Egyetemi doktori (PhD) értekezés tézisei

Objektumorientált tervezési alapelvek és tervezési minták döntésalapú elemzése, a döntésösszevonás elmélete és gyakorlata

Decision Based Interpretation of Object-Oriented Design Principles and Design Patterns, Decision Contraction Theory with Practice

Márien Szabolcs

Témavezető: Dr. Terdik György

DEBRECENI EGYETEM
Informatikai Tudományok Doktori Iskola
Debrecen, 2011

TARTALOMJEGYZÉK

1. A DOKTORI ÉRTEKEZÉS ELŐZMÉNYEI ÉS CÉLKITŰZÉSEI 2

2. AZ ÉRTEKEZÉS ÚJ TUDOMÁNYOS EREDMÉNYEI 9

2.1 DÖNTÉSREDUNDANCIÁK MEGSZŰNTETÉSE DÖNTÉSÖSSZEVONÁSOKKAL 9
2.2 „ÖRÖKLÉST DÖNTÉSREDUNDANCIA MEGSZŰNTETÉSRE ALKALMAZZUNK!”, „DÖNTÉS-REDUNDANCIÁT KERÜLJÜK!” TERVEZÉSI ALAPELVEK BEVEZETÉSE 10
2.3 DÖNTÉSREDUNDANCIÁK MÉRÉSE ÚJ METRIKÁKKAL 11
2.4 TERVEZÉSI MINTÁK JELLEMZÉSE, OSZTÁLYOZÁSA DÖNTÉSSZERKEZETEK ALAPJÁN; TERVEZÉSI MINTÁK DÖNTÉSSZERKEZETEINEK JML ALAPÚ SPECIFIKÁCIÓJA 12

3. AZ ÉRTEKEZÉS TÉMÁJÁBAN SZÜLETETT PUBLIKÁCIÓK 14

4. AZ ÉRTEKEZÉS TÉMÁJÁVAL KAPCSOLATOS PROJEKT 15

5. ANTECEDENTS AND AIMS OF THE DOCTOR'S DISSERTATION 16

6. THE NEW SCIENTIFIC RESULTS OF THE DISSERTATION 22

6.1 DISSOLVING DECISION REDUNDANCIES BY DECISION-CONTRACTIONS 22
6.2 THE INITIATION OF THE „USING INHERITANCE TO DISSOLVE DECISION-REDUNDANCY”, „AVOID DECISION-REDUNDANCY” DESIGN PRINCIPLES 23
6.3 MEASURING THE DECISION-REDUNDANCIES WITH NEW METRICS 24
6.4 ANALYZING AND CLASSIFYING THE DESIGN PATTERNS BASED ON DECISION STRUCTURES 25

7. PUBLICATION IN THE TOPIC OF THE DISSERTATION 27

8. PROJECT IN CONNECTION WITH THE TOPIC OF THE DISSERTATION 28

1. A doktori értekezés előzményei és célkitűzései

Munkám során számos projektben vettem részt tervezőként és projektvezetőként. Tapasztalatom szerint a fejlesztések sikerességét nagyrészt a megfelelő tervezés és az arra épülő minőségbiztosítás garantálja. Minőségbiztosításon a tervezési és fejlesztési hibák kezelését, visszacsatolását értem.

A tervezés terméke a rendszerterv, modell, melynek minősége nehezen mérhető, a tervezési hibák sokszor túlkésőn derülnek ki, amikor már a sok függőség miatt a javítás költséges lenne. Tervezői munkám alatt gyakran észlelem, hogy a tervezők és fejlesztők a függőségek átláthatatlan szövevénye miatt nem is veszik észre, hogy a probléma, amivel szembesülnek az csak egy következmény, egy korábbi nem megfelelő tervezési megoldás tükröződése.

Az objektumorientált programozási módszertan célja az, hogy jól strukturált, és így átlátható, könnyen bővíthető, és így karbantartható; újra felhasználható, és így moduláris programok szülessenek. Ezen célok elérése, azaz megfelelően stabil és jól szervezett program létrehozása azonban nagyon sok tervezői és fejlesztői tapasztalatot igényel.

A célok elérése érdekében olyan általános szabályok születtek, amelyek az objektumorientált rendszerek minőségi kérdéskörének kulcspontjaként a csatoltság és függőség – implementációs függőség mértéket határozzák meg, amely magas foka esetén a rendszer karbantarthatatlanná válhat inkonzisztens modulokkal téve az adott rendszert instabillá.

Csatoltságon az egyes rendszerkomponensek kapcsolatainak erősségét értjük. Két komponens közötti csatoltság növekszik, ha egyik a másik valamilyen mezőjére utal, metódusát hívja vagy egy metódus visszatérési típusa a másik komponens típusa. A csatoltság legerősebb formája az öröklés. Laza csatolás esetén az osztályok módosítása nem eredményez jelentős változtatáskényszert a többi osztályban.

Az implementációs függőségek a csatoltság következményei, amelyek növelik az instabilitás kockázatát, csökkentik a karbantarthatóságot.

Az implementációs függőség csökkentése kritikus, mivel hozzájárul a karbantarthatóság, újra felhasználhatóság tekintetében kiemelt fontosságú laza csatolás kialakításához.

Laza csatolással elősegíthető autonóm rendszerkomponensek kiépítése, amelyek cserélhetősége, karbantarthatósága könnyebben megoldható. Laza csatolás nélkül a függőségek átláthatatlan rendszert eredményeznek, amelyek a karbantarthatóságot és későbbi felhasználhatóságot teszik lehetetlenné.
A tervezői munka nagy szabadságfoka, a megoldások tömegéből a legésszerűbb megoldás kiválasztása nagy kockázat, amely csak nagy tervezési tapasztalattal csökkenthető.
Célszerű tehát a tervezés szempontjából korábban bevált megoldásokat szabályokként megfogalmazni, és azok újrahasznosítását elősegíteni. Ezért különböző objektumorientált rendszerekben hasonló osztály- és objektumszerkezetű struktúrákat találhatunk, amelyekkel az újra és újra feltűnő hasonló tervezési problémákra adnak a szakemberek hasonló megoldásokat. Ezeket a visszatérő tervezési problémákra adott újrahasznosítható tervezési megoldásokat, általános érvényű „recepteket” hívjuk tervezési mintáknak, amelyek a minőségi terv és megvalósítás eszközei. Specifikusabban egymással együttműködő objektumok és osztályok leírásai, amelyek esetre szabottan valamilyen általános tervezési problémát oldanak meg. A tervezési minták a tervezési tapasztalatok összegyűjtésének eredményeként jöttek létre, melyek a tervezés során a tervezőknek tanácsot adnak, hogy miként tudják az adott tervezési helyzeteket megfelelően megoldani. Számos esetben hasonló, általánosítható tervezési problémával szembesülünk, amelyre érdemes szakmailag megvitatott és elfogadott válaszokat megfogalmazni, és azokat esetspecifikusan felhasználni.
A tervezési minták a tervezési alapelvek céljainak megfelelően esetspecifikusan, de kellőképpen absztrakt szinten fogalmazzák meg azokat a módszereket, melyek alkalmazásával az adott tervezési helyzetekben megfelelő minőségű objektumorientált felépítés jöhet létre.

