SZABADKAI MŰSZAKI SZAKFŐISKOLA
SZABADKA

OBJEKTUMORIENTÁLT TERVEZÉS

beszámoló
Software Engineering tárgyból

Tanár: Dr. Simon János

hallgató: Kovács Róbert
leckekönyv: 16218119
szakirány: Műszaki informatika

Szabadka, 2020/21

Tartalom

Objektumorientált Tervezés

A szoftvertermékek fejlesztési tevékenységeit a fejlesztés során alkalmazott módszertan határozza meg, amely szorosan illeszkedik a fejlesztést végző szervezet felépítéséhez, a szervezeten belüli emberek képességeihez és a fejlesztendő rendszer jellemzőihez. A szoftveriparban ennek köszönhetően számos fejlesztési módszertan alakult ki. A kifejlődött módszertanok többségére elmondható, hogy az objektum-orientált fejlesztési elvre épül, amelyben a kifejlesztendő rendszert együttműködő objektumokként modellezzük a fejlesztés minden fázisában.

Az objektumorientáltság egy szemléletmód, amelyben a valós világ dolgait egymással kapcsolatban lévő objektumoknak tekintjük, mint pl. emberek, állatok, városok, épületek stb. Objektumként definiálható általában bármi, ami egyértelműen behatárolható. Az általánosan alkalmazott definíció szerint az objektum egy valós vagy absztrakt entitás (egyed), amely a vizsgált rendszer egy jól azonosítható szereplője. Egy objektum más objektumokra hatást gyakorolhat és más objektumok hatással lehetnek rá. Az objektum egy belső struktúrával rendelkezik és minden időpillanatban egy meghatározott állapottal rendelkezik, amelyet a rajta értelmezett és végzett műveletek befolyásolnak. Az állapotot az objektum attribútumainak halmazaként adjuk meg. Az objektum metódusai pedig szolgáltatásokat biztosítanak a többi objektum számára, amelyek ezekre a szolgáltatásokra bizonyos tevékenységek elvégzése érdekében igényt tarthatnak.

A külső szemlélő (többi szereplő) nem lát bele az objektumba, a struktúrára és az állapotra csak a viselkedésből következtethet. Ennek megfelelően azt sem látja, hogy a viselkedés során milyen az objektum belső működése. Ez az információelrejtés elvének következetes alkalmazása, amit egységbe zárásnak (encapsulation) nevezünk. Az objektum egységbe zárja az állapotát tároló adatokat és azok szerkezetét, valamint a rajtuk végrehajtott, az objektum viselkedését meghatározó műveleteket. Az objektum tervezőjének feladata a belső struktúra, a lehetséges állapotok kialakítása és a viselkedés programozása.

A megegyező viselkedésű és struktúrájú objektumokat egy közös objektumosztályba (class) soroljuk. Az objektumosztály az azonos viselkedésű és struktúrájú objektumok forrásának tekinthető. Az egy objektumosztályba tartozó objektumok az állapotukban különbözhetnek. Ha szűkebb értelemben az objektumorientált programfejlesztést tekintjük, akkor az objektumosztály egyszerre egy típusspecifikációnak és egy objektumok létrehozására szolgáló sablonnak tekinthető, amely az adott osztályba tartozó objektummal kapcsolatos összes attribútum és művelet deklarációját tartalmazza. Egy konkrét objektumot az objektumosztály egy példányának (instance) is szokás nevezni.

Az objektum tulajdonságait és állapotát meghatározó, az objektumban tárolt adatok az attribútumok. Az attribútumok értékei az objektum élete során változhatnak. Az attribútumok száma és típusa definiálja az objektum struktúráját, vagyis azt, hogy milyen belső változói vannak. Minden attribútum egy, az osztályra nézve egyedi névvel rendelkezik. Az objektum-orientált programozásban az attribútumok változóknak tekinthetők, amelyek típusuknak megfelelő értéket vehetnek fel. Az attribútumok lehetnek egyszerű vagy összetett típusúak, közvetlen adatok vagy referenciák (mutatók, pointerek). Az attribútumok lehetnek osztályváltozók is, azaz olyan változók, amelyek egy objektum példány tárolására alkalmasak. Mivel az objektumot a neki küldött üzenetekkel érhetjük el, az objektum változónak küldött üzenetet az általa éppen tartalmazott objektum kapja meg.

