Önálló laboratórium 2 beszámoló

Szokodi Ákos
Konzulens: Dr. Pataricza András

2014. december 19.

Tartalomjegyzék

1. Kivonat

Manapság egy üzleti alkalmazás fejlesztésekor kérdéses, hogy milyen platformokat támogasson az alkalmazás.

A felhasználói felületek készítése igen költséges, ezért a fejlesztők a lehető leggenerikusabb megoldást választják általában, mely kompromisszumokkal járhat (felhasználói élmény hiánya, teljesítmény csökkenés, stb. . .).

Felmerül az igény a felhasználói felületek fejlesztésének automatizálására, mely során egy platformfüggetlen modellezési nyelv segítségével elkészített modellből tetszőleges operációs rendszerre és kijelzőméretre lehessen kódot generálni.

Az önálló labor során megvizsgáltam a különböző felhasználói felület modellezési nyelveket (Conceptual Interface Patterns, Just-UI, ConcurTaskTrees, Interaction Flow Modeling Language), majd összehasonlítottam őket.

A legmegfelelőbbnek bizonyult nyelvet (IFML) tovább vizsgáltam, illetve egy példaalkalmazást (példa IFML modellt és hozzá egy kódgenerátort) próbáltam készíteni az IFML modellező nyelvhez tartozó Eclipse plug-in segítségével, az IFML Editor-ban.

Problémák merültek fel a demo alkalmazás készítésekor, de a GitHub-on az IFML Editor fejlesztői készséggel segítettek és kijavítottak egy – az eszközben előkerült hibát.

Áttekintettem a lehetséges további feladatokat, melyek között a MARIAE és MARIA XML technológiák megismerése, valamint a munkám végső céljaként elkészülő rendszer ismertetése szerepel.

2. A modell alapú felület tervezés igénye

Egy modern üzleti alkalmazás fejlesztése során felmerül a kérdés, hogy mely platformokat támogatására kerüljön sor. Manapság a felhasználók az eszközök igen széles skálájából választhatnak mind a készülékek mérete és az eszközön futó operációs rendszer tekintetében.

Az üzleti alkalmazásokhoz tartozó kliens fejlesztése az eltérő platformokra és kijelzőméretekre időigényes, ezért gyakori megoldás, hogy az alkalmazásnak egy web alapú (HTML5 és JavaScript technológiákkal fejlesztett) kliensét készítik el.

A web alapú megoldásban különböző problémák merülhetnek fel:

- Teljesítményigényes alkalmazás esetén a böngészőben futtatott webes megoldás gyakran a felhasználói élmény rovására mehet (lassulás lép fel az alkalmazásban, stb. . .).
- Mivel minden platformon ugyanazon webes klienst érik el a felhasználók, előfordulhat, hogy az alkalmazás kinézete nem illeszkedik bele az operációs rendszer stílusába, nem felel meg az úgynevezett platform specifikus ,,guideline-oknak”.
- Bár a reszponzív fejlesztési technika segítségével elkészíthetők a webes alkalmazás felületének különböző kijelzőhöz optimalizált variánsai, nehéz a megjelenítendő objektumok megfelelő elrendezése (például az összetartozó felületi elemek kijelzőmérettől függetlenül legyenek egymáshoz közel, a lehető legkevesebb navigáció közbeiktatásával).

Amennyiben a web alapú kliensalkalmazás nem kielégítő a fenti problémák miatt, minden platformhoz (és minden kijelzőmérethez) külön meg kell tervezni és el kell készíteni a platformspecifikus klienst.

A felület tervezése és fejlesztése költséges munka, így felmerül az igény a folyamat automatizálására. Ehhez szükséges egy platformfüggetlen nyelv, mely segítségével úgy lehet leírni egy alkalmazás felületét, hogy ebből generátorok segítségével különböző kijelzőméretre és különböző platformokra lehet leképezni a felületet.

