13. fejezet

Programfejlesztési modellek

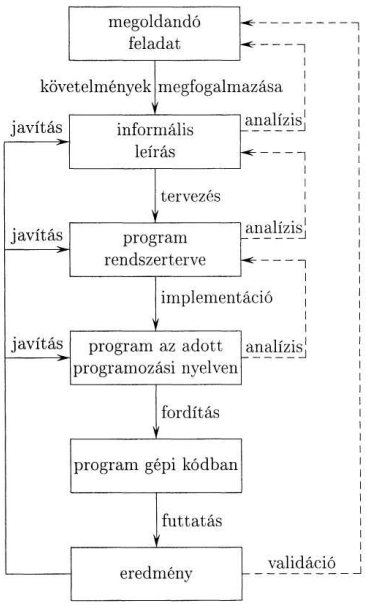
Nagy rendszerek fejlesztési fázisai. Az objektumelvű programozás kialakulása. Típusöröklődés. Az objektumelvű modellezés nézetrendszerei. UML eszközök. Tervminták fogalma.

13.1. Nagy rendszerek fejlesztési fázisai

Nagy rendszerek tervezésekor a teljesen formális specifikálás a rendszer egészére nem, legfeljebb néhány kisebb, kiemelt részfeladatra valósítható meg – a formális specifikáció helyébe a rendszer tervének strukturált, többszintű leírása lép, az implementáció számára szükséges ajánlásokkal, előírásokkal kiegészítve.

A programkészítés hagyományos fázisai.

- követelmények leírása,
- specifikáció,
- tervezés,
- implementáció,
- verifikáció és validáció (ellenőrzés),
- karbantartás.



13.1. ábra. Nagy rendszerek egy általános fejlesztési modellje

13.1.1. Követelmények leírása, specifikáció

Egy probléma előtt meg kell vizsgálni annak megvalósíthatóságát, a megvalósíthatóság mikéntjére vonatkozó lehetőségeket és korlátokat.

13.1.1. Definíció (Megvalósíthatósági tanulmány). *A megvalósíthatósági tanulmány a következő kérdésekre válaszol:*

- *szükséges technikai erőforrások (hardver, szoftver),*
- *szükséges emberi erőforrások,*
- *a rendszer létrehozásának költségei,*
- *a rendszer elkészítésének várható időtartama,*
- *az üzemeltetéssel kapcsolatos feladatok és költségek.*

A követelmények elemzése prototípus segítségével

A rendszerrel szemben támasztott követelmények felmérése többféle módon történhet, ma sokan használják a prototípus segítségét igénybe vevő megközelítést. Ezt nevezzük *evolúciós modell*nek.

Ez a módszer a felhasználón keresztüli visszacsatoláson alapul: a programozó(k) a felhasználó igényei alapján elkészít(enek) egy *prototípust*, mely látszólag magában hordozza a program szükséges funkcionális elemeit. Ezt megtekintve a felhasználó finomíthatja a rendszerrel szemben támasztott igényeit, amihez igazodva újabb prototípus készülhet.

A követelmények két csoportja.

- *funkcionális követelmények:* a rendszer által nyújtandó szolgáltatások,
- *nem funkcionális követelmények:* erőforrások, környezet, átadás, karbantartás, ...

A követelmény leírásának szintjei.

- követelmény-definíció (informális leírás),
- követelmény-specifikáció (szintekre tagolt, egyértelmű, részletes leírás),
- program-specifikáció (rögzített szabályok szerint megfogalmazott, absztrakt leírás).

Nagy rendszerek esetén a követelmények leírása szinte soha nem teljes.

13.1.2. Tervezés

A tervezés során elkészített modellek:

1. Statikus modell
   - a rendszert felépítő programegységek,
   - a programegységek között statikus kapcsolatok, relációk,
   - az egyes egységek feladata.
2. Dinamikus modell
   - a probléma megoldásának módszere,
   - a megoldás során a programegységek együttműködési kapcsolatai,
   - a programegységek közötti információáramlás,
   - a rendszer, illetve az egyes egységek lehetséges állapotai és állapotátmenetei.

3. Funkcionális modell

    - a szolgáltatások megvalósulása (a résztvevő programrészek, információáramlások),
    - az adatáramlásokban megjelenő leképezések, transzformációk,
    - implementációs ajánlások (implementációs stratégia, programozási nyelv, programozási környezet, tesztelési terv).
    - az egyes egységek feladata.

