# Studiu comparativ privitor la specificarea abstractă a interfeței cu utilizatorul folosind UsiXML şi UIML

**Simina Tofan, Anamaria Pradais, Sabin Buraga**
Facultatea de Informatică, Universitatea „Alexandru Ioan Cuza" din Iaşi, România
Str. Berthelot, nr. 16, Iaşi 700483 - România
**{simina.tofan, anamaria.pradais, busaco}@info.uaic.ro**

**REZUMAT**
Lucrarea îşi propune să pună la dispoziție un studiu referitor la manierele de specificare abstractă a interfeței unui joc interactiv folosind limbajele descriptive UsiXML şi UIML. Sunt prezentate aspecte privind proiectarea interacțiunii bazată de modele în contextul CADUI (*Computer-Aided Design of User Interface*) şi considerații referitoare la utilizarea efectivă a unor instrumente de modelare.

**Cuvinte-cheie**
specificare abstractă, modelare, limbaje de descriere.

**Clasificare ACM**
H5.2. Information interfaces and presentation (e.g., HCI): Miscellaneous.

**PREAMBUL**
Lucrarea de față prezintă un studiu comparativ de analiză a modalităților actuale de specificare abstractă a interfeței cu utilizatorul pe baza a două limbaje descriptive: UsiXML (*User Interface eXtensible Markup Language*) şi UIML (*User Interface Markup Language*).

Problematica se înscrie în contextul metodologiilor de proiectare bazată pe modele, dintre care se distinge MDA (*Model-Driven Architecture*) [6]. Aceste metodologii permit dezvoltarea aplicațiilor interactive complexe. Modelul/modelele facilitează o vedere abstractă a interacțiunii, separând clar maniera de prezentare a conținutului, de partea de procesare şi de stocare efectivă a datelor.

Demersul acestui articol se focalizează asupra proiectării asistate de calculator a interfeței cu utilizatorul – CADUI. Aceasta vizează aplicarea diverselor metodologii, toate având în comun următoarele [4]:

- la nivel abstract, descrierea sistemului interactiv se realizează via un model independent de calculator: CIM (*Computation-Independent Model*). Modelul ia în considerație cerințele – de exemplu, *mission statement*, funcționalitățile, cazuri de utilizare etc. –, ceea ce conduce la formularea şi specificarea cerințelor referitoare la interacțiunea sistemului şi maniera de realizare a sarcinilor, concurente sau nu.
- descrierea independentă de platformă se realizează prin intermediul unui model PIM (*Platform-Independent Model*) care defineşte la nivel conceptual sistemul vizat. Din punctul de vedere al ingineriei software, în cadrul acestui model vor fi specificate modelul obiectual, dinamic, funcțional şi – foarte important în cazul nostru – cel referitor la interacțiunea cu utilizatorul.
- dezvoltare focalizată asupra unei platforme-țintă este modelată grație PSM (*Platform-Specific Model*), proces care necesită utilizarea de instrumente de transformare (compilare), bazate pe un model al aplicației (*application model*).
- implementarea efectivă urmează un model de programare, numit CM (*Code Model*), vizând crearea/generarea de cod-sursă pe mai multe straturi: interfața, aplicația, persistența datelor.

**DESCRIEREA BAZATĂ PE MODELE A INTERFEȚELOR**
Dezvoltarea unei aplicații folosind proiectarea bazată pe modele oferă o viziune abstractă asupra interacțiunii cu utilizatorul. O problemă actuală este adaptarea rapidă a acesteia în contextul în care apar modificări – minore sau majore – ale aplicației, cum ar fi: redefinirea sarcinilor, implementarea pe noi platforme de dezvoltare, evoluția modelului conceptual al aplicației, dinamica unor cerințe ale clienților etc.

O soluție este aceea de a considera ca bază o unică interfață abstractă, independentă de dispozitiv, care descrie la nivel conceptual interfața cu utilizatorul. Această descriere poate fi ulterior folosită pentru dezvoltarea concretă a interfeței pentru diverse platforme cu minimul de efort.

În susținerea acestui scop, un limbaj de descriere a interfeței cu utilizatorul (UIDL – *User-Interface Description Language*) reprezintă o abordare declarativă de a furniza mijloacele de a specifica la nivel înalt, în scopul reutilizării în diverse contexte şi modalități, o interfață. Cele mai recente cercetări în acest sens se bazează pe meta-limbajul XML (*Extensible Markup Language*), distingându-se două specificații importante: UIML şi UsiXML. Următoarele secțiuni vor descrie succint fiecare limbaj.

Limbajele declarative de specificare a interfețelor cu utilizatorul nu îşi propun rezolvarea tuturor aspectelor ce vizează utilizabilitatea. Metodologic, se foloseşte o abordare descendentă *top-down* în care interfața cu utilizatorul este divizată în extremele: *top end* şi *bottom end*. Limbajul declarativ vizează *top-end*-ul drept viziune generală a operațiilor asupra unui dispozitiv de intrare/ieşire (abstract). *Bottom-end*-ul sistemului – care specifică partea efectivă a funcționării interfeței – este în continuare codificat într-un limbaj de programare convențional. Astfel, limbajul de tip UIDL funcționează ca un „organizator" de nivel înalt pentru asamblarea unor componente majore în cadrul *puzzle*-ului interfeței.

Cele mai importante caracteristici dezirabile pe care trebuie să le prezinte un limbaj descriptiv de specificare a interfețelor sunt: aplicabilitatea pe orice platformă-țintă, aplicabilitatea în orice context de interacțiune, flexibilitatea, extensibilitatea şi simplitatea.

