FIATAL MŰSZAKIAK TUDOMÁNYOS ÜLÉSSZAKA

Kolozsvár, 2003. március 21-22.

TERVEZÉSI MINTÁK METAMODELLEZÉSE

Petri Dániel

Összefoglalás

Design patterns capture low level object-oriented concepts. They contain several informál parts like problem description, solution and consequences. The formalization of these parts simplifies their application in tools that deal with e.g., automatic design pattern matching. Therefore we created a design pattern metamodel that allows storing information described in the informál sections in a formai way attached to the pattern's *class diagram*. Using this metamodel based on a software specification the right design patterns can easily be found without knowing the tremendous amount of patterns and querying the properties of a pattern is also possible.

A tervezési minták alacsony szintű objektum-orientált vezérelveket tartalmaznak. A minták számos informális fejezetből állnak, mint például egy probléma és megoldásának leírása, valamint a megoldás alkalmazásának következménye. Ezen részek formalizálásával lehetővé tesszük olyan eszközök használatát, amelyek tervezési minták automatikus keresését végzik. A cél érdekében létrehoztunk egy tervezési minta metamodelt, amely segítségével az informálisan dokumentált információ formálisan leírható és a minta *osztály diagramjához* csatolható. E metamodel segítségével automatikusan megtalálhatók a megfelelő tervezési minták egy szöveges specifikációból kiindulva anélkül, hogy ismernénk az eddig elkészült hatalmas mennyiségű tervezési mintát.

1. Bevezető

A tervezési minták építészetből átvett alapötletét az 1990-es évek elején kezdték el alkalmazni a szoftver világban. Tervezési minták formájában tetszőleges absztrakciós szinten fellépő probléma és bevált, esetleg bizonyítottan helyes megoldásának váza dokumentálható, valamint az újrafelhasználáskor könnyen eldönthető, hogy az adott megoldás milyen következményekkel alkalmazható. A tervezési mintákkal kapcsolatban [1] és [2] úttörő jelentőségű művek, az koncepció és a formátum bemutatása mellett számos alapvető tervezési mintát ismertetnek. Megjelenésük óta hatalmas mennyiségű mintát dokumentáltak és a minták száma gyorsan növekszik.

A tervezési minták négy fő részből állnak: (i) a minta adminisztrációs célokat szolgáló fejrészből, (ii) a probléma ismertetéséből egy adott kontextuson belül, (iii) a probléma megoldásának vázából, valamint (iv) a megoldás alkalmazásának következményeiből.

Ezek a fejezetek szöveges, többnyire angol nyelvű leírásokat tartalmaznak. A *megoldás* statikus és dinamikus viselkedésének ismertetésére többnyire UML¹ osztály és szekvencia diagramot alkalmaznak, valamint szöveges formában ismertetik a megoldás résztvevőit, ezek felelősségeit és együttműködéseiket.

A szakirodalomban számos tervezési mintával kapcsolatos eszközt és ötletet dokumentáltak a következő célokkal: (i) tervezési minták automatizált keresése, (ii) minta specifikációja és implementációja közötti konformitás ellenőrzése, (iii) minták azonosítása egy implementációban, valamint (iv) minták közötti kapcsolatok megállapítása. Az irodalomban megtalálható formalizálási törekvések (pl. [3], [4]) is ezeket a célokat szolgálják, de többnyire a mintában található UML diagramokra koncentrálnak. Ahhoz, hogy az automatizált tervezési minta keresést lehetővé tegyünk, hangsúlyt kell fektetnünk az informális fejezetekben található információra is. Ezt a célt támogatja a következő fejezetben ismertetett minta metamodel.

2. A minta metamodel

Az ábrán az elkészült metamodel látható. Létrehoztuk a *Tervezési minta* absztrakt ősosztályt, amelynek minden eredeti, a mintában szereplő osztály és objektum közvetve a leszármazottja. Ehhez kapcsolódik egy-több megfeleltetéssel az *Állítmány*

osztály, majd ehhez több-több megfeleltetéssel a *Tárgy* osztály. Ezen osztályok *Név* attribútumaiban tároljuk a minta viselkedését és tulajdonságait leíró információt. Az absztrakt *Névosztály* a minta eredeti nevét veszi fel, ő a közvetlen őse a mintában szereplő eredeti osztályoknak, valamint az *Általános leírás* osztálynak, amely a *Probléma*, a *Megoldás*, valamint a *Következmények* absztrakt osztályok közvetlen őse. A minta tulajdonságait az OCL² segítségével írjuk le, a kereséshez pedig e nyelv lekérdezési lehetőségeit használjuk. A *Tervezési minta* osztályból létrehoztunk továbbá egy

¹ Unified Modeling Language: a szabványos vizuális objektum-orientált modellező nyelv
² Object Constraint Language: Formális nyelv kifejezések leírására, amelyek tipikusan invariáns feltételeket fogalmaznak meg UML modellek esetében. [5]

rekurzív asszociációt, amely azt a célt szolgálja, hogy a *Névosztályból* kiindulva OCL-ben tetszőleges alosztály tulajdonságai lekérdezhetők legyenek.

