Az mbeddr és az okosvillanyóra

Szoftver verifikáció és validáció
Házi feladat

Írásos összefoglaló és saját tapasztalat „Markus Voelter, Daniel Ratiu, Bernd Kolb, Bernhard Schaetz: mbeddr: Instantiating a Language Workbenchin the Embedded Software Domain”[1], valamint „Markus Voelter és Bernd Kolb: Smart Meter: an mbeddr Case Study”[2] és „Markus Voelte: Preliminary Experience of using mbeddr for Developing Embedded Software ”[3] című műve alapján

Készítette: Majdán András

1. Bevezetés

Az mbeddr egy olyan language workbench (lásd 2. fejezet) és integrált fejlesztői környezet, amely növelni kívánja a beágyazott szoftverfejlesztés produktivitását. A szerzők – akik egyetemi kutatók, illetve az itemis cég alkalmazottai és egyben az mbeddr fejlesztői – ezt a környezet használták fel egy éles projektben, a Smart Meter fejlesztésekor.

2. Mi az a language workbench?

Martin Fowler szerint [4] egy olyan rendszer ahol:

- A felhasználók tetszőleges nyelveket definiálhatnak és azok együttműködnek egymással
- Az információ forrása egy absztrakt reprezentáció
- Egy nyelvnek három fő eleme van: séma, szerkesztő(k) és generátor(ok)
- A felhasználó projekciós szerkesztőprogramon (lásd 4.2 fejezet) keresztül képes a programot fejleszteni
- Tartalmazhat részleges vagy ellentmondó információkat az absztrakt reprezentációban

Az mbeddr is egy ilyen nyílt forráskódú language workbench-re épül, a Jetbrains által fejlesztett MPS-re. A mbeddr akár tekinthető az MPS beágyazott szoftverfejlesztésre specializált változataként is.

3. Mi az a Smart Meter?

A Smart Meter egy itemis által fejlesztett háromfázisú okosvillanyóra, ami folyamatosan naplózza a fogyasztási adatokat és továbbítja a szolgáltató fele, aki így monitorozni és számlázni is tud utána. Ez a rendszer nem nyílt forráskódú, így csak a szerzők különböző cikkeiben való leírásokra lehet támaszkodni. Hardver alapja egy MSP-430 mikrokontroller, a rajta futtatott szoftver pedig 2014-ben kb. 45 000 sorra volt tehető.

4. Miért van szükség erre a language workbench-re?

A projekt meglévő C forráskódja minőségileg kifogásolható volt, ezért a refaktorációt inkrementálisan végezve szép lassan minden kódot átírtak az mbeddr által kínált absztrakcióra. Az

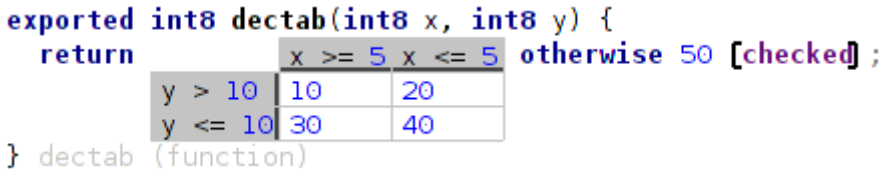
mbeddr C kiterjesztései lehetővé tették a kód átláthatóságának javítását és az eszköz tanúsítása felé vezető utat. A projekciós szerkesztési mód (lásd 4.2 fejezet) is sokat javít a kód átláthatóságán és a fejlesztés gyorsaságán.

4.1 Nyelvi kiterjesztések

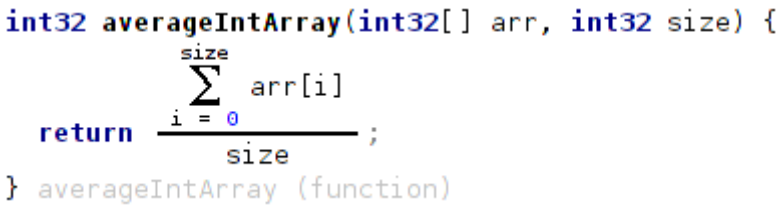
Az MPS-ben saját nyelveket definiálhatunk, gyakorlatilag az mbeddr is hozzáadott nyelvekből, modell ellenőrzési és dokumentálási eszközökből áll. A mbeddr nyelvi kiterjesztései modulárisak és integrálhatóak akár egy forráskódban. Tehát ugyanabban a fájlban lehet például egy állapotgép és egy C forráskód is (mindkettőnek megvan a maga nyelve), az egymásra való hivatkozás is megoldott. A fordítás végén persze minden nyelv a transzformáció(ka)t követően C-re fordul, de ez szemben áll hagyományos megközelítéssel, ahol különböző fordítókat alkalmaznak, majd linkelik a keletkezett binárisokat.