Elterjedt tervezési módszerek a funkcionális és az *objektumelvű* megközelítés. Előbbiben a rendszer funkciói, a működés részfeladatai felől közelítjük meg a problémát, míg a második módszer középpontjában a program adatai állnak.

13.1.3. Megvalósítás

Kérdések:

- az adatok ábrázolása (adatszerkezetek, reprezentáció),
- a leképezések és események megvalósítása (algoritmusok, optimalizálás).

Módszerek:

- alulról felfelé,
- felülről lefelé,
- alrendszer-izolációval alrendszerenként,
- terv alapján automatikus kódgenerálással.

13.1.4. Ellenőrzések, tesztelés

13.1.2. Definíció (Verifikáció). *A verifikáció során azt ellenőrizzük, hogy a program megfelel-e a specifikációnak.*

13.1.3. Definíció (Validáció). *A validáció során azt ellenőrizzük, hogy a program megfelel-e a felhasználó által támasztott követelményeknek, és teljesíti-e az előírt minőségi tulajdonságokat (pl. rendszerigény, stabilitás, ...).*

13.1.4. Definíció (Egységteszt). *A program egy részegységének ellenőrzése izolált környezetben, a specifikációval összevetve.*

13.1.5. Definíció (Rendszerteszt). *A rendszer egésze működésének ellenőrzése, az egységek kapcsolatrendszerének és együttműködésének helyessége, stabilitása.*

13.1.6. Definíció (Fekete doboz tesztelés). *A tesztadatok elkészítésénél nem használunk fel a program műkdésére, megvalósítására vonatkozó ismereteket. Célja a hibás működések felfedése.*

13.1.7. Definíció (Fehér doboz tesztelés). *A tesztelés során ismerjük a program megvalósításának részleteit, ez alapján a tesztesetek hivatottak lehetőleg minden működési útvonalat ellenőrizni. Segítségével az esetleges hibák lokalizálhatók.*

13.1.5. Karbantartás

A karbantartás három nagy feladata:

- rejtett hibák javítása,
- adaptáció (áthelyezés új üzemeltetési környezetbe),
- továbbfejlesztés.

13.2. Az objektumelvű tervezés

13.2.1. Kialakulás

Történetileg a procedurális tervezés jelent meg előbb. Ennek során a feladatot egy (a bemenő adatokból a kimenő adatokat előállító) leképezésnek tekintjük, és a leképezés kiszámításának módját egyre kisebb *részműveletek*re bontjuk. Ennek során olyan műveletek keletkez(het)nek, melyek közös erőforrásokat, adatokat használnak. Ezen eljárásokat egy *programegység*be (modulba) összefogva csökken a rejtett hibák lehetősége. Ezután *absztrakció* alkalmazásával az erőforrásokhoz szabályos, általános hozzáférési felületet készíthetünk, melynek segítségével az egyes modulok közötti függőségeket csökkentjük.

13.2.1. Definíció (Objektumelvű programozás). *Az objektumelvű programozást támogató programozási eszköztől elvárjuk, hogy lehetőség legyen benne:*

- *absztrakt adattípusok és*
- *típusöröklődés*

megvalósítására.

13.2.2. Típusöröklődés

13.2.2. Definíció (Típusosztály).

1. *paraméterek (a paraméterek tulajdonságai),*
2. *export (a típusobjektumok halmazának és a rajta értelmezett műveleteknek a leírása, vonatkozó korlátozások),*
3. *import (más osztályoktól átvett szolgáltatások),*
4. *törzs (a típusosztály ábrázolása és megvalósítása).*

Típusöröklődés specializációval

Specializáció esetén a leszármazott:

- átveszi az ősosztály minden absztrakt tulajdonságát (*export*),
- az absztrakt tulajdonságok a származtatás során átnevezhetők és átdefiniálhatók,
- módosulhat az osztály *import* része,
- új ábrázolási formák és megvalósítások is felléphetnek,

A specializáció következményei:

Polimorfizmus: a leszármazott osztály példánya használható úgy, mint ha az ősosztály példánya lenne (*statikus* és *dinamikus* típus).

Dinamikus összekapcsolás: a végrehajtás során a dinamikus típus szerinti megvalósítás hozzárendelése a függvényhez, attribútumhoz.

