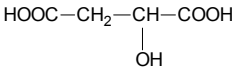
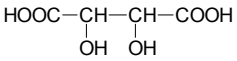


almasav(2-hidroxiborostyánkősav)



borkősav(2,3-dihidroxibutándisav)

Hasznos a savmolekulákból származtatott csoportok megnevezésének helyes ismerete. A karbonsavak karboxilcsoportjából kétféleképpen képezhetünk savszármazékot:

a) a karboxilcsoportról a HO- csoport eltávolításával acilszármazékot kapva. Megnevezésükkor a sav utótag helyett -oil végződést használunk:

CH_3-$[CH_2]_4$-CO- hexanoil- -OC-$[CH_2]_8$-CO- dekándioil-

A triviális nevek esetében az -oil vagy -il végződéseket, a karbonsav utótaggal megnevezett savak származékainál karbonil utótagot használunk:

CH_3-CO- acetil- -OC-CH_2-CO- malonil- C_6H_{11}-CO- ciklohexánkarbonil-

b) a karboxilcsoportról H- atom eltávolításával savmaradékot kapunk, amelyet a sav nevéből a sav szó elhagyásával és az –oát, vagy -át végződéssel nevezünk meg:

CH_3-$[CH_2]_4$-COO- - hexanoát CH_3-COO- -acetát

Felhasznált irodalom

1] *Útmutató a szerves vegyületek IUPAC-nevezéktanához,* (Nyitrai József, Nagy József szerkesztők, Magyar Kémikusok Egyesülete, Bp. 1998)

Máthé Enikő

Alkalmazások tervezése

Az alkalmazások általában nagyobb terjedelmű munkák, és legtöbb esetben egy adott alkalmazási területhez tartoznak. Minden területnek megvannak a saját – informatikától nem függő – szabályai, amelyeket betartva vagy feljavítva tudunk létrehozni egy alkalmazást.

Példák alkalmazási területekre:

- programozási környezetek
- DTP (DeskTop Publishing – kiadványszerkesztés)
- gazdaság: *általános gazdasági alkalmazás* – sok egyéni felhasználó számára készül (pl. könyvelés, raktározás, fizetésszámolás stb.); *sajátos gazdasági alkalmazás* – egyéni megrendelésre készül (pl. vállalatvezetés)
- oktatás: *egyedülálló oktatóprogram* – didaktikai jellegű; *társított oktatóprogram* (pl. *WINDOWS* alatti Help rendszerek)
- alkalmazástervező környezetek (vizuális tervező, forrásszöveg generátor, CASE (Computer Aided Software Engineering)
- mérnöki tervezőrendszerek – CAD (Computer Aided Design), stb.

Általában két kategóriájú alkalmazást különböztetünk meg: *kérésre írt* vagy *egyedi*, illetve *tömegfelhasználásra írt alkalmazást*.

Mindkét alkalmazástípus fejlesztése a következő alapfolyamatokat igényli:

- célkitűzés megfogalmazása
- adatgyűjtés és specifikáció megfogalmazása
- analízis, elemzés, elemzési dokumentáció
- tervezés, tervezési dokumentáció
- programozás, kódolás, telepítőrendszer írása, kódolási dokumentáció és megjegyzések a forrásszövegben
- tesztelés, tesztelési napló
- felhasználói kézikönyv, help megírása
- telepítés, betanítás, oktatás
- karbantartás és aktualizálás, minőségbiztosítás, felülvizsgálat

Az alkalmazások méretét a programozók számához, a befektetett munkaidőhöz és a forráskód hosszához (sorok száma) szoktuk hasonlítani. Ilyen értelemben egy lehetséges osztályozás a következő:

Osztály	*Programozók*	*Időtartam*	*Sorok száma*
egyszerű	1	1–4 hét	500
kis	1	1–6 hónap	1000–2000
közepes	2–5	1–2 év	5000–50 000
nagy	5–20	2–3 év	50 000–100 000
nagyon nagy	100–1000	4–5 év	1 000 000
óriási	2000–10 000	5–10 év	1 000 000–10 000 000

Egyedi alkalmazások tervezése

Egyedi alkalmazások tervezése esetén a kiindulópont két tárgyalófél közös munkájának az eredménye, egyik fél a *megrendelő (kliens)*, a másik fél a *munkavégző*. Ahhoz, hogy a tárgyalás eredményes legyen, a következőkre van szükség:

- A megrendelő intézmény nevezzen ki legalább egy embert, aki tudja, hogy az intézménynek mire van szüksége, és képes ezt a kérést érthető formában a munkavállaló elé terjeszteni.
- A munkavállaló olyan embert (akár többet is) küldjön ki a munka felmérésére, aki képes egy adott gyakorlati területbe nagyon hamar betanulni, képes nagyon jól elvonatkoztatni és átlátni, mert általában ettől az embertől függ a munka időtartamának felbecsülése, az informatikus munkacsoport méretének eldöntése, a programozási módszer meghatározása, amelyek nagymértékben befolyásolják a terv költségvetését.
- A munkavállaló ismerje meg, (ha szükséges), azt a gyakorlati folyamatot, amelyet neki kell átlátni. A megrendelővel folytatott eszmecserének nem szabad egy- vagy kétszeri alkalomra szűkülnie. Nem indíthatunk egy háromnégy hónapos útra egy tucat programozót és egy vagy két tervezőt úgy, hogy az átadáskor derüljön ki, hogy a megrendelő nem is azt vagy nem is úgy akarta, ahogyan azt a programozók megoldották.
- A programozókat a munka minőségi igényeinek megfelelően kell kiválasztani, mert munka közben bejöhet egy bonyolultabb megrendelés, és ha a legjobb programozók foglaltak valami egyszerű, de hosszadalmas feladattal, akkor ez egyértelműen a munkaerő nem optimális kihasználásához vezet.

- Miután tisztázódott, hogy mit tartalmaz a megrendelés, a kiküldött ember vagy csoport visszatér a specifikációhoz szükséges kész adatokkal. Amennyiben ez nem lehetséges, adatgyűjtés közben már megkezdik az elemzést. Az analízis az a folyamat, amelyben több lehetséges megoldás közül kiválasztjuk a mindkét fél számára legelőnyösebb változatot. A tervező feladata megszervezni, a programozó feladata pedig beosztani a munkát a kért terminusok közé.
- Vigyázzunk, hogy a specifikációkról mindkét félnél hiteles dokumentum maradjon, az utólagos viták elkerülése végett. Személyzetcserét a specifikációgyűjtő illetve szolgáltató csoportban csak szükség esetén végezzünk.
- Az alkalmazás tervezése egy vagy több személy elvonatkoztató képességén alapszik, amelynek eredményeképpen összeáll a képernyőformátumokkal, algoritmusokkal, általános változónevekkel, típusdefiníciókkal, nyomtatási eredmények formáival tűzdelt tervezési dokumentáció, amelyből nem hiányozhat az illető alkalmazási terület szakkifejezéseinek értelmezése. Ez az alapdokumentum fogja végigkísérni a munkát egészen az átadásig, sőt még azután is fontos lesz az esetleges módosításokhoz szükséges információk miatt.
- A programozás egy programozási dokumentációt igényel, amelynek két fontos célja van: 1. a programozási munkában résztvevő programozók munkaviszonyának megszűnése nem érintheti a munkaadó által vállalt kötelezettségeket. 2. az utólagos módosítások legyenek megvalósíthatók más programozókkal is. Ez a dokumentáció a használt programozási elemek (általános változók, más fontosabb változók, eljárások, függvények, típusdefiníciók, osztályhierarchiák, egységek, fontosabb algoritmusok, esetleg adatbázisok) leírásából, valamint a forrásszövegben levő megjegyzésekből áll. Fontos megnevezni minden programozási elem helyét a tervben és minden tervelem helyét a programban, különben ha a kért eredmények nem felelnek meg az elvárásoknak, a logikai hibák keresése nagyon hosszadalmas lesz. A program jobb megértésére szolgál, ha beszélő neveket használunk változók, függvények, eljárások és adatbázismezők azonosítóiként. Megpróbáljuk minél kisebb struktúrákra bontani a programot, és kell legyen legalább egy általános rajz a programozási dokumentációban, amelyiken minden fontosabb elem fel van tüntetve.
- A tesztelést végrehajthatja (1.) a programozó, ő csak azt fogja észrevenni, hogy jó adatokra jó-e az eredmény, (2.) a szakember a megrendelő részéről, aki szakmai szempontból fogja az összes lehetséges algoritmuságat tesztelni, de csak azokat a logikai hibákat fogja megtalálni, amelyek ezekben az algoritmusokban vannak, (3.) a titkárnő, aki semmit sem ért az egészhez, csak pont őt tették oda, hogy valami papírokról valami számokat a számítógépbe beüssön, ő a rosszul bevitt adatokból származó hibákat és elég sok működési hibát fog megtalálni.
- A felhasználáshoz szükséges egy írott dokumentáció, amelyet felhasználói dokumentációnak (kézikönyvnek) szoktunk nevezni, és amely a felhasználót segíti az alkalmazás funkcionális működésének megértésében.
- Az utolsó és leghosszabb fázis a karbantartás és aktualizálás. Ebben a fázisban háromféle módosítás fordul elő: hibajavítás, törvénykezésből származó változások megvalósítása, valamint hardver vagy szoftver elemek miatti korszerűsítések. Fontos, hogy ez a fázis egy előre megkötött szerződés alapján