Az önálló laboratóriumom során ezt a témakört próbáltam kutatni. Megvizsgáltam a létező platformfüggetlen felületleíró modellezési nyelveket, összehasonlítottam azokat és kiválasztottam a célnak legmegfelelőbb nyelvet, melyet aztán tovább vizsgáltam.

3. Felületleíró modellezési nyelvek

Manapság több felületleíró modellezési nyelv is létezik. Ezek támogatottsága változó, néhányat már nem fejlesztenek, illetve eszköztámogatottságuk is megszűnt.

A modellezési nyelvek közül kiválasztottam néhány elterjedtebb, még manapság is használt nyelvet, melyeket megvizsgáltam majd összehasonlítottam egymással. Ezen modellezési nyelvek a Conceptional Interface Patterns, ConcurTaskTrees és az Interaction Flow Modeling Language.

3.1. Conceptual Interface Patterns és Just-UI

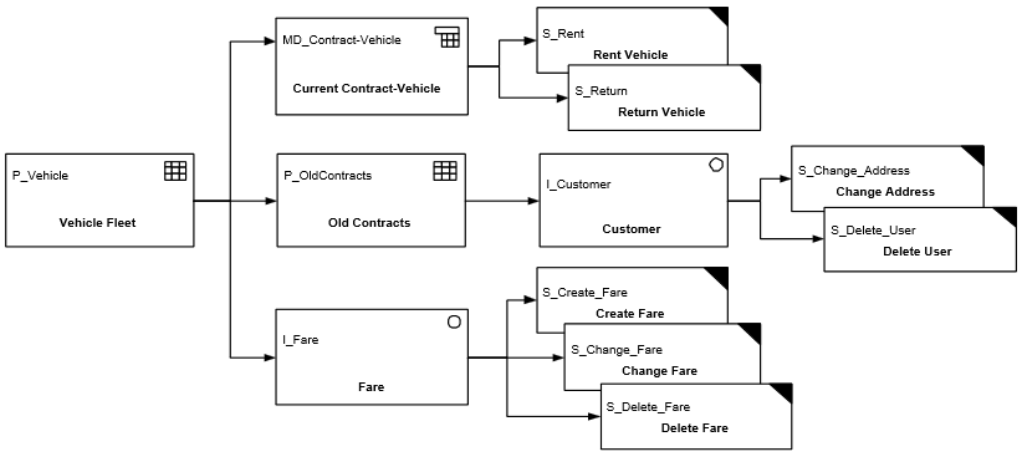
A Conceptual Interface Patterns [1] modellezési nyelv különböző mintákat definiál a felhasználói felületekre, melyek segítségével koncepcionális szinten tudjuk modellezni a felhasználói felületet.

A mintákat négy nagyobb csoportba sorolja a nyelv:

- Master/Detail minták: két nézettel rendelkező minták, melyben az egyik nézet több objektumot jelenít meg (master), míg a másik a master nézetből kiválasztott objektum részleteit mutatja a felhasználónak (detail).
- Instance minták: üzleti objektumok elérését (olvasását/módosítását) teszik lehetővé.
- Service minták: egy adott szolgáltatás végrehajtásáért felelős minták (például egy – a szerver oldalon definiált – webszolgáltatás meghívása).
- Population minták: több üzleti objektum együttes kezelésére szolgáló minták (például listában megjelenő objektumok közötti rendezést, szűrést megvalósító minták).

A Conceptual Interface Patterns mintákat a Just-UI [2] nevű modellezési nyelv használja fel úgy, hogy a CIP mintákat navigációs kapcsolatokkal köti össze.

A példában látható, hogy a Vehicle Fleet egy Population mintát valósít meg (autók egy listáját tartalmazza).



1. ábra. Egy jármű bérlő alkalmazás menedzselő felületére példa a Just-UI modellezési nyelv segítségével

Itt megnézhetjük Master/Detail nézetben a jelenleg bérelhető járműveket (Current Contract-Vehicle), melyeken különböző szolgáltatásokat hívhat meg az operátor (autó bérlése, visszavitele).

