Agilis Szoftverfejlesztők Egyesülete előadás (Scrum, DDD), 2010. január 14.

By Marhefka István | Published 2010. 01. 15.

A mai nap résztvettem az Agilis Szoftverfejlesztők Egyesületének klubdélutánján. Kb. 15-en voltunk, egészen családias volt a hangulat. Nekem úgy tűnt, hogy sok ember már régóta ismeri egymást.

Nem ismerem az egyesületet, de szimpatikus a kezdeményezésük, hogy széles körben népszerűsítsék az agilis elveket. Mint megtudtam, ezek a klubdélutánok rendszeresek, a következő egy hónap múlva lesz újra.

Az estének két témája volt: az első a *Scrum*-hoz és a becsléshez kapcsolódik, a második pedig a *Domain Driven Design*-hoz (szakterület vezérelt tervezés). Mindkét téma alapból érdekel.

Időalapú tervezés kontra komplexitás alapú tervezés

Az előadó Dr. Bodó Árpád Zsolt volt. Rövid bemutatkozásként megtudthattuk, hogy a *Sprint Consulting* vezetője, és évek óta segít fejlesztői csapatoknak *Scrum*-bevezetésben. Érdekességképpen megemlítette, hogy 5-6 éve saját erőből próbálták a *Scrum*-ot és az agilis módszertanokat meghonosítani, de miután egyik kudarc a másikat követte, arra a felismerésre kellett jutniuk, hogy megfelelő továbbképzés nélkül ez nem lehetséges. A tréning elvégzése után *Certified Scrummaster* bizonyítványt szerzett, és végre megértette, hogy miről szól ez az egész, és hogyan lehet sikeresen alkalmazni a gyakorlatban a módszert.

Zsolt egy rövid *Scrum*-os összefoglalóval kezdett azok kedvéért, akik nem ismerik. Szerintem ez alatt az 1-2 perc alatt elég jól sikerült az összefoglalás, bár kérdés, hogy azok, akiknek ez új volt, bírták-e követni. Azért azt hiányoltam, hogy a *Scrum*-folyamat lépéseinél kimaradt a demó utáni *retrospective*, és nem kapott kellő hangsúlyt a *Scrum*-ban a tanulás.

Az előadás a becslésről szólt, azaz, hogy hogyan becsüljük meg a *product backlog* sztorijait, ill. egy sprinttervezésen a lebontott feladatokat. Természetes reakciónak tűnik, hogy az idő a megfelelő módszer, a standard kérdés az lehetne, hogy hány óráig/napig tart lefejleszteni egy-egy funkciót. Ezzel azonban több gond is van: egyrészt az egyes emberek gyorsabban, mások lassabban dolgoznak. Valahol átlagosan egy nap 4 órát, a jobb(?) helyeken 6 órát dolgoznak a napi 8 munkaórából. Megtudtuk azt is, hogy ha ismert a határidő, akkor az időbecslések hajlamosak olyan irányba ferdülni, hogy határidőre teljesíthetőek legyenek a feladatok (pszichológiai hatás). Az időbeli becslés tehát az első gondolat ellenére nem túl precíz.

Jobb hatásfokot érhetünk el, ha alternatív módszerként az agilis módszertanokban elterjedt relatív becslést alkalmazzuk. A lényege, hogy veszünk egy olyan feladatot, amelyet mindenki ismer a csapatból, ezt etalonnak nevezzük ki, és ennek a komplexitásához, méretéhez képest viszonyítjuk a kérdéses, becsülendő feladatot. A becslést a csapat közösen végzi a *planning poker* segítségével. A mértékegység ebben az esetben nem idő, hanem egy dimenzionélküli szám. (Amelyet nevezhetünk sztoripontnak, ahogy a *Scrum* teszi.) Így tehát függetleníthetjük magunkat az időtől, elvonatkoztatunk tőle, és egy absztrakt mértékegységben végezzük a becslést.

Ugyan az előadó hangsúlyozta, hogy az idővel nem kalkulálunk, azért valljuk be, pontosan erről van szó, csak egy más formában. (Minek becsülnénk, ha nem arra lennénk kíváncsiak, hogy mennyi ideig tart a feladat?) A becsült komplexitást egyenesen arányosan az időnek feleltetjük meg: ha van egy kéthetes sprintünk, tudnunk kell, hogy eközben a csapat pl. 30 sztoripontot tud elvégezni. Tehát, akármennyire is becsüljük a feladatokat sztoriponttal, 30 sztoripont = 2 hét, azaz 3 sztoripont = 1 nap. A lényeg, hogy a becslés pillanatában felejtsük el az időt, és relatív becslést alkalmazzunk egy már ismert feladat alapján.