Az objektum viselkedése az általa végrehajtott tevékenységsorozatban nyilvánul meg, a rendszer működése pedig az objektumok kölcsönhatásaként. Viselkedésének jellegét tekintve egy objektum lehet aktív vagy passzív. A passzív objektum mindaddig nem tesz semmit, amíg valamilyen környezeti hatás nem éri, például üzenetet nem kap egy másik objektumtól. Az aktív objektum folyamatosan működik, valamilyen tevékenységet hajt végre, aminek során más objektumokat is működésre bír, azokra hatást gyakorol.

Az objektumok egymásnak küldött üzeneteken (metódus hívásokon) keresztül lépnek kölcsönhatásba egymással. Az üzenet az objektumok közötti adatcsere eszköze, másrészt az objektumok működésének vezérlésére szolgáló eszköz. Egy üzenet két fontos komponenssel rendelkezik: egyrészt van neve, másrészt vannak paraméterei, amelyeket az üzenet aktuális tartalmának is tekinthetünk. A paraméterek alkalmasak a működés közben keletkező dinamikus információk átadására.

Az objektumhoz érkező üzenet hatására az objektum valamilyen cselekvést hajt végre. Ha egy objektum képes fogadni egy adott nevű üzenetet, akkor erre az üzenetre az üzenet neve által meghatározott metódusa végrehajtásával reagál. Az objektum viselkedésének pontos leírása, implementációja a metódusainak kódjában található.

Az objektumokban tehát egyszerre megjelenik a viselkedés és az állapotinformáció. Az objektum által végrehajtott metódus megváltoztathatja az objektum belső állapotát. Az objektum állapotváltozóit kívülről közvetlenül nem láthatjuk, így nem is érhetjük el azokat. Az objektum így az információ elrejtése egyik eszközének tekinthető.

Egy objektumorientált rendszer saját állapotukat karbantartó és erről az állapotról információs műveleteket biztosító, egymással együttműködő objektumokból áll. Az állapot reprezentációja privát, az objektumon kívülről közvetlenül nem hozzáférhető. Egy objektumorientált tervezési folyamat az objektumosztályoknak és az azok közötti kapcsolatoknak a megtervezéséből áll. Ezek az osztályok definiálják a rendszer objektumait és a közöttük lévő interakciókat. Ha egy futó program megvalósít egy tervet, akkor a szükséges objektumok az osztálydefiníciók alapján dinamikusan jönnek létre.

Az objektumorientált tervezés az objektumorientált fejlesztés része, amelyben fejlesztési folyamat során objektumorientált stratégiát használunk:

1. Az objektumorientált elemzés az alkalmazás objektumorientált szakterületi modelljének kialakításával foglalkozik. A kifejlesztett objektum modell objektumai a megoldandó problémával kapcsolatos egyedeket és műveleteiket tükrözik.

2. Az objektumorientált tervezés a meghatározott követelményeknek megfelelő szoftverrendszer objektumorientált tervezési, vagy implementációs modelljének kialakítása.

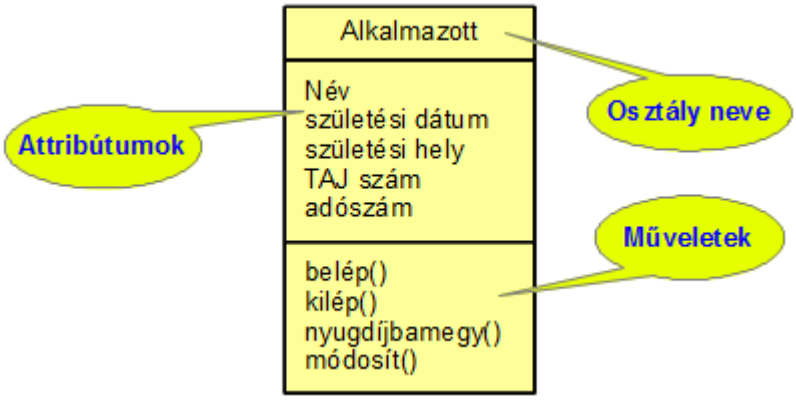
3. Az objektumorientált programozás a szoftverterv objektumorientált programozási nyelven történő megvalósítása. Egy objektumorientált programozási nyelv biztosítja az objektumosztályok és az objektumok ezen osztályokból történő létrehozását.