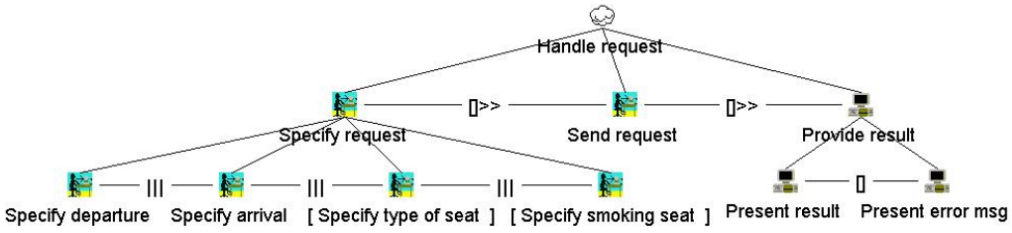
Szintén Population minta segítségével megtekinthetők a már lejárt szerződések (Old Contracts), amikhez egy-egy bérlő megjelenítése tartozik (Customer).

A bérlő adatainak módosításához törléséhez a megfelelő szolgáltatást tudja meghívni az operátor.

3.2. ConcurTaskTrees

A ConcurTaskTrees [3] modellezési nyelv taszkokat és közöttük levő kapcsolatokat definiál, melyek segítségével absztrakt módon írható le egy felhasználói felület.

A nyelvben az egyes taszkok különböző típusúak lehetnek: bemeneti taszkok, amik információt kérnek be a felhasználótól (Specify departure), kimeneti taszkok (Present result), amik megjelenítik az üzleti logika objektumainak attribútumait, illetve komplex taszkok amelyek gyermekei eltérő típusú (bemeneti és kimeneti)



2. ábra. Repülőjegy foglalás CTT-vel modellezve

taszkok (Handle request).

A taszkok között definiálhatunk gyerek-szülő relációkat, melynek jelentése egy példán: „ahhoz, hogy a Handle Request taszk végbe menjen, meg végbe kell mennie a Specify request, Send request és Provide result taszkoknak”. Ezen reláción kívül további testvér-taszkok közötti relációkat is definiál a nyelv:

- Engedélyezés (adat küldéssel vagy küldés nélkül): Az egyik taszk addig nem mehet végbe, amíg a tőle balra levő testvér taszk végbe nem ment (Send request előtt a Specify request-nek végbe kell mennie). Az engedélyezéskor lehetőség van taszkok közötti adatküldésre is.
- Választás: Amennyiben a gyerek taszkok közül pontosan egy végbemegy, úgy a szülő taszk is végbe ment (a Provide result taszk pontosan akkor megy végbe, ha vagy a Present result, vagy a Present error msg taszk végbe ment).
- Konkurencia: A gyerek taszkok között definiált konkurens kapcsolat jelzi, azaz hogy egy időben futhatnak le (például egy időben történhet a Specify departure, Specify arrival, Specify type of seat és Specify smoking seat taszkok végbe menetele, tehát például egy képernyőre is kerülhetnek).