A minőségbiztosítás kiemelt fontosságú, hiszen garanciákra van szükség, hogy az előálló termék megfelel az elvárásoknak, és stabilan képes nyújtani az elvárt működést. De a minőséget tudnunk kell mérni. A mérnöki szakma a létrehozott termékek minőségének jellemzésére mérőszámokat használ. Más mérnöki szakterületekkel ellentétben a szoftverfejlesztésben nincsenek a minőséget egyértelműen meghatározható mutatószámok, csupán indirekt

mérőszámokat definiálhatunk, amelyek valamilyen aspektusból teszi mérhetővé az adott szoftver minőségét.
Számos megoldás született annak mérésére, hogy az említett célok mennyire teljesülnek. A teljesülés méréséhez különböző metrikákat használnak, amelyek számos szempontból próbálják a strukturáltságot minősíteni.
A projektvezetőknek és a tervezőknek az ismert metrika és mérő eszközrendszerek közül nem egyértelmű a megfelelő metrikák kiválasztása, melyekkel a program minőségét a kívánt szempontok alapján mérni tudják. Ennek oka, hogy a mérhető tulajdonságok száma végtelen, ezért fontos, hogy olyan metrikákat használjunk, amelyeknél egyértelműen bizonyított, hogy az adott metrika használatát figyelembe véve milyen rendszertulajdonság vagy annak gyártási folyamatának melyik vetülete mérhető. Projektvezetői munkám során mindig nagy figyelmet szenteltem a megfelelő metrikák alkalmazásának, melyekkel a felvetődő problémák idejében észlelhetők, és kezelésük költséghatékonyan végrehajtható.

A tervezési alapelveket és tervezési mintákat a tapasztalt tervezők tudják jól használni, mivel nem egyértelmű azonosítani azokat a tervezési helyzeteket, amikor azokat használni lehet.
Ennek kezelését először a tervezési minták leírási módjainak javításával próbáltam elérni, amely során elsőrendű logikai formulákkal ábrázoltam a tervezési minták osztály és objektumszerkezetét. Ugyanakkor észrevettem, hogy a felhasználást nem a leírási módok hiányosságai akadályozzák, hanem az, hogy a tervezési minták leírásakor nem a tervezési helyzetek értelmezésével foglalkozunk, hanem azokat a megoldást jelentő osztály és objektumszerkezetekkel jellemezzük.
Célom ennek megfelelően annak meghatározása, hogy az egyes tervezési minták milyen tervezési helyzetekben használhatók.

Észrevételem szerint a tervezési alapelvek és azok céljai szerint megfogalmazott tervezési minták legfontosabb célja az implementációs függőségek típusain belül a különböző döntésszerkezetek szerinti döntésredundanciák, mint egyfajta implementációs függőségek feloldása. Nem megfelelően implementált, redundáns döntések implementációs függőségeket

eredményeznek, melyeket döntésszerkezeti implementációs függőségnek nevezek.
A programban lévő döntések során arról döntünk, hogy egy adott ponton milyen funkcionalitásra és/vagy adatszerkezetre van szükségünk. Amikor egy döntést definiálunk a döntés döntéslehetőségeinek funkcionalitását és/vagy adatszerkezetét adjuk meg. A döntéslehetőségen a döntés egy lehetséges kimenetelét értjük, amely a döntéslehetőségnek megfelelő funkcionalitás és/vagy adatstruktúra érvényre jutását jelenti, illetve a döntéslehetőség teljesülését jelentő döntéspredikátumot, ami alapján eldől, hogy egy adott döntési helyen melyik döntéslehetőség jut érvényre.
Ahhoz, hogy a döntésredundanciák feloldásának lehetőségeit megértsük az objektumorientált módszertan alapeszközének, az öröklődésnek az értelmezését kell kiterjeszteni: Az öröklődés a programban lévő döntésredundanciák feloldásának eszköze, amely a döntések összevonása és kiemelése után azok deklarációjára ad egy absztraktabb keretet. Ezzel természetesen a többi objektumorientált alapfogalom is más megközelítésbe kerül. A refaktorálás, mint a kód minőségjavítás módszere, foglalkozik a döntésstruktúrák osztály-alosztály kapcsolatokba történő kiemelésének lehetőségeivel; ugyanakkor nem részletezi a döntésredundanciák esetei szerinti döntésösszevonások lehetőségeit, melyet dolgozatom egyik eredményeként a döntésstruktúrák refaktorálással történő optimalizálásának fő lehetőségének gondolok.
A programokban lévő döntésismétlődések megszüntetésére a döntések osztályhierarchiákba történő kiemelésével vagy már kiemelt döntések összevonásával van lehetőség, amely után a döntések lehetőségeihez tartozó adattartalmak és funkcionalitások egy osztály alosztályaiban kapnak keretet. Tehát a döntésösszevonás lehetőségét megadó döntéskiemelés során a döntés definíciót egy absztraktabb formában, azaz egy osztályhierarchiában adjuk meg, ahol a döntés „felületét” egy polimorf metódus jelenti. Kiemelve a döntéseket és a döntéslehetőségeket egy osztályba és annak alosztályaiba, a megfelelő döntésösszevonásokat elvégezve lehetővé válik, hogy a döntéseket egy helyen meghozva, annak eredményét a kiemeléskor, összevonáskor megadott osztályba és annak alosztályaiba „zárva” (példányosítás) archiváljuk. A létrejövő objektumot egy referencia változóval hivatkozzuk, amely típusa a szülőosztály. Az adott döntés további döntési pontjain már a