Hogy a lépések közötti átmenet biztosított legyen, a fenti lépések mindegyikében egységes jelölésrendszert kell használni. A fejlesztés következő szakaszba lépése maga után vonja az előző szint finomítását a már létező objektumosztályok részletezésével és a további funkciókat biztosító új osztályok hozzáadásával.

Az objektumorientált rendszereknek a más elven fejlesztett rendszerekkel szemben több előnyük is lehet, pl. a változtatások végrehajtása ezen rendszerekkel könnyebb lehet. Az objektumok egymástól függetlenek, különálló entitásokként foghatók fel és módosíthatók. Egy objektum implementációjának megváltozása vagy új szolgáltatásokkal történő bővülése nem befolyásolhatja a rendszer többi objektumát. Az objektumok újrafelhasználható komponensek, mivel az állapotnak és a műveleteknek független egységbe zárásai. A tervek fejlesztése kiindulásként az előző tervekben létrehozott objektumok felhasználásával kezdhetők el. Ez csökkenti a tervezés, a programozás és az ellenőrzés költségeit, valamint elvezethet a szabványos objektumok használatához és csökkenti a szoftverfejlesztésben meglévő kockázatot.

UML (Unified Modeling Language)

Az UML-ben mind az osztályt, mind pedig az objektumot ábrázoló grafikai elem egy három részre osztott, névvel ellátott téglalap: a felső rész tartalmazza az osztály és/vagy objektum nevét, a középső részben az attribútumok neve vagy azok értéke található, az alsó részben pedig a metódusok vagy műveletek felsorolása szerepel. Amennyiben az attribútumokat vagy metódusokat nem akarjuk jelölni a középső vagy az alsó, vagy akár mindkét téglalap elhagyható.



Ábra 1: Osztály ábrázolása

Az attribútumok elnevezése mellett megadhatjuk azoknak típusát és az előre definiált kezdőértéket is. A metódus nevének megadását követheti a zárójelbe tett paraméterlista, valamint a metódus által szolgáltatott eredmény típusa. A metódusok és a programozási nyelvek függvényei közötti hasonlatosság szándékos, mivel az objektum metódusait általában függvényekkel implementáljuk.

Vannak olyan rendszerek, amelyben különböző objektumok is rendelkezhetnek azonos nevű metódussal, így az üzenetre (metódus hívásra) adott választ az objektum viselkedése határozza meg, azaz a kérdéses metódus implementációja. Ezt az objektum-orientált fejlesztésben polimorfizmusnak nevezzük. A polimorfizmus kifejezés magyarul többalakúságot jelent. A polimorfizmus a gyakorlatban, az osztályhierarchiák kialakítása során az öröklődéssel jelenik meg. Az 2. ábra által mutatott példában az egyes osztályok az azonos nevű Terület() metódussal rendelkeznek. A polimorfizmus abban nyilvánul meg, hogy mindegyik osztály esetében a síkidom területének számítását más-más összefüggés alapján implementáljuk.

Ábra 2: Osztályok közötti polimorfizmus

Az objektumok úgy kommunikálnak, hogy szolgáltatásokat kérnek más objektumoktól (meghívják azok módszereit), valamint szükség esetén kicserélik egymás között a szolgáltatás ellátásához szükséges információkat. A szolgáltatás végrehajtásához szükséges információ és a szolgáltatás végrehajtásának eredményei paraméterként adódnak át.