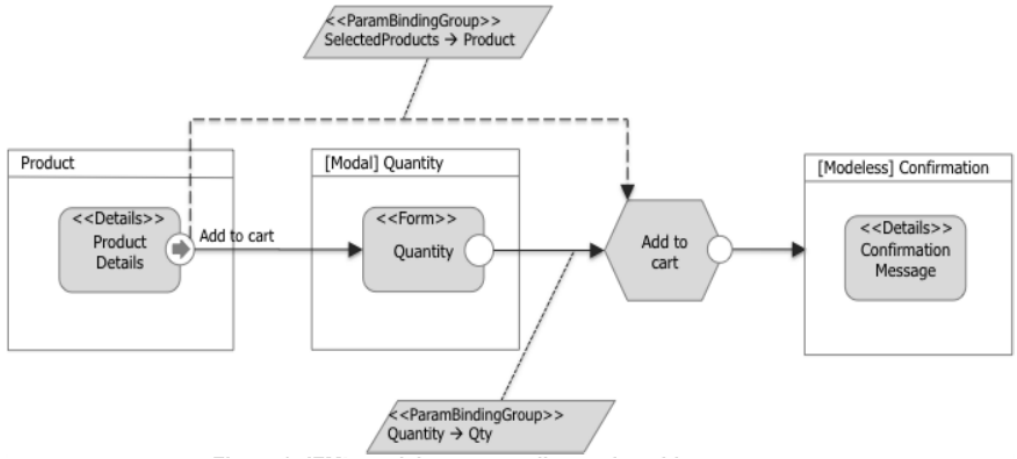
Több különböző felhasználói felület tervező eszköz is a ConcurTaskTrees modellezési nyelvet használja. Ilyen eszközre példa a MARIAE [4] absztrakt és konkrét felhasználói felület modell készítő szoftver. Az önálló labor keretein belül nem került sor az eszköz megvizsgálására.

3.3. Interaction Flow Modeling Language

Az Interaction Flow Modeling Language [5] nyelvben absztrakt felületi elemek segítségével lehet változatos interakciókat és azok hatásait modellezni úgy, hogy közben UML diagram segítségével definiált üzleti objektumokat köthetünk hozzá a felületi modellben definiált elemekhez adatkötés segítségével. Az IFML egy OMG szabványos nyelv, mely jelenleg is béta állapotban van.

A nyelv fejlesztői készítettek egy nyílt forráskódú diagramszerkesztő alkalmazást is, melyet jelenleg is fejlesztenek. A szabvány és szerkesztő eszköz mögött a WebRatio cég áll, akik saját kommerciális alkalmazásukban is implementálták az IFML egy bővített változatát, amivel HTML-re lehet leképezni az elkészített diagramot [6].

Nagy előnye a nyelvnek, hogy könnyen bővíthető. Egy kutatócsoport például mobilos kiegészítést készített a nyelvhez [7], de maga az IFML is modulárisan épül fel (egy úgynevezett „core” nyelvi elemek halmazából és erre épülő kiegészítő elemekből áll).



3. ábra. Egy online könyvesbolt modellezése (részlet: termék kosárba helyezése)

Az IFML-ben a legfelső felületi elem az ablak (Window), melyben számos előre definiált vezérlőt helyezhetünk el. Ilyen vezérlő lehet például a lista nézet,

vagy űrlap.

A vezérlőkhöz a már korábban említett üzleti objektumokat lehet rögzíteni adatkötés segítségével (Data Binding). Az egyes események vagy akciók (Action) hatására egy másik Window elembe térhet át a vezérlés egy navigációs nyíl mentén.

A navigáció során a két Window elem között adat átvitelére is van lehetőség. Ezt a ParameterBinding nyelvi elem végzi: megadható, hogy a navigáció kezdőpontján levő ablakban definiált változó, melyik – a navigáció végpontján levő – változóra képeződjön le.

A példában három Window objektum látható (Product, Quantity, Confirmation). A Product ablakban egy Details vezérlő található, ami jelen esetben egy könyv részletes paramétereit tárolja (az ábrán adatkötés nem látható a példa egyszerűsége miatt).

Amennyiben az Add to Cart esemény bekövetkezik (például a felhasználó megnyomja a felületen a gombot), úgy áttérünk egy modális ablakra, ahol ki kell választania a felhasználónak a vásárolni kívánt könyv mennyiségét.

Ezután a kiválasztott könyv (felső Parameter Binding) és a megadott mennyiség (alsó Parameter Binding) segítségével az Add to Cart akció végbe tud menni (ami például meghív egy webszolgáltatást a kapott paraméterekkel és kosárba helyezi a megfelelő könyvet/könyveket). Ennek sikerességéről végül egy – a vásárlás részleteit tartalmazó – értesítő üzenetet kap a felhasználó.