4.2 Projekciós szerkesztés

Az MPS egyik fő előnye a projekciós szerkesztés (és megjelenítés), ahol is a különböző nyelvi konstrukcióknak különböző megjelenítési módjai is lehetnek (1.ábra és 2.ábra).



1.ábra: Döntési tábla ábrázolása



2.ábra: Matematikai képlet ábrázolása

Ezt úgy éri el az MPS, hogy a forráskód nem egy egyszerű szöveges állomány még szerkesztés közben sem, hanem AST (abstract syntax tree) amely egy strukturált XML fájlba mentődik. A szerkesztéskor intelligens kiegészítés, illetve az adott pozícióban elérhető nyelvi elemek felsorolása is igénybe vehető. A másolás és beillesztés művelet is végezhető a fa egy ágára. A Smart Meter egy nagyjából 109 ezer csomópontú AST.

5. A Smart Meter fejlesztése

A cég a fejlesztés során az mbeddr által kínált nyelvi alapkiterjesztések mellé saját nyelvi kiterjesztéseket is írt, amelyek nagy részét az mbeddr újabb verzióiban nyílt forráskóddal elérhetővé tette, így már harmadik felek is használhatják.

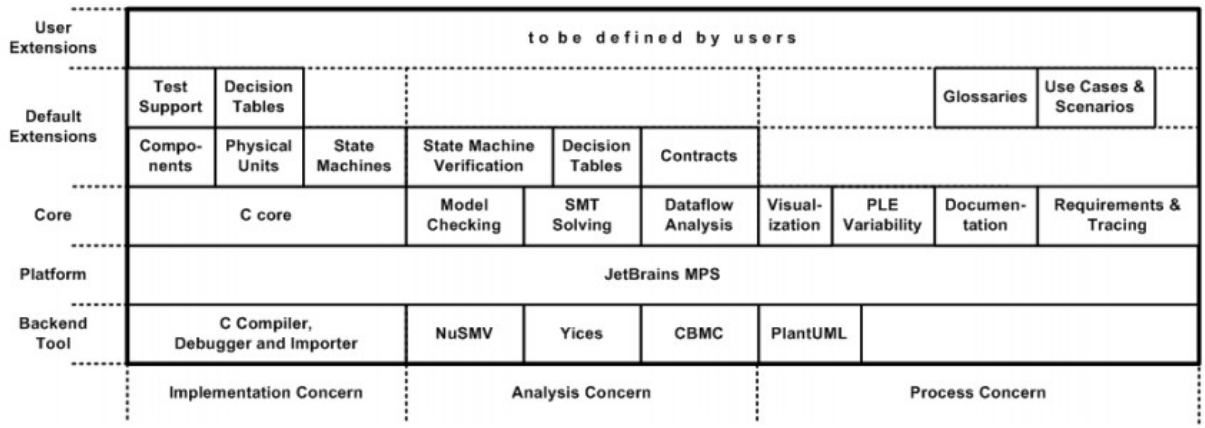
A projekt által hasznosított nyelvi kiterjesztések:

- *Komponensek*: Interfészeknek (funkciók prototípusai) az implementációi. A program struktúráját javítják azzal, hogy elválasztják a specifikációt a különböző megvalósításoktól.
- *Állapotgépek*: A különböző kommunikációs protokollok állapotgépként sokkal jobban olvashatóak és értelmezhetőek, mintha C-ben lennének megírva. A projekt során ezek átirata is folyamatban van.
- *Elvárások nyomon követése (Requirements Tracing):* Ezzel lehet nyomon követni, az ügyfél igényeit és azok megvalósítását (verifikációt támogató nyelvi elem).
- *Mértékegységek*: a Smart Meter különböző fizikai mértékegységekkel végez műveleteket, amelyek nyelvi támogatása növeli az olvashatóságát és csökkenti a hibalehetőségeket (más-más mértékegység feltételezése, konverziós hibák). [1] A szoftver 102 mértékegység deklarációt és 155 konverziós szabály tartalmaz.
- *Regiszterek*: egy valós beágyazott rendszerben gondolni kell az alacsonyabb szintű I/O műveletekre is, mint például regiszterek olvasása és írása. Erre külön nyelvi kiterjesztést hoztak létre a készítők, amely nem csak definiálja ezeket a regisztereket, hanem működésüket is szimulálja, ha azokon valamilyen számítás hajtódik végre. Így a különböző szenzorok olvasása és az UART kommunikációs interfész is megvalósítható.
- *Megszakítások*: a processzorban megszakítások jöhetnek létre (például időzítés vagy külső esemény hatására), ezek kezelésére is létrehoztak egy új kiterjesztést, amely megszakítás érkezése esetén képes meghívni a hozzá tartozó megszakítás kezelőt (a komponens egy metódusát).
- *Emuláció*: a megszakításokat meg kell hívni valahogy, mert a PC-n való futtatáskor nem fog jelentkezni az adott esemény. Erre szolgál az emulációs réteg, amely bizonyos feltételekre kiváltja ezeket (felhasználói interakció, regiszter változása).

Az mbeddr architektúrából (3. ábra) látható hogy az mbeddr eszközkészletének töredékét vette igénybe csak a projekt. Ez talán annak is tudható, hogy a Smart Meter kódját, nem elölről írták, hanem egy szolgáltatótól kapott kódot kellett iteratívan refaktorálni, így az újabb elemek felhasználása a jövőben várható, valamint egyes nyelvek (például a dokumentációs nyelv) jelenleg is fejlesztés alatt van.

---

1) Az 1998-as Mars Climate Orbiter fatális meghibásodása óta ismert, hogyan lehet 125 millió dollárt elveszteni egy mértékegység tévedésen, ahol is metrikus egység helyett birodalmit használtak az egyik modulban [5].



3. ábra: Az mbeddr stack az MPS language workbench-re épülve

6. Szoftver tesztelés és verifikálás a Smart Meter fejlesztése során

54 teszt esetet vizsgáltak és 1415 asszertációt alkalmaztak a hibák elkerülésére. Az eredeti C kódban lévő egymásba ágyazott feltételes (if) szerkezeteket döntési táblába (1.ábra) írták át a fejlesztők és így a kódot teljességre és determinizmusra is tudták ellenőrizni. A protokollokat leíró állapotgépeket is modell ellenőrzésnek vetették alá, amely során találtak több hibát is, amelyeket a refaktoráció során követtek el (amikor is a C kódot állapotgép nyelvre írták át). A mértékegységeket bevezető nyelvi kiegészítő alkalmazásakor is felütötte magát egy számítási hiba, ahol is egy hőmérséklet értéket négyzetre emelő kód önmagának adott értéket (T = T*T). A mértékegység Kelvin volt és azonnal kibukott, hogy egy Kelvin mértékegységű változó nem vehet fel Kelvin*Kelvin mértékegységű értéket.

A tesztesetekre és asszertációkra külön nyelvi kiterjesztéssel rendelkezik az mbeddr, a verifikációra pedig három eszköz is rendelkezésre áll beépített támogatással az IDE-ben:

- NuSMV

  A NuSMV egy szimbolikus modellellenőrző, amely kombinálja a BDD és SAT alapú modellellenőrzést. Az mbeddr-ben ezzel verifikálják az állapotgépeket, melyre az automatikus ellenőrzésen (minden állapot elérhető-e, minden átmenet végbemehet) kívül lehetőség van temporális logikában (LTL vagy CTL) meghatározni a tulajdonságokat.

- Yices

  A Yices egy SMT megoldó, ami szimbólumok esetén is ismeri az egyenlőséget és a lineáris aritmetikát. A döntési táblák konzisztenciájának és teljességének eldöntésére használják az mbeddr-ben.

- CBMC

  Korlátos modellellenőrző C és C++ programokra. Képes kiszűrni a puffer túlcsordulásokat (tömb korlátok) valamint a felhasználó által specifikált asszertációkat is ellenőrzi. Mindezt a ciklusok okos kifejtésével éri el.

7. Konklúzió

Az mbeddr moduláris nyelvi kiegészítőinek használata sikeresnek bizonyult a Smart Meter projektben. Az új nyelvi kiegészítések másfél nap alatt íródtak (regiszterek, megszakítások), ami nagyon gyorsnak számít egy ekkora projektben. Ezekkel nem csak a biztonság, hanem a kód tesztelhetősége is javult.