meghozott, archivált döntés eredményét – szülőosztály típusú referencia típusú változót használhatjuk fel úgy, hogy nem kell figyelni arra, hogy a döntés során mely döntéslehetőség érvényesült, azaz melyik alosztályt példányosítottuk, hiszen a szülőosztály típusával hivatkozva a későbbi felhasználások során rejtve, bezárva marad, mivel a döntés felületét egy polimorf metódus adja.
A döntéseknek két állapotát különböztetjük meg attól függően, hogy az adott döntés definiálása, azaz a döntéslehetőségek adatstruktúrái, metodológiája egy metódusban van-e megadva vagy már kiemelten egy osztályban és annak alosztályaiban.
Meghatározom azokat a döntésszerkezeti jellemzőket, amelyek esetén a döntésredundanciák megszűntetése indokolja a döntésösszevonásokat. A döntésösszevonások megfogalmazott esetei kulcsfontosságúak a döntésszerkezetek alapján javasolt tervezési alapelvek és metrikák, tervezési minták jellemzői szempontjából.
Megfogalmazok két új tervezési alapelvet. Az első alapelv – „Öröklődést döntésredundanciák megszűntetésére használjunk!” – egyértelműen meghatározza azon eseteket, amikor az öröklődés, mint objektumorientált eszköz használata az objektumösszetétellel szemben indokolt lehet. Mivel a már meglévő tervezési alapelvek egy kritikus pontja annak eldöntése, hogy az öröklődés használata mely esetekben indokolt, ezért az új elv lényeges kiegészítést jelent a már meglévő alapelvek tekintetében. A döntésredundanciák csökkentése, mint új cél elérése érdekében megfogalmazott másik új alapelv: „A döntésredundanciákat kerüljük!”.
Az elképzelésnek megfelelően az objektumorientált alapeszközök és a már meglévő tervezési alapelvek értelmezését is kiegészítem, amely alapján a tervezési alapelvek céljaival összhangban levő tervezési minták céljai, leírásai is változnak.
Olyan új objektumorientált metrikákat vezetek be, melyekkel lehetőség van a programban szereplő döntésredundanciák mértékének megállapítására, illetve, melyekkel a bevezetett tervezési alapelvek teljesülése mérhető. („Döntésabsztrakciós metrika”, „Döntéskiemeléssel létrejött öröklődések aránya”, „Ekvivalens döntésesetek aránya”, „Döntéspredikátumában eltérő egyéb viselkedésében egyező döntésesetek aránya”, „Ekvivalens döntéspredikátumú döntésesetek aránya” metrikák)

A tervezési minták új értelmezés szerinti célja a jól strukturáltság elérése a döntésismétlődések megszűntetésével. Tehát a tervezési minták a döntéskiemelésekre adnak recepteket, vagyis arra, hogy a különböző strukturált döntési helyzetekben a döntésismétlődés megszüntetése miként biztosítható. A tervezési minták osztályozhatók aszerint, hogy milyen döntésszerkezetek esetén adják meg az optimális döntésdeklarációkat, illetve, hogy a döntésarchiválás mely módját támogatják.
A tervezési minták alkalmazásának egy új irányát vehetjük észre, miszerint a döntések nem kiemelt állapota esetén is észlelhetők azok a döntéskapcsolatok, amelyek az adott tervezési mintára jellemzőek, így a tervezési minták nem csak a jelenlegi formájukban ismerhetők fel, hanem a döntések nem kiemelt állapota esetén a tervezési minták „lenyomatai" is észlelhetők.
Ez új lehetőséget ad annak tisztázására, hogy milyen esetekben lehetséges egy adott tervezési minta felhasználása, hiszen egy tervezési minta leírásakor nem egy optimális megoldás paramétereit kell, hogy megadjuk, hanem egy olyan tervezési helyzet körülményeit, amikor egy adott tervezési minta felhasználásával egy optimálisabb struktúra kialakítása érhető el. Tehát a tervezési minták bevezetésének lehetősége sokkal érthetőbbé válik, mivel konkrét megoldástól független jellemzők, a döntésszerkezet szerint határozható meg egy tervezési minta alkalmazhatósága.
Abban az esetben, ha a döntések elemzése automatizált eszközökkel is megtehető, akkor lehetőség nyílhat a tervezési minta alkalmazás szükségességének automatikus felismerésére.
Hogy az elképzelésnek megfelelően a strukturáltságot, azaz a jól strukturált program irányelveinek teljesülését elemezni tudjuk, szükséges egy olyan formalizációs eszköz, amellyel a döntések viselkedése vizsgálható. Erre a Java programok viselkedés interfész specifikációs nyelvét, a JML-t használom, amely lehetőséget ad a döntés lehetőségeinek adatstruktúráját, metodológiáját logikai kifejezésekre alapozva specifikálni. Az így formalizált döntések vizsgálata a döntésösszevonás szabályai szerint már lehetséges.
A JML alapú döntésspecifikációra alapozva a tervezési mintákat jellemző döntésszerkezet is modellezhető. A „Híd" tervezési mintában található döntésszerkezet deklarációk különböző állapotainak formális leírását külön-külön adom meg, amely alapján megfogalmazom azokat az általános szabályokat, amelyek a JML-es

döntésszerkezet szerinti tervezési minta formalizálás alapjai lehetnek.

Az elgondolás helyességének vizsgálataként az elmúlt másfél évben folyó „eFilter” kutatás-fejlesztéses projekt projektvezetőjeként és vezető tervezőjeként a tervezési fázisában kezdeményeztem, hogy a tervezési minták alkalmazhatóságát a döntésszerkezetek elemzésével végezzük, mely során számos komplex tervezési problémára sikerült optimális megoldást találni. A projekt során a döntésszerkezetek – tervezési minták összefüggéseinek egy új irányát sikerült felfedezni, miszerint a tervezési mintákra jellemző döntési helyzetek már a használati eset modellekben is megjelennek. Új kutatási irány ennek megfelelően az, hogy a tervezési minták a használati eset modellekben miként jelennek meg, illetve hogy a tervezési minták döntésszerkezetei hogyan tükröződnek a használati esetek szintjén.