Aki új volt a *Scrum*-ban, az hallhatott arról, hogy nem csak idő alapon becsülhetünk, hanem attól elvonatkoztatva, aki pedig már ismerte a problémakört, az mélyíthette ismereteit. Nekem például új volt az, hogy eredetileg a *Scrum*-ban idő alapon történt a becslés, és a fejlődése során később tértek át a relatív becslésre.

Az előadás lendületes, lényegretörő volt. Az előadó a sok személyes példa által számomra hitelesnek tűnt, és az is érződött, hogy már sokszor adott elő.

Domain Driven Design szituációs játék

Bevallom őszintén, hogy amikor decemberben erre a klubdélutánra bukkantam, és azt olvastam, hogy ennek a szituációs játéknak a témája az, hogy egy szakterületi tudást hogyan alakítsunk *UML*-modellé, letettem arról, hogy eljöjjek. Én nem hiszek ebben a megközelítésben. Végül Csuti meggyőzött, hogy nézzünk el.

Ez nem egy előadás, inkább egy interaktív *"show"* volt. Zsuffa Zsolt először egy rövid bevezetőt tartott. Említésre került, hogy a *DDD* atyja *Eric Evans*, akinek a könyvét én is minden fejlesztőnek ajánlom, akit érdekel az, hogy hogyan lehet a komplex üzleti problémákat jól megoldani.

Zsolt kiemelte, hogy a *DDD* lényege a *Domain Model,* a paradigma hátterében pedig az objektum orientált programozás és az *UML* áll – bár bevallotta, hogy ezt így nyíltan (hogy az *UML* a *DDD* része lenne) *Evans* soha nem állította. És ebben nagyon igaza is van! A könyv és Evans megnyilvánulásai ugyanis mind azt hangsúlyozzák, hogy az *UML* csupán egy eszköz, amely **segítheti a kommunikációt**, de semmiképpen nem az a megjelenési forma, amely elsődlegesen közli a *Domain Model*-t. Ugyanis **az maga a kód**. Az *OOP*-vel kapcsolatban pedig az a helyzet, hogy leginkább ez a paradigma alkalmas ma arra, hogy a *Domain Model*-ben rejlő nagyfokú komplexitást a leghatékonyabban kezeljük. Maga *Evans* is emleget más paradigmákat a könyvében (pl. logikai programozás), de arra a következtetésre jut, hogy nem alkalmasak a gyakorlatban a felmerülő problémák hatékony kezelésére. Tehát a *DDD* nem igazán az *OOP*-ről szól, annak ellenére, hogy a megvalósítás során az *OOP*-t használja, mint eddigi legalkalmasabb leíró paradigmát.

Zsolt elképzelésének hátterében az áll, hogy amikor a fejlesztő (elemző) az ügyféllel beszélget, akkor a megismert fogalmak és az ezek közötti összefüggések alapján egy *UML*-osztálydiagramot célszerű készíteni. Jó ötletnek tartottam, hogy ezt élőben is prezentálta: a Magyar Szabadalmi Hivatal egy szakértőjét hívta meg, aki a szabadalmak, oltalmak, ügyek, beadványok, kérelmek, intézkedések, eljárások stb. világába vezetett el minket. A szakértő az általunk feltett kérdésekre válaszolt, ez alapján készült folyamatosan az ábra. A beszélgetés során feltárásra kerültek a fogalmak és a közöttük lévő kapcsolatok (pl. egy beadvány leír egy vagy több találmányt). Személy szerint sajnáltam, hogy a szakértővel történő beszélgetést félbe kellett hagyni, mert felizzott bennem a rejtett Sylar-képességem: úgy éreztem, nem volt elég az információból, meg akarom érteni az egészet! 😃

Zsolt a lerajzolt *UML*-osztálydiagramot méltatta. Erős kritikai érzékem fellángolt:

- *"Az ügyfél előtt rajzolunk, ezáltal ő is látja, érti, miről beszélünk."* – Mi is próbálkoztunk ezzel, de nem igazán jött be a dolog. Pedig semmi rosszat nem mondhattunk az ügyfélre. Jó gondolkozása volt, csak nem az *OOP* volt az élete. Meggyőződésem, hogy doksikba is teljesen felesleges *UML*-t rajzolni a mezei ügyfélnek, ha kövspecről van szó. (Lehet, hogy a bankos rendszerszervezők megértik, de a többség nem, és nem is érdekli.) Ha már rajzolni akarunk nekik, akkor jobban tetszik a gondolattérképes megközelítés. Mi már csak arra használjuk az osztálydiagramokat, ha valamilyen konkrét megoldást akarunk felvázolni egymásnak (fejlesztők). Utána letöröljük a táblát,

és tekodoljuk.