Az objektumokkal való modellezés célja, hogy a valós dolgokat egymással kölcsönhatásban lévő objektumokkal írjuk le. Egy számítógépes rendszerről, egy alkalmazásról számos modell készíthető, amelyek a rendszert különböző nézőpontból írják le.

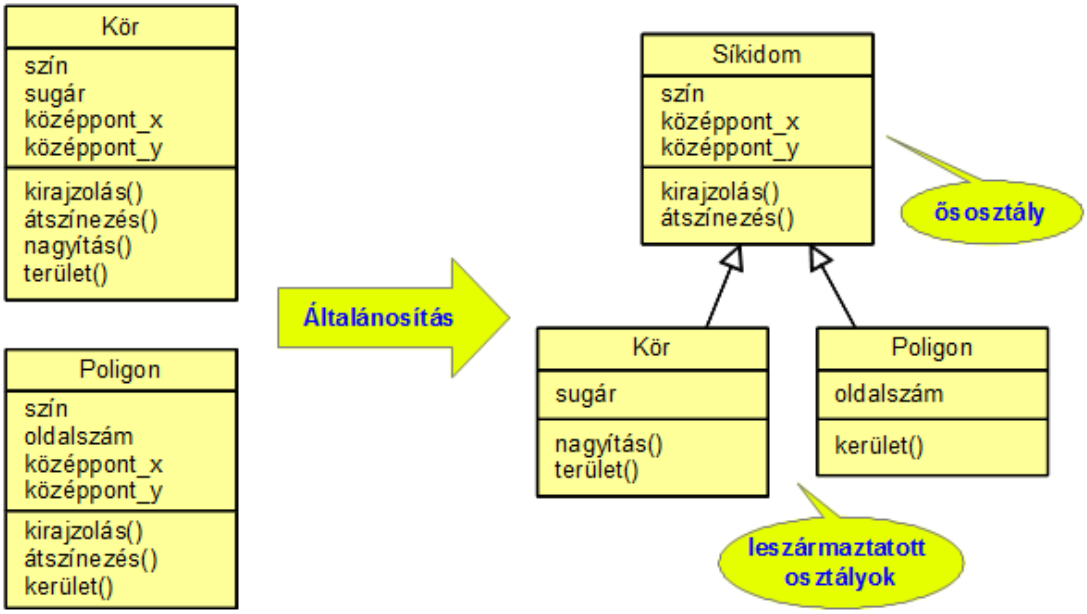
Az úgynevezett objektummodellek alkalmazásával írjuk le a rendszerbeli objektumok struktúráit, azok attribútumait és metódusait, valamint az objektumok közötti kapcsolatokat. Az objektummodell kialakításával az a célunk, hogy rögzítsük az alkalmazási területnek a feladat szempontjából lényeges fogalmait és a rendszer statikus viszonyait. Az objektumokat és az objektumok közötti kapcsolatokat jeleníti meg az osztálydiagram.

Öröklés

Az öröklés olyan implementációs és modellezési eszköz, amelyik lehetővé teszi, hogy egy osztályból olyan újabb osztályokat származtassunk, amelyek rendelkeznek az eredeti osztályban már definiált tulajdonságokkal (attribútumokkal), szerkezettel és viselkedéssel (metódusokkal).

Azt az osztályt, amelyből örökölünk, ősosztálynak nevezzük. Az az osztály, amelyik örökli a struktúrát és a viselkedést, a származtatott osztály. Az öröklés mögött álló relációt általánosításnak, vagy specializációnak nevezzük. Egy származtatott osztályban az öröklött attribútumok és metódusok halmaza nemcsak bővíthető, hanem az öröklött metódusok át is definiálhatóak. Az új definíció finomítja vagy helyettesíti az öröklött metódust.

Az objektumosztályok az öröklődésnek megfelelően egy öröklődési hierarchiába szervezhetők, amely az általános és a specifikus objektumosztályok közötti kapcsolatot jeleníti meg. A specifikusabb objektumosztályok teljesen konzisztensek az általános objektumosztályokkal, de a specializációnak megfelelően további információt tartalmaznak.