2. Az értekezés új tudományos eredményei

2.1 Döntésredundanciák megszűntetése döntésösszevonásokkal

Döntés, döntéskiemelés fogalmainak bevezetése
Bevezetem a megvalósítástól független döntés fogalmát, amely kifejti, hogy az öröklődéses osztályszerkezetek a döntések absztraktabb deklarációs módjai, illetve kijelenti, hogy a döntéskiemelés során a döntések viselkedése és adatszerkezete nem változik. Ezek a definíciók a refaktorálás témakörében már alkalmazott feltételes utasítások öröklődéssel és polimorf metódussal történő definiálására épülnek.
(3.1. Döntés és döntéskiemelés)

Döntésredundanciák megszűntetése döntésösszevonásokkal
A döntés, döntéskiemelés fogalmára alapozva bevezetem a döntésösszevonás fogalmát, mint a döntésredundanciák megszűntetésének eszközét. A különböző döntésredundancia típusokra külön adom meg a döntésösszevonás módszerét.
Ezek az esetek a későbbiek folyamán kritikusak, mivel ezek motiválják a tervezési alapelvekkel kapcsolatos javaslatok megszületését, illetve a kód minőség mérés lehetőségeit kiterjesztő metrikák bevezetését. A tervezési minták osztályozása szintén ezen esetekre épül.
(3.2. A döntésredundanciák elkerülésének esetei, 3.3. Döntésösszevonás)

Annak érdekében, hogy azon helyzetek felderíthetők legyenek, amikor a döntéskiemelés, döntésredundanciák feloldása indokolt, szükséges egy formalizációs eszköz. Felvázoltam a program formalizációs eszközök céljait, ismertettem fejlődési irányukat, mellyel a Floyd féle induktív állítások módszerére és Hoare féle deduktív levezetési rendszerére alapozva kialakult a JML, ami egy Java viselkedés specifikációs eszköz.

Döntéskiemelés során a döntésszerkezetek JML specifikációja miként változik:

A döntések formális leírására és elemzésére a JML-t használom, mely szerint ismertetem az egyszintű és többszintű döntések nem kiemelt és kiemelt állapot szerinti JML specifikációjának tulajdonságait.
A döntésszerkezetek JML alapú formalizálásával kapcsolatban új eredménynek tartom az egyszintű és többszintű döntésszerkezetek JML alapú formalizáció módjának ismertetése során a nem kiemelt és az osztályszerkezetbe kiemelt döntések JML specifikációs lehetőségeinek bemutatását, amely alapján érzékelhető, hogy milyen változások következnek be a viselkedési szerződésben a kiemelés során.
4.2 Döntések formalizálása nem kiemelt állapotban
4.3 Döntések formalizálása kiemelt állapotban

Döntésösszevonás eseteinek JML alapú megfogalmazásai:
A „3.3 Döntésösszevonás” fejezetben ismertetettem azon esetek, amikor a döntésredundancia megszűntetése indokolja a döntésösszevonást. Az ott megfogalmazott kritériumokat JML specifikációs eszközökkel is definiálom a „4.4 Döntésösszevonás eseteinek vizsgálata JML-lel” fejezetben.

2.2 „Öröklést döntésredundanciák megszűntetésre alkalmazzunk!”, „Döntés-redundanciát kerüljük!” tervezési alapelvek bevezetése

Látnunk kell, hogy a döntésösszevonás esetei a „2.2 Az objektumorientált tervezés alapelvei” fejezetben taglalt számos elvet új megvilágításba helyeznek. A leglényegesebb, hogy a döntésösszevonás esetei egyértelmű segítséget adnak annak eldöntésében, hogy mikor szükséges és indokolt öröklődés, azaz osztályhierarchia bevezetése.
Bevezetek két olyan új objektumorientált tervezési alapelvet, melyek a döntésredundancia elkerülés szabályaiból táplálkozva fogalmaznak meg két új szabályt. Az „Öröklést döntés redundanciák megszűntetésre alkalmazzunk!” javasolt alapelv egyértelműen próbálja deklarálni azon eseteket, amikor az öröklődés használata indokolt.

„A döntésredundanciát kerüljük” tervezési alapelv általánosan fogalmazza meg a döntésredundanciák elkerülésének szükségességét, amely a feltételezésem szerint a programminőség javulását eredményezi.
(3.6 Objektumorientált tervezési alapelvek kiegészítése)

2.3 Döntésredundanciák mérése új metrikákkal

Olyan új objektumorientált metrikákat vezetek be, melyekkel lehetőség van a programban szereplő döntésredundanciák mértékének megállapítására.

Ezek közül a legegyszerűbben mérhető a „Döntésabsztrakciós metrika”, amely a kiemelt – nem kiemelt (polimorf metódushívások száma – polimorf metódushívások száma+„if-else” utasítások száma) döntésesetek számosságának arányát vizsgálja.
Kísérleti eredményekkel igazolom, hogy a kiemelt döntésesetek arányának növekedése javítja a kód stabilitását.

A „Döntéskiemeléssel létrejött öröklődések aránya” metrikát az „Öröklést döntésredundanciák megszűntetésre alkalmazzunk!” új tervezési alapelv teljesülésének mérésére vezetem be.
Feltételezem, hogy, ha egy osztály-alosztály kapcsolatban van polimorf metódus (a kiemelt döntések felületeit a polimorf metódusok adják), akkor az adott öröklődés döntésredundanciák megszűntetésére jött létre.
Kísérleti eredményekkel igazolom, hogy a döntéskiemeléssel létrejött öröklődések arányának növekedése javítja a kód stabilitását.