- *"Ez a követelményspecifikáció!"* – Ezt azért erős túlzásnak tartom. Valaki meg is jegyezte, hogy léteznek nemfunkcionális követelmények is, és azok nem ábrázolhatóak osztálydiagrammal. Másrészt pedig hiába térképeztük fel az üzleti fogalomrendszert, és rajzoltunk hozzá osztálydiagramot, ez nem sokat jelent az ügyfélnek: egyrészt ő tisztában van a saját fogalmaival, nem igazán érdekli a modell sem, másrészt kicsit nehezen bólint rá egy kövspecben egy osztálydiagramra, ami egyszerre 20 entitást ábrázol, és amin az entitások közötti kapcsolatrendszer leginkább egy szabásmintára emlékezteti. Másrészt egy osztálydiagram vajmi kevés. Egy statikus képet ábrázol, semmi folyamat nincs benne. Erre más leírások is kellenek (*user story*-k, *use-case*-ek, folyamatábrák – ki miben hisz).
- *"Az osztálydiagramból a megfelelő modellező eszközzel legenerálható a forráskód. És ez nagy előny, mert* ***a domain model a legfontosabb!****"* Igen, legenerálható, de semmi haszna nincs. Mit érünk vele, hogy egy 20-30 entitásból álló *Domain Model* osztályait nem kézzel, hanem automatikusan generálva hozzuk létre? A kapcsolatokhoz létrejön egy *Set* vagy egy *List*, és akkor mi van? Ez a forráskódnak csak elenyésző része, hiszen ez csak az entitások osztályfejeinek definíciója + néhány mező deklarációja. A forráskód legnagyobb része magában a metódusokban van. Ha változik a *domain model*, nincs lehetőségünk újra generálni (hiszen minden, amit utána írtunk bele a kódba, elveszlik/elveszhet), ill. a táblára felrajzolt osztálydiagramot korántsem biztos, hogy egy az egyben kell megfelelteni az osztályoknak. Két dolog jut ezzel kapcsolatban kapásból az eszembe: az öröklődés helytelen használata (rajzon könnyű előrukkolni vele, kódoláskor sokszor hiba használni), a másik pedig, hogy *ORM* alkalmazása esetén bizonyos korlátok közé vagyunk szorítva. (A kódgenerálással kapcsolatban az MDA hívei biztosan másképpen vélekednek, és most megköveznek. Én nem hiszek abban, hogy a módszerük komplex rendszereknél hatékonyan működik.)
- *"Nem attól lesz jó a program, hogy jó a felhasználói felülete, hanem attól, hogy jó a Domain Model-je."* – Kinek mi a jó… Ha nagyon jó a felhasználói felület, akkor az ügyfelet nem érdekli a mögötte lévő kód. Sajnos, ha trehányul van megírva egy kód, akkor ott minőségi és megbízhatósági problémák fognak fellépni. Sok esetben a trehány kódból következik, hogy a felhasználói felület sem lesz különb. Nem szól ez másról csak az igényességről.

Az volt az érzésem, hogy a hallgatóság igen nehezen fogta Zsolt szavait (ez leginkább az oda nem illő kérdésekből látszott). Nem volt túl összeszedett, kicsit csapongó volt, és önmagának is ellentmondott időként – ezt mintha el is ismerte volna 😃 (*"Ha lenne pénz, vessző, paripa, akkor tökéletesen meg lehetne csinálni a kövspecet."* – Nem igaz, mert az ügyfél nem tudja, hogy pontosan mit akar! Ez így ellentmond az agilis megközelítésnek: fejlesszünk iteratívan és inkrementálisan.).

Én jórészt nem értettem vele egyet. Láttam, hogy lelkes, és várja a visszajelzéseket, de szerintem csak a felszínt kapargatta. Sajnáltam, hogy ezt a kedvenc témakörömet nem sikerült illően evangelizálni.

A klubdélutánról felvétel készült. Remélem, közzéteszik az egyesület honlapján.