A generált C kód elég hatékonynak bizonyult, annak ellenére a magasabb absztrakciók erőforrás és teljesítmény problémákat okozhatnak. Az eredeti C kódra tervezett hardveren megfelelően futott az új kód. A jelenlegi C kód importálása azonban nehéz feladatnak bizonyult, ugyanis csak a header fájlokra van importálási lehetőség az mbeddr-ben és akkor sem tudja kezelni a makrókat (például #ifdef).

Az mbeddr használata akár néhány nap alatt elsajátítható, bár a sok nyelv kulcsszavainak megtanulása időt vehet igénybe, de erre talán nincsen is szükség az intelligens kódkiegészítés és a nyelvek modularitása miatt. A projekciós szerkesztési mód megszokást igényel, de elméletben gyorsíthatja a fejlesztést.

8. Saját tapasztalatok

Egy beágyazott fejlesztési folyamat során magam is találkoztam az mbeddr rendszerrel, amely tapasztalatom alapján – jelen formájában – valós rendszer fejlesztésére harmadik fél számára nem alkalmas, csak legfeljebb az mbeddr fejlesztőinek, akik ki tudják javítani a rendszer hibáit, ha találkoznak velük. A szerzők, akik az összefoglalóm alapjául szolgáló cikkeket írták az mbeddr fejlesztői, így a leírás nem objektív.

Az mbeddr alapjául szolgáló JetBrains MPS egy jól megírt és használható metaprogramozó rendszer. Egyetlen hibája talán az AST szerkesztés szövegszerű projekciójában van, amelyet nehéz megszokni (ezt az mbeddr fejlesztői is elismerik).

Maga az mbeddr kiegészítés, amit az MPS-hez kiadtak és jelenleg a 0.9-es verziószámot viseli még kezdetleges, ráadásul telepítője sincsen. A 0.8.1-es verzióban (utolsó telepítős változat) nem lehet az állapotgépeknek eseményt küldeni, se állapotot beállítani C kódból egy bug miatt. Bár elvileg a verziók között lehet projekteket hordozni, azonban ez már a mellékelt példa kódoknál sem működik. Például egy egyszerű main függvényben a régi „string” típust nem volt képes fordítani az új „string”-re. Bár a tanító videók és a cikkek is NuSMV-vel verifikálnak, ez utoljára a 0.6-os verzióban működött, mert utána ugyan mellékelték a programot, de hiányzik a megfelelő devkit. Ennek oka, hogy a fejlesztők az újabb verziókban a CBMC-re fókuszálnak. Ez azért is probléma, mert a 0.6-os verzióban állapotgépek esetén a verifikálás nem támogatott kompozit állapotok esetén. Az alkalmazott szintaktikán is változtattak a verzióváltások közben, ami szintén produktivitás romboló. Mindez átgondolatlanságra és a tervezés hiányára utal a nyelvek felépítésekor.

Tehát bár az mbeddr irány – véleményem szerint is – a beágyazott szoftverfejlesztés jövője, még nem áll készen arra, hogy valós feladatra használják a hibái, hiányosságai miatt. Elnézve azonban a fejlesztés üzemét, néhány év múlva akár a beágyazott szoftverfejlesztés Eclipse-e lehet, így érdemes foglalkozni vele.

Irodalomjegyzék

[1] M. Voelter, D. Ratiu, B. Kolb, B. Schaetz: mbeddr: Instantiating a Language Workbench in the Embedded Software Domain, Automated Software Engineering, 2013, Vol.20, Issue 3, pp 339-390.

[2] M. Voelter és B. Kolb: Smart Meter: an mbeddr Case Study http://mbeddr.com/files/mbeddr_casestudy_smartmeter.pdf (2012)

[3] M. Voelte: Preliminary Experience of using mbeddr for Developing Embedded Software, in 10th Dagstuhl Workshop on Model-based Development of Embedded Systems, 2014, p.10.

[4] M. Fowler: Language Workbenches: The Killer-App for Domain Specific Languages? http://martinfowler.com/articles/languageWorkbench.html (2005)

[5] Douglas Isbell, Don Savage: Mars Climate Orbiter failure board releases report, numerous NASA actions underway in response, (1999) http://mars.jpl.nasa.gov/msp98/news/mco991110.html