történjen, mert ha nem, vagy a felhasználó tartja túl magasnak a pillanatnyi beavatkozás árát, vagy a munkavállaló fogja alacsonynak tartani a karbantartással eltöltött időért kapott összeget.

Tömegesen használt alkalmazások tervezése

A széleskörű felhasználásra tervezett alkalmazások létrehozási folyamata nem tér el nagymértékben az egyedi alkalmazások tervezési folyamatától, de mégis van néhány említésre méltó különbség:

- A specifikáció megfogalmazása piackutatás eredménye (pl. a COREL Corporation egy évig piackutatással foglalkozott, mielőtt megalkotta volna a COREL DRAW 1.0-t). A piackutatás magas költségei miatt csak bizonyos cégek valósíthatnak meg nyereséggel, tömegesen használt alkalmazásokat. Egy komoly piacfelmérés meghatározza a kereslet szintjét, mert ettől függ a termék ára; a felhasználók igényességét, mert ez határozza meg az interfész megírására szánt pénzmennyiséget; a létező konkurencia alkalmazásainak erősségeit, ez határozza meg a kompatibilitás megvalósítására szánt pénzmennyiséget; a létező konkurencia alkalmazásainak gyenge pontjait, ez határozza meg a reklámunk jelmondatát; az alkalmazási terület összes hiányosságát, ezek pótlásának mértékétől függ, hogy egy új verziót vagy csak egy alverziót valósítsunk meg.
- A tervezés lehet egy- vagy többcsoportos párhuzamos tervezés (nem szabad egy pár emberre bízni a tervezést és kivitelezést, amikor 2 000 000 ember fogja használni a terméket).
- A termék lehet: (1.) próbálkozás egy új alkalmazási terület kialakításáért, (2.) egy meglévő alkalmazási terület kibővítése a konkurensek lépéseire való reagálásképpen, (3.) egy saját termék javítása, amikor a konkurensek nem jönnek számításba, de a cégnek vannak új ötletei, és megvalósításuk által újabb jövedelemre szeretne szert tenni.
- Az alkalmazás tesztelése lehet belső, de ez eléggé gyenge minőségű szokott lenni, ezért inkább a nagyközönséget kérik fel külső tesztelésre, *alfa-* illetve *bétatesztek* segítségével, ezzel a módszerrel biztosítják a vásárlókör elég jelentős részét, ugyanis a tesztelők nagy része meg is vásárolja a terméket.
- A felhasználó dokumentálása a help rendszer, amely rendelésre készült alkalmazásoknál rendszerint hiányzik, valamint a felhasználói útmutatás vagy dokumentáció, ami nagyon részletes.
- Amennyiben egy létező alkalmazás új verziójáról van szó, a már meglévő vevők munkájának tiszteletben tartása végett fontos az előző verziókkal való adatformátum-kompatibilitás megvalósítása. Ez a kompatibilitás lehet teljes, a formátum megtartásával, vagy csak írás, olvasás szintű (konverzió).
- Amennyiben egy kidolgozott alkalmazási területre akarunk betörni egy új termékkel, fontos a konkurencia termékeivel való adat- és kezelési kompatibilitás.

Kovács Lehel