Ábra 3: Öröklődési hierarchia kialakítása.

A 3. ábra egy öröklési hierarchia kialakítását mutatja be. Az ábra bal oldalán szereplő két osztály, a Kör és Poligon, tartalmaz azonos attribútumokat és metódusokat. Az általánosítás során a két osztály közös attribútumait és metódusait kiemeljük és az általánosításként létrehozott Síkidom nevű osztályhoz rendeljük. Az öröklési viszonyt az üres háromszög fejű nyilakkal fejezzük ki. A hierarchiában a Síkidom tölti be az ősosztály szerepét, míg a Kör és Poligon osztályok lesznek a leszármaztatott osztályok. Ezek az osztályok öröklik az ősosztály attribútumait és metódusait.

Az általánosítás relációt gyakran olyan alaposztályok létrehozására használjuk fel, amelyek célja kizárólagosan az, hogy az öröklés révén átadják az attribútumaikat és a metódusaikat a származtatott osztályoknak. Az ilyen osztályokat absztrakt osztályoknak nevezzük. Ezen osztályokból nem képezünk példányokat, azaz nem képződik belőlük objektum, csak osztályokat származtatunk belőle.

Az objektum-orientált tervezési folyamat

A szoftverfejlesztési módszertan a szoftverkészítés meghatározott technikákat és leírási módokat alkalmazó szervezett folyamata. A módszertan végrehajtandó lépések sorozataként jelenik meg, amelyben minden lépéshez definiáljuk az alkalmazandó technikát és jelölésrendszert. A fejlesztés főbb fázisai a követelményelemzés, elemzés (analízis), tervezés, implementáció és tesztelés.

Az elemzési fázis célja az alkalmazási szakterületének megértése és modellezése az adott terület fogalmainak felhasználásával, azaz a fogalmi (elemzési) modell kialakítása. Az elemzési fázis bemenete a követelmény specifikáció, amely leírja a megoldandó problémát és körvonalazza a kialakítandó rendszer céljait és funkcióit. Az elemzés során az adott alkalmazási terület szakértőivel és a megrendelővel való folyamatos konzultáció révén pontosítjuk a kiinduló információkat, célokat.

A fogalmi modellhez tartozó, neki megfeleltethető legkedvezőbb implementációs modell létrehozása a tervezés feladata. A tervezés során az elemzési modelleket leképezzük a rendelkezésre álló hardver és szoftverelemek szolgáltatásaira. A rendszer globális struktúrájának kialakítása az architekturális tervezés feladata. Az objektum modellből kiindulva a rendszert alrendszerekre bontjuk, és ezzel összefüggésben kialakítjuk a dinamikus modellek implementálásának elveit. A tervezés második fázisa, az objektumtervezés, során a fogalmi modellről az implementációs modellre tevődik át a hangsúly. Kiválasztjuk a fő funkciókat realizáló algoritmusokat, majd az algoritmusok alapján kialakítjuk a hatékonyan implementálható változatot. Az architekturális tervezéskor kialakított vezérlési rendszer részletes terveit elkészítve, az alrendszereket modulokra bontjuk.

Elemzés

A követelményekből kiindulva az elemzési fázisban a valóság egy bizonyos absztrakciós szinttel rendelkező modelljét hozzuk létre. A modellalkotás során azonosítjuk a rendszer lényeges szakterületi jellemzőit. Az analízis eredményeként létrejött elemzési modell azt ábrázolja, hogy mit csinál a rendszer, függetlenül annak konkrét megvalósulásától.

Az elemzési modell az objektum, a dinamikus és a funkcionális modellekből, épül fel. Az elemzési modell több célt is szolgál. Egyrészről egyértelműen tisztázza a követelményeket a megrendelő és a fejlesztő számára, másrészről a tervezés és az implementáció is ebből a modellből indul ki.