3.4. Modellezési nyelvek összehasonlítása

A Conceptual Interface Patterns és Just-UI a vizsgáltak szerint túlságosan absztrakt leírók, nehéz a segítségükkel készült modellekhez generátort készíteni. Ezen kívül érdemes megjegyezni, hogy ezen modellezési nyelvek eszköztámogatottsága alacsony.

A ConcurTaskTrees – mint, ahogy a CIP esetében is volt – túl absztrakt modellt állít elő, így a modellből nehezen hajtható végre kódgenerálás. A korábban már említett MARIAE eszköz felhasználja a CTT-t, mely absztrakt felhasználói

felületet épít a nyelv segítségével, ebből készít egy köztes modellt MARIA XML nyelven (platformfüggetlen Concrete felhasználói felület), majd ebből generálható HTML kód asztali számítógép és mobiltelefon rendszerekre. A MARIAE eszköz vizsgálata egy lehetséges továbbhaladása az önálló laboratóriumnak.

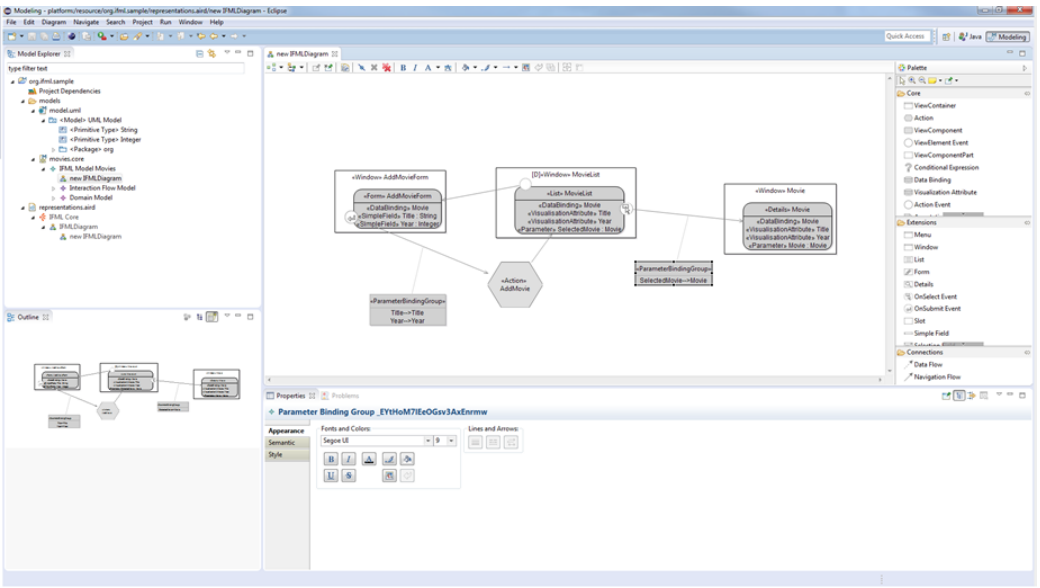
Véleményem szerint az Interaction Flow Modeling Language segítségével lehet platformfüggetlen, de nem túl absztrahált modellt készíteni, melyből könnyen generálható felület. További előnye az IFML-nek, hogy egy OMG szabvány nyelv, kibővíthető, valamint erős eszköztámogatottsága van (nyílt forráskódú, jelenleg is fejlesztés alatt).

Munkám további részében az IFML nyelvvel foglalkoztam és az IFML Editor diagramszerkesztőben készítettem példaalkalmazást.

4. Az IFML Editor

Miután kiválaszottam modellezési nyelvnek az IFML-t, megvizsgáltam a diagramszerkesztő alkalmazását.