A „Ekvivalens döntésesetek aránya”, „Döntéspredikátumában eltérő egyéb viselkedésében egyező döntésesetek aránya”, „Ekvivalens döntéspredikátumú döntésesetek aránya” metrikákkal a döntésredundanciák eseteinek teljesülése mérhető. Ezek felhasználásával szofisztikáltabb képet kaphatunk a döntésszerkezetek állapotáról, melyekhez viszont a döntések viselkedési szerződésének elemzés szükséges. Ezen metrikák

alkalmasak „A döntésredundanciát kerüljük!” tervezési alapelv teljesülésének mérésére.
(3.7. Döntésszerkezet elemzése új metrikákkal)

2.4 Tervezési minták jellemzése, osztályozása döntésszerkezetek alapján; Tervezési minták döntésszerkezeteinek JML alapú specifikációja

Tervezési minták jellemzése döntésszerkezetekkel
A tervezési minták jellemzését a tervezési helyzeteket leginkább kifejező döntés szerkezetekkel fejezem ki. A tervezési minták felhasználhatósága így könnyebbé válik, mivel jellemzésük tervezési megoldás független módon, de kellő egzaktsággal tehető meg.
A „5.1. Tervezési minták döntésszerkezetek szerinti értelmezése” alfejezetben a „GOF”könyvben (Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.; Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley Professional Computing Series; 1995.) összegyűjtött tervezési minták döntésszerkezeteit mutatom be.
Az elképzelés alkalmazhatóságának igazolása céljából az „eFilter” kutatásfejlesztési projekt tervezési szakaszában a döntési struktúra elemzésével sikerült felfedezni néhány tervezési minta alkalmazhatóságát, melyek közül egy esetet az "5.2. Sablonfüggvény tervezési minta döntésszerkezet szerinti felismerése az eFilter projekten” fejezetben, mint esettanulmányt ismertetek.

Tervezési minta felismerés, automatikus felismerés lehetősége döntésszerkezet alapján, megvalósítástól függetlenül
A tervezési minták döntésszerkezete alapján lehetőség van arra, hogy a tervezési minták alkalmazhatóságának lehetőségét a programtervezők könnyebben észrevegyék.
A döntésszerkezet szerinti tervezési minta felismerés esettanulmánya az „5.3. Tervezési minták jellemzése döntéskiemelés előtt és után” fejezetben található, ahol a „*Híd*” tervezési minta döntésszerkezetének megjelenését ismertetem döntéskiemelés és összevonás előtt és után. Egy példaprogrammal

teszem érthetőbbé a „*Híd*” tervezési minta döntésszerkezeteinek megvalósulásait.
A tervezési minták döntésszerkezeteire alapozva a tervezési minták automatikus felismerése is támogatható úgy, hogy a megvalósítástól független döntésszerkezetek szerint tudjuk észrevenni a tervezési minták „lenyomatait” a tervben vagy forráskódban. Ez a tervezési minta automatikus felismerés egy újszerű megközelítése lehet, amit új kutatási lehetőségnek tartok.

Tervezési minták osztályozása döntésszerkezetek szerint
A tervezési minták osztályozhatók aszerint, hogy milyen döntésszerkezet esetén adják meg az optimális döntésdeklarációkat, mellyel a döntésredundanciák elkerülése biztosítható.
A „5.1.4. Tervezési minták osztályozásának új módszere” fejezetben bemutatott osztályozást olyan új eredménynek tartom, ami hozzájárulhat ahhoz, hogy a tervezési minták döntésszerkezetből fakadó esetleges hasonlóságait észrevehessük, felhasználhatóságukat javítsuk.

Tervezési minták döntésszerkezeteinek JML alapú specifikációja
A JML lehetőséget ad arra, hogy a tervezési minták döntésszerkezeteit egzakt módon, a döntések reprezentációjának módjától függetlenül deklaráljuk. Így megoldható a döntések viselkedési szerződéseinek összehasonlításával azon döntésszerkezetek felismerése, melyek az egyes mintákra jellemzők.
A tervezési minták JML alapú formalizálását a „*Híd*” tervezési minta döntésszerkezetének JML alapú specifikációjával szemléltetem: „5.4.1. „Híd” tervezési minta JML alapú formális leírása”, amit egy esettanulmánnyal teszek érthetőbbé az „5.4.2. „Híd” tervezési minta JML alapú formális leírása példán keresztül” fejezetben. Ezekre alapozva az „5.4.3. Tervezési minták döntésszerkezetek szerinti JML alapú formalizálásának szabályai” fejezetben összegzem tapasztalataimat.

3. Az értekezés témájában született publikációk

Referált folyóirat publikációk:

- Márien Szabolcs
  Decision Based Examination of Object-Oriented Programming and Design Patterns
  Teaching Mathematics and Computer Science 6/1, pages: 83-109, 2008
- Márien Szabolcs
  Decision Based Examination of Object-Oritented Methodology Using JML
  Annales Mathematicae et Informaticae , pages: 95-121, 2008

Konferencia kiadványában megjelent lektorált publikációk:

- Márien Szabolcs, Kusper Gábor
  Understanding Design Patterns as Constructive Proofs
  Proceedings of ICAI-2004, Volume II. 173-182, Eger, Hungary, January 2004.

Konferencia beszédek egy oldalas összefoglalóval a konferencia kiadványban:

- Márien Szabolcs
  Automated excavation and detection of Design Patterns
  Proceedings of CSCS-2004, Szeged, Hungary, July 2004.
- Kusper Gábor (Eszterházy Károly Főiskola), Márien Szabolcs (Wit-Sys Zrt.), Dr. Kovács Emőd (docens - Eszterházy Károly Főiskola)
  Innovatív megoldások az eFilter projektben
  Informatika a felsőoktatásban 2011. Konferencia, Debrecen

4. Az értekezés témájával kapcsolatos projekt

„eFilter" kutatásfejlesztési projekt: Egészségügyi profil alapján szűrt élelmiszer és fogyasztási adatbázisokból nyert információkat kezelő rendszer

Leírás: Étel allergiások, cukorbetegek és tudatosan táplálkozók személyre szabott egészségügyi profiljuk alapján a rendszer segítségével könnyen és önállóan tudják különválasztani a fogyasztás szempontjából engedélyezett-tiltott élelmiszereket. Az információkérés a széles körű hozzáférés érdekében mobiltelefonon keresztül vonalkód leolvasásával történik.

Szerepkör: Projektvezető, vezető tervező

Erőforrás – projektméret: 6000 embernap

Időszak: 2010.-2011.

Projekt keretében megjelent/elfogadás alatt álló publikációk:

- Élelmiszer adatbázis szűrése mennyiségi megszorítások alapján logaritmikus indexeléssel; AIK 2011 (Elfogadás alatt)
- Developing an Expert System for Diet Recommendation; SACI 2011; http://conf.uni-obuda.hu/saci2011
- Valós időben választ adó egészségügyi profil, mint több dimenziós megszorítás mátrix, alapján élelmiszert szűrő domain specifikus algoritmus; NETWORKSHOP 2011.
- Innovatív megoldások az „eFilter" projektben; Informatika a felsőoktatásban 2011. Konferencia; Debrecen
- Betegségek, allergiák, ételérzékenységek leírására alkalmas XML séma tervezése; NETWORKSHOP 2011.

