira, hanem a fejlesztői környezet által nyújtott, kész megoldásokra kell gondolni. Az elemi tevékenységek meghatározása gyakran az újrafelhasználás jegyében történik: amennyiben már van egy kész komponensünk, amely a feladathoz várhatóan jól használható, akkor azt használhatjuk elemi tevékenységként. Gyakori példa az alulról felfelé tervezés illusztrálására a LEGO játék: itt adott az elemkészlet, amelyből létre kell hozni a célunknak megfelelő játékot, például egy robotot. A robot végső kinézetét nagyban befolyásolja a rendelkezésre álló elemkészlet, de az építkezés során valószínűleg a robot lábát, törzsét, kezét, fejét állítjuk össze az elemi komponensekből, majd ezen elemeket állítjuk össze robottá.

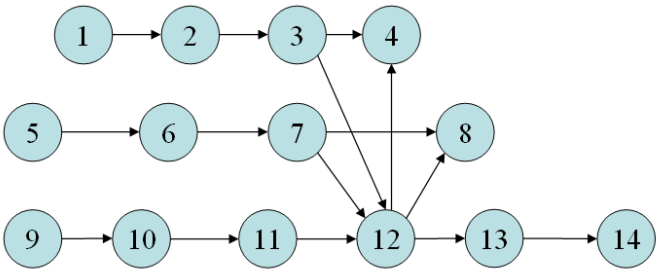
12.2. példa: Tervezzük meg a gyertyafényes vacsora menetét alulról felfelé tervezéssel.

Az első lépés szerint meghatározzuk, összegyűjtjük a felhasználható elemi tevékenységeket. Ez esetünkben pl. a következő lista lehet:

(1) gyertyát vásárlok
(2) a gyertyát az asztalra teszem
(3) a gyertyát meggyújtom
(4) a gyertyát eloltom
(5) szalvétát vásárolok
(6) a szalvétát összehajtom
(7) a szalvétát az asztalra teszem
(8) a szalvétát kidobom
(9) alapanyagot vásárolok
(10) vacsorát készítek

(11) tálalok
(12) eszünk
(13) leszedem a vacsorát
(14) elmosogatok

A második lépésben a tevékenységek függőségeit vizsgálom meg. Például a gyertya asztalra helyezése (2) célszerűen meg kell előzze a gyertya meggyújtását (3), valamint a gyertya eloltása (4) csak az evés (12) után következhet. Az ilyen típusú függőségek ábrázolásának eszköze a függőségi gráf lehet, amelynek csúcsai a tevékenységek, egy A és B csúcs között pedig akkor vezet irányított él (A-ból B-be mutató nyíl), ha B tevékenység függ A tevékenység végrehajtásától. Tehát a fenti probléma függőségi gráfjában pl. létezni fognak a (2) → (3) és (12) → (4) élek. A teljes függőségi gráf a 12.2. ábrán látható.



12.2. ábra. A gyertyafényes vacsora elemi tevékenységeinek függőségi gráfja

A harmadik lépésben a tevékenységek sorrendjét rendezzük át úgy, hogy a függőségi gráf kötöttségeit figyelembe vegyük. Formálisan: keressük az irányított függőségi gráf csúcsainak olyan $c_1, c_2, .., c_n$ sorrendjét, hogy bármely c_i és c_j-re igaz legyen, hogy ha a függőségi gráfba létezik c_i → c_j él, akkor $i < j$. Ráadásul egy olyan sorrendet próbálunk meg keresni, amelyben a hasonló jellegű feladatok egymás mellett foglalnak helyet. Ez a gyakorlatban a gráf egy olyan topológia rendezését jelenti, ahol a rendezett gráfban pl. csak balról jobbra vezetnek élek.

Példánkban a függőségi gráf csomópontjainak egy lehetséges rendezése a következő:

R1: (1), (2), (3), (5), (6), (7), (9), (10), (11), (12), (13), (4), (8), (14)

Egy másik lehetséges rendezés a következő lehet:

R2: (1), (5), (9), (2), (6), (10), (3), (7), (11), (12), (4), (8), (13), (14)

Ezen kívül a csúcsoknak (vagyis a tevékenységeknek) még számos lehetséges rendezése elképzelhető.