Az IFML Editor [8] egy nyílt forráskódú Eclipse plug-in (GitHub-on található a forráskódja). Az IFML modell implementációja EMF technológiával, a diagramszerkesztő felület pedig Sirius technológiával készült.



4. ábra. Az IFML Editor

Munkám során egy egyszerű példa IFML modellt szerettem volna készíteni (megfelelő UML-lel ábrázolt üzleti objektumokkal), de sajnos problémákba ütköztem, ugyanis már a legegyszerűbb IFML modellt sem tartotta megfelelőnek az eszköz validációs mechanizmusa.

Ezután a GitHub-on mellékelt előre elkészített példa UML és IFML diagramokat töltöttem be az IFML Editor-ba, de továbbra is hibásnak ítélte a diagramot az eszköz.

Következő lépésként a szabványban leírt IFML metamodell és az Editor-ban implementált metamodell vizsgálata után sem találtam megoldást a hibára, így

felvettem a kapcsolatot a fejlesztőkkel a GitHub-on.

Egy hibajegyben rögzítettem a problémámat, majd az egyik fejlesztő válaszolt a hibajegyre. Valóban probléma volt az alkalmazásban és a kiadott példa modellben is, így ezeket kijavították, majd az új verzió már valóban helyesnek találta a példa modellt.

A kijavított IFML Editor-ban szerettem volna készíteni egy kódgenerátort a GitHub-on mellékelt példamodellre, de további nehézségekbe ütköztem: a projekt oldalán nem volt mellékelve fejlesztői leírás (például környezet beállítása) az alkalmazás fejlesztéséhez. A fejlesztő ismét válaszolt és egy lépésről-lépésre pontsorozatot írt a fejlesztés menetéről:

1. Eclipse Luna Modeling Tools letöltése (fontos: ne legyen feltelepítve az IFML plug-in ebbe az Eclipse-be)
2. Sirius és Acceleo Eclipse plug-inek telepítése
3. Az IFMLEditor, IFMLEditor.edit és IFMLEditor.editor projektek importálása
4. Új példány Eclipse indítása, ahol importáljuk az IFMLEditorDesign projektet
5. Acceleo kódgenerátor készítése a példány Eclipse-ben

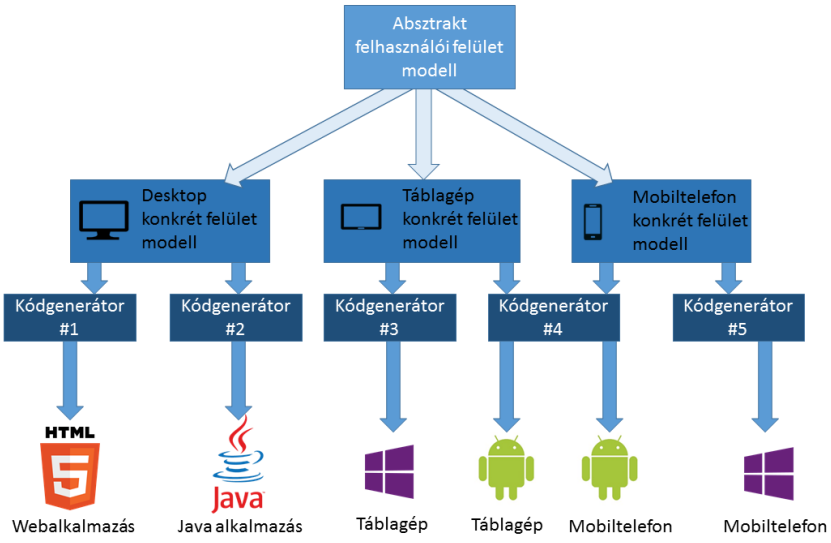
Idő hiányában sajnos már nem tudtam a példagenerátor elkészíteni, ezért ez egy további feladat lesz az önálló labor folytatása kapcsán.

5. További munka

A továbbiakban mindenképp érdemes a korábban már említett MARIAE eszközt és a benne implementált MARIA XML nyelvet megvizsgálni.