3. Minták kiegészítése a metamodel alapján

A minták kiegészítése a metamodelnek megfelelően egyelőre kézzel történik, az automatikus kiegészítés lehetősége további kutatás tárgyát képezi. Új minták esetében a készítő számára ez a kiegészítés nem okoz jelentős többletmunkát. Meglevő minták esetében ahhoz, hogy a kiegészítést a metamodelnek megfelelően elvégezzük, a leírás alapján össze kell gyűjteni az állítmányokat és a hozzájuk tartozó tárgyakat, ahogy azt a következő fejezetben ismertetett példa mutatja. Célszerű egy-egy tulajdonságot többféleképpen is megfogalmazni, így növelhető a minta megtalálásának esélye. A *Probléma*, a *Megoldás*, valamint a *Következmények* osztályok segítségével pedig lehetőség nyílik az összefoglalás kategorizálására. Az átalakított tervezési mintákat egy hierarchikusan strukturált adatbázisban kell tárolni, amelyben a minta részletei is kereshetők.

4. Példa: az Adapter minta

A minta célja egy létező osztály interfészét a kliensek elvárásainak megfelelően átalakítani. Így olyan osztályok is együtt tudnak működni, amelyek az Adapter minta nélkül a különböző interfészeik miatt erre nem volnának képesek. A minta leírását végigolvasva az ábrán látható állítmányok és tárgyak kigyűjtése tűnik célravezetőnek. Mivel a minta dokumentálása eredetileg angolul történt és az Interneten fellelhető minták többsége is angolul van, az általános kereshetőség érdekében célszerű volt az eredeti nyelv mellett maradni és az állítmányokat és tárgyakat nem lefordítani.

5. Minták keresése

Az alábbi OCL keresőkifejezéssel azokat a mintákat találhatjuk meg, amelyek az újrafelhasználhatóságot hivatottak elősegíteni:

```
context Általános_leírás
inv: self.Állítmány->select(Név='Increase')-> exists(Tárgy.Név='Reusability')
```

Ezt a keresőkifejezést minden egyes tervezési mintára le kell futtatni a teljes minta adatbázison belül. A kifejezés első fele kiválasztja azokat a mintákat, ahol az *Állítmány Név* attribútuma megegyezik az 'Increase' szóval, és ezen a halmazon belül *igaz* boolean eredménnyel tér vissza azon minták esetében, ahol a *Tárgy Név* attribútuma megegyezik a 'Reusability' szóval.

6. A megközelítés előnyei

Ha egy adott szöveges rendszerspecifikáció alapján keressük a megfelelő tervezési mintákat, kiindulásként választhatunk Internetes keresőket is. Azonban még abban az esetben is, ha a megfelelő keresőkifejezéseket írtuk be, nem garantált, hogy a megfelelő találatokat kapjuk, mert a megadott

keresőkifejezés szavai nem biztos, hogy a megfelelő környezetben, illetve a megfelelő sorrendben vannak. A cikkben ismertetett megközelítés legnagyobb előnye a célzott minta kereshetőség a viselkedés és a tulajdonságok alapján, mert az ezeket leíró állítmányok (igék) és a hozzájuk tartozó tárgyak párosítva vannak, ami pontosabbá teheti a keresést. Számos állítmányt rendelhetünk egy tervezési mintához és tetszőleges számú tárgyat rendelhetünk az állítmányokhoz és fordítva.

A megközelítés másik előnye az, hogy nem csak a tervezési minta globális viselkedését és tulajdonságait írhatjuk le, hanem annak tetszőleges részeit is jellemezhetjük és a keresés során le is kérdezhetjük. Ha az adatbázis nagy mennyiségű, a metamodell alapján kidolgozott mintát tartalmaz, akkor a fejlesztők nagy valószínűséggel megtalálják a megfelelő mintákat a specifikáció alapján.

Míg egy szöveges specifikáció általában a „MIT” kérdésre ad választ, egy tervezési minta esetében a „HOGYAN” kérdés megválaszolásán van a hangsúly. A megközelítés harmadik előnye, hogy ezt a hézagot segít kitölteni, hiszen a „MIT” kérdésre a választ a minta esetében is segít megválaszolni.

Hivatkozások

- [1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York, NY, 1995.
- [2] J. O. Coplien and D. C. Schmidt, Eds., "Pattern Languages of Program Design", Addison-Wesley, Reading, MA, 1995.
- [3] A. H. Edén, "Lepus: A visual formalism for object-oriented architectures", in *The 6th World Conference on Integrated Design and Process Technology, Pasadena, CA, June 26-30, 2002*.
- [4] A. L. Guennec, G. Sunyé, and J.-M. Jézéquel, "Precise modeling of design patterns" in *UML 2000 - The Unified Modeling Language. Advancing the Standard, Third International Conference, York, UK, October 2000, Proceedings*, A. Evans, S. Kent, and B. Selic, Eds., vol. 1939 of *LNCIS*, pp. 482-496. Springer, 2000.
- [5] Object Management Group, *Object Constraint Language*, 2002.

Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, Dr. Pataricza Andrásnak, aki tanácsaival segítette munkám előrehaladását.

Petri Dániel doktorandusz hallgató

Budapesti Műszaki és Gazdaságtudományi Egyetem,

H-1117 Budapest, Magyar tudósok körútja 2.

Tel.: +36-1-4633579, Fax: +36-1-4632667, E-mail: dpetri@mit.bme.hu