5. Antecedents and aims of the Doctor's dissertation

During my work, I have taken part in many projects as designer and project leader. According to my experiences the success of a development is guaranteed by the adequate designing and the quality assurance based on it. By quality assurance, I understand the handling and feedback of the design and development errors.

The product of designing is the system plan, the model, but it's quality is not easy to measure, the design errors come to light to late, when the correcting would be to expensive due to the many dependencies. During my work as a designer, I often noted that, because of the opaque web of dependencies, the designers and the developers do not realize that the problem they face is just a consequence, the mirroring of a previously done non-appropriate design solution.

The goal of the methodology of object-oriented programming is to help the creation of well-structured and so, unambiguous and easily expendable and so, maintainable and recycled and so, modular programs. But reaching these goals, namely to create a satisfactorily stable and well organized program, requires a lot of designing and developing experience.

To reach the goals, general rules have been created which determined ***coupling*** and ***dependency*** – implementation dependency - as the key qualities for object-oriented systems. With a high degree of the later, the system may become unmaintainable by inconsistent modules.

By *coupling*, we understand the strength of the connection between the different system components. The coupling between two components strengthens if one points to a field in the other, calls it's method or the method's returning type is the type of the other component. The strongest form of coupling is the inheritance. With a loose coupling, the modification of one class does not cause modification compulsion in the other classes.

The implementation dependencies are consequences of the coupling, which increase the risk of instability and reduce the maintainability.

Reducing the implementation dependency is of critical importance, as it adds to the loose coupling that is highly important for maintainability and recyclement.
The loose coupling contributes to the creation of autonomous system components, which can be easily replaced or maintained. Without loose coupling, the dependencies result in an opaque system, which makes the maintainability impossible.
The great freedom of the designing work, the selection of the most reasonable solution out of the bulk of solutions, is very risky, which can only be lowered with great designing experience. Therefore, it is practical to formulate the solutions that worked before (in designing aspect) as rules to help their re-using. That is why we find structures with similar class- and object-structures in different object-oriented systems that are used by specialists to give similar solutions to reoccurring similar problems. We call these re-usable designing solutions, general "recipes" that are used as solutions for reoccurring designing problems as design patterns, which are devices of the quality design and development. More specifically, they are descriptions of objects and classes working together, that solve, tailored to a case, some general designing problem. The design patterns are created as the result of collecting designing experiences, which give advices to the designer during the design process on how to solve the given design solution properly. In many cases, we face similar, generalizable design problems, that worth drawing professionally discussed and accepted answers and then use those specified to the case.
The design patterns formulate the methods, in an abstract way, but according to the goals of designing principles, that can be used in the given design situations to create proper object-oriented structure.

The quality assurance is of prime importance, as guarantees are needed to ensure that the created product is suitable for the expectations and can safely provide the required operation. But we also need to be able to measure the quality. The engineering profession uses standards to describe the quality of the created product. Contrary to other fields of engineering, software development has no such standards that clearly define the quality, only indirect standards that allows the measurement of a software's quality from one aspect or another.

Many solutions have born to measure the realization of the mentioned goals. Different metrics are used to measure realization that try to qualify the structuredness from many point of views.
For the project leaders and designers, choosing the correct metrics and measuring device systems, that can be used to measure the quality of a software according to the selected point of views is not obvious. This is caused by the fact that the number of measurable abilities is infinite, so it is important to select a metrics, where it is unambiguously proven, that using the given metrics which system ability or projection of the manufacturing process can be measured. During my work as a project leader, I always devoted great attention to applying the appropriate metrics, that can help to notice the arousing problems in time and their handling can be done cost efficiently.

According to my experiences, experienced designers can use the designing principles and patterns well, as it is not easy to identify those design situations when these elements can be used.
I tried to handle this by correcting the describing method of the design patterns first, in which I represented the class- and object structure of the design patterns with first-order logic formulaes. In the same time, I recognized that the usage is not hindered by the deficiencies of the describing method, but the fact, that when describing design patterns, we do not deal with understanding the design situations, but we try to describe them with solving class and object structures.
According to this, my goal is to define which design patterns can be used in what design situation.

As I noticed, the most important goal of design patterns and design principles is the elimination of the decision structure based redundancies as a kinds of implementation dependencies. The not properly implemented, redundant decisions result in implementation dependencies, which are called decision structure implementation dependencies.
During the decisions in the application, we decide about the functionality and/or data structure needed at a given point. When we define a decision, we give the functionality and/or data structure of the decision's decision options. By decision option, we mean one

possible outcome of a decision, which means the enforcing of the functionality and/or data structure appropriate for the decision option, and the decision predicate which decides which decision option will set off at a given decision location.
In order to understand the options of dissolving the decision redundancies, we need to extend the basic concept of object-oriented methodology, the inheritance: The inheritance is the tool to dissolve the decision redundancies in the software, which gives a more abstract framework to declare decisions after their contraction and raising. With this, all the other basic concepts of object-oriented programming change as well.
Refactoring, as the tool of improving the quality of the code, handles the options of raising the decision structures into class-subclass relations, but in the same time, it does not detail the options of decision contraction according to the cases of decision redundancies, that, as a result of my thesis, I think to be the main option of optimizing the decision structures by refactoring.
To extinguish the decision-recurrence in softwares, we have the option to organize the decisions into class hierarchies or to contract the already organized decisions, after which the data that belongs to the decisions' options and functionalities will get a frame in a class' subclasses. So during the decision-raising that gives the opportunity of the decision-contraction, the decision definition is given in a more abstract form, namely in a class hierarchy, where the "surface" of the decision is meant by a polymorph method. By raising the decisions and decision options into a class and it's subclasses, than executing the appropriate decision contractions, it becomes possible to make the decision in one place and to archive it by closing the results into the same class and it's subclasses given during the raising and contraction (instantiating). The object created this way is referenced with a reference variable which's type is the parent class. On the further decision locations of the given decision we can use the result of archived decision – as a variable with a reference type and parent class type – in a way that we do not need to check what decision option was enforced during the decision, namely which subclass had been instantiated, because it remains hidden and closed during further usage as it is referenced with it's parent class because the surface of the decision is given by a polymorph method.