A munka végső célja egy olyan alkalmazás fejlesztése, mely három szinten kezeli a felhasználói felületet (a MARIAE eszközhöz hasonlóan).

Az alkalmazás legyen képes egy absztrakt felhasználói felület modellt készíteni (platform és kijelzőméret független) például IFML nyelv (vagy ennek kibővítése) segítségével. Az absztrakt felhasználói felületből modelltranszformáció segítségével elő lehessen állítani egy olyan platformfüggetlen felhasználói felület modellt, ami már kijelző függő elemeket tartalmaz (külön asztali környezetre, táblagépre, mobilkliensre optimalizált platformfüggetlen modellek). Ezen konkrét modellekből pedig kódgenerátorok segítségével elő lehessen állítani platformspecifikus (illetve kijelzőspecifikus) felületet (5. ábra) .



5. ábra. A három szintű felhasználói felület kezelése

A további munka legfontosabb része az absztrakt felhasználói felület modell és konkrét felület modellek közti modelltranszformációs logika végiggondolása

és implementálása például szabály alapokon.

Példaként egy sok elemet tartalmazó absztrakt nézet eltérő módon fog leképeződni asztali, táblagép, illetve mobiltelefon modellre. A cél az, hogy abban az esetben, amikor az elemek nem férnek el egy kisebb kijelzőn, a „feldarabolt" nézetek – melyek összetartoznak – ne essenek túlságosan messze egymástól (a lehető legkevesebb interakcióval el lehessen jutni az egyik elemtől a másikig).

Erre egy lehetséges megoldás például, hogy a mögöttes UML diagramon készített üzleti logikai objektumok közül vizsgáljuk, hogy akik közel állnak egymáshoz (alacsony számú asszociáció mentén érhetőek el például), azok a megjelenítésben is álljanak egymáshoz közel.

6. Összefoglalás

Ebben a félévben megvizsgáltam a különböző modell alapú felhasználói felület tervezési technikákat, valamint modellezési nyelveket (Conceptual Interface Patterns és Just-UI, ConcurTaskTrees, Interaction Flow Modeling Language).

Összehasonlításuk után részletesen megvizsgáltam az IFML-t és az IFML Editor diagramszerkesztő eszközt, melyben a kezdeti problémák után sikerült elérni, hogy az eszközben egy – használhatóságot erősen korlátozó – hibát kijavítsanak és megadjanak egy lépésről-lépésre leírást azok számára, akik szeretnének magán az eszközön javítani, vagy kódgenerátort szeretnének készíteni az IFML modellhez.

Hivatkozások

[1] User Interface Conceptual Patterns - Petro J. Molina, Santiago Meliá, Oscar Pastor
http://pdf.aminer.org/000/162/806/user_interface_conceptual_patterns.pdf

[2] Just-UI: A user interface specification model - Pedro J. Molina, Santiago Meliá, Oscar Pastor
http://www.dlsi.ua.es/ santi/papers/just-ui_CADUI2002.pdf

[3] ConcurTaskTrees: An Engineered Approach to Model-based Design of Interactive Systems - Fabio Paternò
http://giove.isti.cnr.it/attachments/publications/2003-A1-07.pdf

[4] MARIAE - HIIS Laboratory (Istituto Di Scienza E Tecnologie Dell'informazione "A. Faedo"
http://giove.isti.cnr.it/tools/MARIAE/home

[5] Interaction Flow Modeling Language
http://www.omg.org/spec/IFML/1.0/Beta2/PDF/

[6] WebRatio IFML
http://www.webratio.com/portal/content/en/ifml-standard

[7] IFML Mobile extensions presented at MobiWIS
http://automobile.webratio.com/2014/09/ifml-mobile-extensions-presented-at-mobiwis/

[8] Interaction Flow Modeling Language Editor - GitHub
https://github.com/ifml/ifml-editor