Ha az R1 rendezést használjuk, ahol olyan tevékenységeket helyeztünk egymás mellé, amelyeknek tárgya azonos (pl. a szalvéta, a gyertya): így is logikus felépítést kapunk, ha nem is célszerűt (ugyanis többször is el kell mennünk a boltba beszerezni a szükséges eszközöket). Az R2 rendezésben inkább a tevékenység hasonlósága szerint csoportosítottuk a tevékenységeket, így célszerűbb sorozatot kapunk. Az alulról fölfelé tervezés kihívása ennek a döntésnek a meghozása: milyen sorrendben, milyen csoportokat alkossunk, hogy programunk logikus, jól strukturált és hatékony legyen.

Az R2 rendezésből folytatva a tervezést a negyedik lépésben a logikailag összefüggő tevékenységeket egyetlen összetett tevékenységbe zárjuk. Pl. az (1), (5), (9) tevékenységek bevá-

sárlás jellegű tevékenységek, ezeket tekinthetjük egyetlen összetett tevékenységnek (Bevásárlás). A (2), (6), (10), (3), (7), (11) előkészületi tevékenységeket ismét egy közös tevékenységbe zárhatjuk (Előkészületek). A 12. tevékenység önállóan alkotja a Vacsora tevékenységet, míg a (4), (8), (13), (14) tevékenységek az Utómunkálatok összetett tevékenységbe zárhatók.

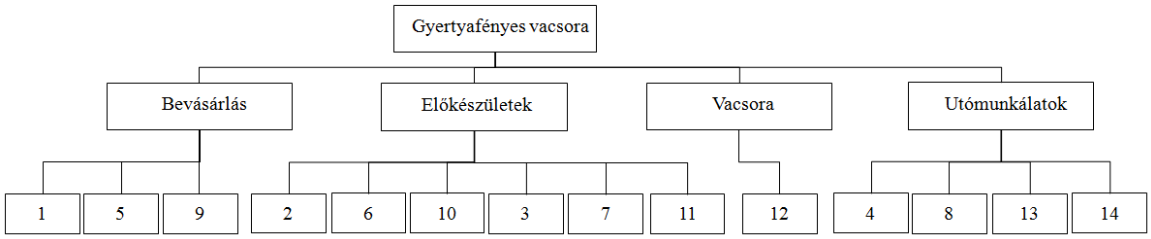
A tevékenységek összevonása után a következő (összetett) tevékenységeink adódtak:

(1') Bevásárlás = {(1), (5), (9)}
(2') Előkészületek = {(2), (6), (10), (3), (7), (11)}
(3') Vacsora = {(12)}
(4') Utómunkálatok = {(4), (8), (13), (14)}

Ezekre a tevékenységekre ismét folytatnunk kell a második lépéstől kezdve a folyamatot. Itt azonban már – a feladat egyszerűsége miatt – elérjük a végső célunkat: ezen négy tevékenység egymás utáni végrehajtása elvezet az eredeti feladat, a gyertyafényes vacsora végrehajtásához. Tehát a gyertyafényes vacsora a következő:

(1") Gyertyafényes vacsora = {(1'), (2'), (3'), (4')}

A teljes megtervezett folyamat a 12.3. ábrán látható.



12.3. ábra. A gyertyafényes vacsora folyamata

Az alulról felfelé tervezés előnyei: Az alulról felfelé tervezés nagyon jól támogatja a már létező szoftverkomponensek újrafelhasználását. A tervezés és implementálás már működő komponensekből történik, így a létrehozott összetett komponensek azonnal ki is próbálhatók, tesztelhetők. Mire a tervezési folyamat végére érünk, egy gyakorlatilag futtatható rendszer áll rendelkezésünkre.

Az alulról felfelé tervezés hátrányai: Mivel kész elemekből építkezünk, szükséges némi mérnöki intuíció ahhoz, hogy a létrehozandó komponenseket jól definiáljuk, mind az ellátandó feladatközöket, mind az interfészeket tekintve. Ezen komponensekből később nagyobb komponenseket építünk, így a komponensek utólagos egymáshoz illesztése komoly feladat lehet.