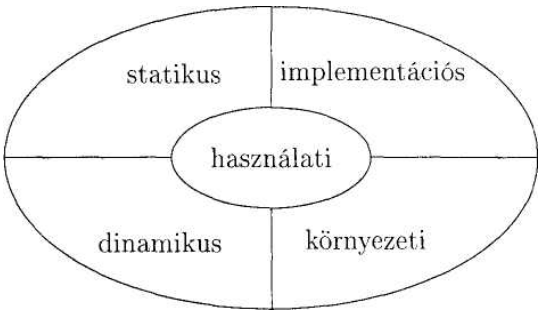
13.2.3. Típusöröklődés újrafelhasználással

Újrafelhasználás esetén a leszármazott:

- átveszi az ősosztály minden absztrakt tulajdonságát (*export*), és azokat a *törzs*ében használja fel,
- az absztrakt tulajdonságok a származtatás során átnevezhetők és átdefiniálhatók,
- az átvett műveletek jelentése *nem* változhat meg.

13.2.4. Nézetrendszerek

Az objektumelvű modellalkotás nézetrendszere:



13.2. ábra. Az objektumelvű modellalkotás nézetrendszere

Az ábrán látható szempontok magyarázata:

- *Használati szempont:* kinek nyújt szolgáltatást a rendszer (személyek, más rendszerek)? teljesülnek-e a felhasználó által elvárt követelmények?
- *Szerkezeti (strukturális, statikus) szempont:* a rendszer milyen egységekből épül fel? ezeknek mi a feladata? a részegységek milyen kapcsolatban állnak egymással?
- *Dinamikus szempont:* a rendszer részegységei hogyan viselkednek a megoldás során? milyen állapotaik lehetnek, és ezeket hogyan változtatják? milyen az egységek közötti együttműködés mechanizmusa?
- *Implementációs szempont:* milyen szoftverkomponensek vesznek részt a megoldásban? ezek között milyen kapcsolat áll fenn?

- *Környezeti szempont:* milyen hardver- és szoftver-erőforrások szükségesek a rendszer működéséhez?

13.3. UML eszközök, az objektumelvű tervezés támogatása

Az UML (*Unified Modelling Language*) egy olyan, szabvány által rögzített grafikus leíró nyelv, mely objektumelvű szoftverrendszerek tervezését hivatott támogatni. Eszközei lehetőséget adnak a probléma *specifikációjára*, *megoldására* és a megoldás *dokumentálására*. Sok eszköz támogatja továbbá az UML-leírásból valamilyen programozási nyelven készült forrásszöveg automatikus előállítását.

13.3.1. Az UML alapelemei

Az UML leírás *reláció*kból, *elem*ekből és *diagram*okból áll.

Elemek:
- strukturális elemek (osztály, objektum, ...),
- megnyilvánulási elemek (művelet, interakció, állapotautomata, ...),
- csoport-elemek (alrendszer, csomag, ...),
- annotációs elemek (megjegyzések, megszorítások, ...).

Relációk:
- függőségi reláció (szemantikai összefüggést jelöl),
- társítási reláció (szerkezeti összefüggést jelöl),
- általánosítási reláció (általános-speciális kapcsolatot jelöl),
- megvalósítási reláció (szemantikai kapcsolatot jelöl egy fogalom és annak megvalósítója között).

A diagramok elemeket és köztük lévő kapcsolatokat tartalmazó ábrák, úgymint:

- osztálydiagram,
- objektumdiagram,
- állapotdiagram,
- szekvenciadiagram,
- együttműködési diagram,

- aktivációs diagram,
- komponensdiagram,
- alrendszerdiagram,
- konfigurációs diagram,
- használati esetek diagramja.

13.4. Tervminták

A tervminták a különböző programrendszerek tervezése során sokszor felhasználható, általános terv-elemek.

Egy tervminta az alábbi részekből áll:

- név (általában utal a minta funkciójára),
- feladat (milyen esetekbe alkalmazható a minta),
- megoldás (a minta részletes leírása),
- következmények (a minta eredményei és a meghozott kompromisszumok leírása).

Keretrendszerek. A keretrendszerek (*framework*) a tervmintákkal rokon, de azoknál specializáltabb, kevésbé absztrakt (konkrét programozási nyelven megvalósított), nagyobb (sok osztályt és azok kapcsolatait magában foglaló) tervezési egységek.