A feladat definíció

Az analízis első, lényegében előkészítő lépése a feladatdefiníció, amelyben beszélt nyelven megfogalmazzuk, hogy a rendszerrel szemben milyen elvárásaink vannak, mik lesznek a rendszer fő funkciói, milyen alapvető fogalmakkal dolgozunk, és kik lesznek a rendszer felhasználói. Az analízis egyik kulcsproblémája a fejlesztők és a rendszer majdani felhasználóinak a korrekt és precíz kommunikációja. A feladatdefiníciónak a következő témákkal kell foglalkoznia:

1. a feladat körvonala, határai, a készítendő rendszer felhasználói
2. szükségletek, igények, elvárások
3. alkalmazási környezet
4. a rendszer alkalmazási környezetére vonatkozó feltételezések
5. teljesítmény, megbízhatóság, prioritások
6. fejlesztési előírások

A rendszer felhasználóinak azonosítását felhasználói szerepelemzésnek nevezzük. A különböző típusú felhasználók a rendszert különféleképpen használhatják, különböző elvárásokat is támaszthatnak vele szemben.

Az objektum-orientált analízis során az elkészült feladatdefiníció feldolgozásának első lépése általában a probléma szereplőinek (objektumok), és a rájuk ruházható felelősségeknek az összegyűjtése. Az első próbálkozásra kiadódó objektumokat tipizáljuk, azaz a hasonló viselkedésű szereplőket egy-egy csoportba osztjuk. Az egyszerű attribútumokat ezen objektum típusokhoz rendeljük, hasonlóképpen a műveleteket, funkciókat ugyancsak az egyes osztályok felelősségi körébe soroljuk. Ennek a lépésnek az eredménye a specifikációs táblázat, amely típusonként felsorolja a feladat szereplőit, illetve a szereplőknek a feladatdefinícióból adódó attribútumait és felelősségeit.

Objektummodellezés

A feladatdefiníciót követő lépés az objektummodell kidolgozása. Az objektummodell leírja a valóságos rendszer objektumait és az objektumok közötti kapcsolatokat. Az objektummodell kialakítása általában könnyebb, mint a dinamikus és funkcionális modelleké, ezért célszerű először ezzel a modellel kezdeni. Az objektummodell felépítésének viszonylagos egyszerűsége abból adódik, hogy a statikus struktúrák könnyebben értelmezhetők, jobban definiáltak és kevésbé függnek az alkalmazás részleteitől, mint a dinamikus jellemzők.

A objektummodell felállításához szükséges ismeretek elsődlegesen a feladatdefinícióból és az alkalmazási terület szakértőitől származnak.

A modellezés során elsőként az osztályokat és asszociációkat azonosítjuk, mivel ezek határozzák meg a rendszer általános struktúráját, majd ezekhez adjuk hozzá az attribútumokat. A következő lépésben az esetleges öröklési struktúrák felderítése következhet.

Objektumorientált tervezés

Az analízis a megvalósítandó rendszer egy lehetséges modelljét alakítja ki, elvonatkoztatva attól, hogy a modellben szereplő dolgokat az implementáció során milyen elemekkel képviseltetjük, illetve attól, hogy a tevékenységeket ki és milyen módon hajtja végre. Informatikai rendszerek kialakítása során a rendszer szereplőit, a végrehajtott tevékenységeket szoftver és hardver elemekkel fogjuk realizálni, ami azt jelenti, hogy az absztrakt modellt le kell képeznünk a fizikai rendszerünk által biztosított szolgáltatásokra. Ezt a leképzési folyamatot nevezzük tervezésnek, amely ily módon az implementáció előkészítése.

A tervezés három szintjét különböztetjük meg. Az architekturális tervezés során a létrehozandó programot, mint egészet érintő kérdésekben döntünk. A külső interfész tervezése a külvilággal történő kapcsolattartás módját és részleteit írja le. Az objektumtervezés célja a programot felépítő elemek, az osztályok és objektumok, egyenkénti részletes tervezése, specifikálása.

A tervezés, mint az elemzés és az implementáció közötti kapocs, nem csupán az elemzés termékeire épít, hanem figyelembe veszi az implementáció lehetőségeit is.

Forrás

Információs rendszerek tervezésének módszertana - Komló Csaba

http://moodle.autolab.uni-pannon.hu/