We differentiate two states of the decisions depending on that if defining the given decision, namely the data structures, methodology of the decision options are given in one method, or are raised to a class and it's subclasses.
I appoint those decision structure parameters, when the elimination of the decision redundancies justifies the decision-contractions. The defined cases of decision-contractions are the keys to the parameters of the design patterns, the designing principles and metrics suggested by the decision structures. metrics.
I formulate two new design principles. The first principle – "Inheritance is used dissolve decision redundancies." – clearly identifies those cases, when the usage of inheritance as object-oriented tool is reasonable over against object composition. As one of the critical points of the existing design principles is the decision of when is the use of inheritance justified, the new principle means a significant supplement in point of the extant principles. In favour of the diminishing of the decision redundancies, as an achievable new goal, the other new principle is the following: "Avoid decision redundancies."
According to the concept, I also expand the interpretation of the basic object-oriented paradigms and the existing design principles, under which the goals and descriptions of the design patterns in consort with the goals of the design principles will change to.
I initiate such new object-oriented metrics, that gives opportunity to determine the degree of decision-redundancies in the software, or rather, with which the fulfillment of the initiated design principles is measurable. (The "Ratio of equivalent decision cases", „Ratio of decision cases different in their decision predicate but the same in any other behavior", „Ratio of decision cases with equivalent decision predicate" "Ratio of inheritances coming into existence by decision-raising", "Metrics of Decision-abstraction" metrics.)
The goal of design patterns according to the new presentation is to achieve a well-structured program by dissolving decision recurrences. So the design patterns give a recipe for decision-raising, or rather to that how can the decision recurrences be avoided in different structured decision situations. The design patterns are classifiable by in what cases of decision structures do they give the optimal decision-declarations, or rather what kind of decision archiving method do they support.

We can notice a new way in using the design patterns, that the decision-structures that are typical to the given design pattern can be noticed even if the decision is not in a raised state, hence the design patterns are not only recognizable in their current form, but the "imprints" of the designing patterns can be noticed in the non-raised state of the decisions.
This gives a new opportunity to clarify in what cases is the usage of a given design pattern is possible, as when describing a design pattern we do not need to give the parameters of an optimal solution, but the circumstances of design situation, when by using the given design pattern a more optimal structure can be developed. Thus the possibility of initiating the design patterns become more understandable, as the applicability of a design pattern can be determined by parameters independent of the factual solution, which is according to the decision structure.
In the case when the analysis of the decision can be done by automated tools, it may be possible to automatically realize the need of using a design pattern.
As to be able to analyze, according to the idea, verify the structuredness, namely the fulfillment of the principles of a well-structured software, we need a formalization tool that can be used to examine the behavior of decisions. For this, I use the behavior interface specification language of the Java applications, the JML, that enables to specify the data structure, methodology of the decisions' options based on logical expressions. The examination of the decisions formalized in this manner is now possible according to the rules of decision-contraction.
Built upon the JML-based decision specification, the decision structure describing the design patterns can be modeled to. I give the formal descriptions of the different states of the decision structure declarations found in the design pattern one by one, by which I formulate those general rules, that can be the basics of the design pattern formalization according to the JML decision structure.

As the examination of the conception's correctness, as project leader and lead designer of the eFilter research-expansion project going on in the last one and a half year, in the design phase I initiated that the design patterns' applicability be analyzed by the decision structures, during which we have managed to find optimal solution for many complex designing problems. During the project,

I have managed to discover a new direction in the connection of the decision structures and design patterns, that the design patterns yet appear in the use case models based on decision situations. According to this, the new research direction is to find out how does the design patterns appear in the use case models, or rather how does the decision structures of the designing patterns reflect on the level of use cases.

6. The new scientific results of the dissertation

6.1 Dissolving decision redundancies by decision-contractions

Initiating the concepts of Decision and Decision-raising
I initiate the realization–independent concept of decision, which unfolds that the inheritance class structures are the more abstract declaration methods of decisions, or rather it states that during decision-raising the behavior and data structure of the decisions do not change. These definitions are based on defining conditional instructions by inheriting and polymorph method used in the topic of refactoring.
(3.1. Decision and decision-raising)

Dissolving decision redundancies by decision-contractions
Based on the concept of decision and decision-raising, I initiate the concept of decision-contraction, as the tool of dissolving decision redundancies. I give the method of decision-contraction one by one for the different decision redundancy types.
These cases are going to be critical later, as these motivate the birth of the suggestions in connection with the design principles, or rather the initiation of the metrics that allow the measurement of the code's quality. The classification of the design patterns is also based on these.
(3.2. The cases of avoiding decision redundancies, 3.3. Decision-contraction)

In order to be able to explore the situations when the decision-raising, the elimination of the decision redundancies are justified, we need to have a formalization tool. I have outlined the goals of the

software formalization tools, I have laid their direction of development, which was used develop the JML, which is a Java behavior specifying device based on Floyd's method of inductive statements and Hoare's deductive derivation system.

How does the JML specification of the decision structures during decision-raising:
I use the JML to the formal description and analysis of the decisions, by which I have laid the JML specification's parameters by the single- and multi-level decisions' raised and non-raised states.
In connection with the JML-based formalization of the decision structures, I find it a new goal: the review of the non-raised and class structure-raised decisions' JML specification options during the review of the JML-based formalization of the single- and multi-level decision structures, that can be used to notice what changes will happen in the behavior contract during the raising.
4.2. The formalization of decisions in non-raised state
4.3. The formalization of decisions in raised state

The JML-based composition of the decision-contraction's cases:
In the 3.3. Decision-contraction chapter, I laid those cases when the elimination of decision redundancy justifies the decision-contraction. I also define the criterias mentioned there by the JML specification tools in the 4.4. Analyzing the cases of decision-contraction by JML chapter.

6.2 The initiation of the „Using inheritance to dissolve decision-redundancy", „avoid decision-redundancy" design principles

We have to see that the cases of decision-contraction set into a new light many of the principles discussed in chapter 2.2. The principles of object-oriented design. The most important is that the cases of decision-contraction give an obvious help to decide when the inheritance, namely the initiation of the class hierarchy is required and justified.

I initiate two such new object-oriented design principle, that are using the rules of avoiding decision-redundancy to formulate two new rules. The "Inheritance is used to dissolve decision-redundancy" suggested principle obviously tries to declare those cases, when the use of inheritance is justified.
The "Avoid decision-redundancy" design principle formalizes the need for avoiding decision-redundancy in general, that will result in the improvement of the software's quality.
(3.6. Amendment of the object-oriented design principles)

6.3 Measuring the decision-redundancies with new metrics

I initiate such object-oriented metrics, that offers the opportunity to measure the degree of decision-redundancies in the software.

Of these, the most easily measurable is the "decision-abstraction metrics", that analyzes the ratio of the raised and non-raised (number of polymorph method callings / (number of polymorph method callings + the number of "if-else" instructions)) decision cases' cardinality. I assume that the improving ratio of the raised decision cases will improve the stability of the code as well.

The „ratio of inheritances coming into existence by decision-raising" metrics is initiated to measure the fulfillment of the new design principle "Inheritance is used to dissolve decision-redundancies".
I assume that, if there is a polymorph method in a class-subclass connection (the surfaces of the raised decisions are given by polymorph methods), than the given inheritance has come into existence to dissolve decision redundancies.

The "Ratio of equivalent decision cases", „Ratio of decision cases different in their decision predicate but the same in any other behavior", „Ratio of decision cases with equivalent decision predicate" metrics can be used to measure the fulfillment of the decision-redundancy cases. By using these, we can get a more sophisticated picture of the state of decision-structures, for which

the analysis of the decisions' behavior contract is needed. These metrics are applicable to measure the fulfillment of the "Avoid decision-redundancies" design principle.
(3.7. Analyzing the decision structures with new metrics)

6.4 Analyzing and classifying the design patterns based on decision structures

Characterizing design patterns by decision structures
I express the characterization of the design patterns with the decision structures that express the design situations the most. The usage of the design patterns becomes easier this way, as their characterization can be done independently from a design solution, but with enough exactness.
In the 5.1. Interpreting design patterns according to the decision structures sub-chapter, I introduce the decision structures of the design patterns gathered in the "GOF" book (Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.; Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley Professional Computing Series; 1995.).
To confirm the usability of the idea, I managed to discover the usability of a few design patterns by analyzing the decision structure during the design stage of the eFilter research-development project, of which I review one as a case-study in chapter 5.2. Recognizing a "Template method" design pattern by decision structure in the eFilter project.

Design pattern recognition, the option of automatic recognition by decision structure, independent of realization
By the decision structure of the design patterns, there is an opportunity for the software designers to recognize option of the usability of the design patterns more easily.
The case-study of design pattern recognition by decision structure can be found in chapter 5.3. Design pattern before and after decision raising, where I review the appearance of the "Bridge" design pattern's decision structure before and after raising and contraction. I use an example software to make the realization of the "Bridge" design pattern's decision structures more understandable.

Based on the design patterns' decision structures, the design patterns' automatic recognition is eligible by noticing the design patterns' "imprints" in the plan or in the source code according to decision structures independent of realization. This can be a new approach to the automatic recognition of design pattern, that I think to be a new research opportunity.

The classification of design patterns by decision structures
The design patterns are classifiable by in case of what decision structure do they give the optimal decision declarations, which are used to ensure the avoidance of decision redundancies.
I find the classification in chapter 5.1.4. A new method to classify design patterns such an accomplishment, that adds to the chance of recognizing the incidental similarities of the design patterns that have their roots in the decision structures and improves their usability.

The JML-based specification of the design patterns' decision structures
The JML gives us the opportunity to declare the design patterns' decision structures in an exact way, independent from the mood of the decision's representation. This way we can solve by comparing the decisions' behavior contracts the recognition of those decision structures that are typical to the certain patterns.
I demonstrate the design patterns' JML-based formalization by the JML-based specification of the "bridge" design pattern's decision structure: 5.4.1. JML-based formal description of the "Bridge" design pattern, which is made more understandable by a case-study in chapter 5.4.2. JML-based formal description of the "Bridge" design pattern through examples.
Based on these, I summarize my experiences in chapter 5.4.3. Rules of the JML-based formalization of design patterns' decision structure.

7. Publication in the topic of the dissertation

Learned journal publications:

- Márien Szabolcs

  Decision Based Examination of Object-Oriented Programming and Design Patterns

  Teaching Mathematics and Computer Science 6/1, pages: 83-109, 2008

- Márien Szabolcs

  Decision Based Examination of Object-Oriented Methodology Using JML

  Annales Mathematicae et Informaticae , pages: 95-121, 2008

Vetted publications on conference-brochures:

- Márien Szabolcs, Kusper Gábor

  Understanding Design Patterns as Constructive Proofs
  Proceedings of ICAI-2004, Volume II. 173-182, Eger, Hungary, January 2004.

Conference speeches with a one-page summary in conference-brochures:

- Márien Szabolcs

  Automated excavation and detection of Design Patterns
  Proceedings of CSCS-2004, Szeged, Hungary, July 2004.

- Kusper Gábor (Eszterházy Károly Főiskola), Márien Szabolcs (Wit-Sys Zrt.), Dr. Kovács Emőd (docens - Eszterházy Károly Főiskola)

  Innovative solutions in the eFilter project

  Informatics in higher education 2011 Conference; Debrecen

8. Project in connection with the topic of the dissertation

„eFilter" Research-development project: a system that handles information it got from health-profile sorted food and consumer databases

Description: With the help of the system, those who have some kind of food allergy, have diabetes or are conscious consumers, can easily and on their own devide what they are allowed to eat and what they are not allowed to eat based on their personal health profile. To provide a wide range of access, the information is given through cell phone by reading a bar-code.

Role: Project leader, lead designer

Resource – project size: 6000 man-days

Time: 2010.-2011.

Publication in connection with the project/under acceptance:

- Sorting a food database by quantity restraints with logarithmic indexing; AIK 2011 (under acceptance)
- Developing an Expert System for Diet Recommendation; SACI 2011; http://conf.uni-obuda.hu/saci2011
- Health profile answering in real time, as a multi-dimensional restraint matrix, food sorting domain specific algorithm; NETWORKSHOP 2011
- Innovative solutions in the eFilter project; Informatics in higher education 2011 Conference; Debrecen
- Designing an XML pattern to describe illnesses, allergies and food irritabilities; NETWORKSHOP 2011.