Conform [5], în crearea unei interfețe cu suficient grad de generalitate trebuie considerate componentele descrise în continuare.

*Sarcini şi Concepte* (*Task & Concepts*): descrie la nivel abstract – eventual folosind diverse maniere de specificare a cunoştințelor (*e.g.*, ontologii) – diverse sarcini ce trebuie îndeplinite şi aferentele concepte orientate pe domeniu. Aceste obiecte sunt considerate instanțe ale claselor ce reprezintă conceptele manipulate. Un astfel de model conceptual este format din sarcini, inclusiv proprietățile asociate, şi relațiile posibile dintre acestea. O sarcină poate fi exprimată sub formă de sub-sarcini, în mod recursiv, până la nivelul sarcinilor elementare (atomice). Sunt folosite trei modele: *context* (descrie toate entitățile care pot influența realizarea sarcinii, trebuind specificate cunoştințele privind utilizatorul, platforma şi mediului de dezvoltare), *domeniu* (prezintă modelul aplicației, uzual recurgându-se la diagrame de clase UML sau la alte mijloace de reprezentare, precum grafuri conceptuale ori specificații OWL) şi *sarcină* (modelează concepte ale domeniului sub forma de studii de caz şi reprezintă relațiile dintre acestea; fiecare sarcină are ataşat un nume, un tip şi alte atribute/meta-date).

*Interfața abstractă* (*Abstract UI* – AUI): defineşte spațiile de interacțiune prin gruparea sub-sarcinilor conform unor criterii. De asemenea, specifică o schemă de navigare printre diverse spații de interacțiune şi selectează diverse obiecte de interacțiune abstractă pentru fiecare concept. Altfel spus, reprezintă exprimarea canonică a prezentării conceptelor şi sarcinilor domeniului aplicației, independent de orice modalitate de interacțiune efectivă.

*Interfața concretă* (*Concrete UI* – CUI): concretizează o interfață definită abstract pentru un anumit context de utilizare şi independent de platformă, creându-se astfel obiecte de interacțiune concretă. La acest nivel, se defineşte structura spațială (*layout*-ul) a elementelor de interacțiune, inclusiv manierele de navigare. De asemenea, interfața concretă poate fi privită ca o abstractizare a interfeței finale (FUI) prin *reverse engineering*.

*Interfața finală* (*Final UI* – FUI): descrie interfața operațională – exprimată, tradițional, în cod-sursă – ce rulează pe o anumită platformă în urma procesării acestui cod.

Proiectarea bazată pe modele a interfeței implică metode care descriu proprietăți statice şi dinamice la diferite nivele de abstractizare.

Suplimentar, se folosesc transformări – abstractizare, reificare şi migrare – de la un model la altul [4].

Unele abordări UIDL se bazează pe modele, în timp ce altele folosesc reprezentările acestora. Anumite limbaje sunt integrate direct în model, iar altele sunt aplicate extern. Unele sunt observabile şi modificabile de către dezvoltator, pe când altele sunt inaccesibile.

## DESCRIERE SUCCINTĂ A USIXML

UsiXML (*User Interface eXtensible Markup Language*) permite crearea de aplicații interactive pentru diferite tipuri de interacțiune, modalități de utilizare şi platforme, ce pot fi descrise în manieră independentă de caracteristicile platformei hardware [2].

Destinat în special analiştilor, descrie la un nivel înalt de abstractizare elementele constituente ale interfețe: controale, containere, tehnici de interacțiune etc. Este independent de dispozitiv şi de platformă şi permite reutilizarea elementelor descrise într-o interfață anterioară.

UsiXML foloseşte multiple instrumente de transformare de la un nivel de abstractizare la altul, dintre care menționăm:

- *IdealXML* – editează modele de tip *Task & Domain* şi AUI, permițând şi specificarea relațiilor dintre modele.
- *KnowiXML* – facilitează transformarea modelului sarcinii în interfață abstractă.
- *GrafiXML* – editează modelul vizual folosit de CUI şi permite generarea automată de cod-sursă.
- *ReversiXML* – facilitează acțiunea de *reverse engineering*: crearea unei interfețe concrete sau abstracte pornind de la o pagină Web marcată în limbajul HTML.
- *TransformiXML* – oferă suport pentru transformări generale de modele, la orice nivel.

Detalii privitoare la fiecare sunt disponibile pe situl UsiXML: *http://www.usixml.org/*.

Pe baza UsiXML, o interfață poate fi specificată la şi de la niveluri multiple de abstractizare în condițiile păstrării fidele a asociațiilor dintre diversele modele. Procesul de dezvoltare poate fi demarat la orice nivel şi finalizat cu una sau mai multe interfețe pentru diverse contexte.

## DESCRIERE SUCCINTĂ A UIML

UIML (*User Interface Markup Language*) este un limbaj descriptiv simplu, oferind o reprezentare canonică a oricărei interfeţe [1]. Ca şi UsiXML, permite descrierea interfeţei în termeni declarativi şi apoi abstractizarea acesteia, pentru orice dispozitiv şi limbaj de programare. Un dezavantaj existent este acela că designerul trebuie să specifice interfaţa pentru fiecare dispozitiv şi apoi să o reprezinte în UIML. Este conceput pentru a permite compilarea eficientă din UIML în orice limbaj.

UIML poate descrie trei aspecte: elementele constituente ale interfeţei, interacţiunea (comportamentul elementelor) şi modalitatea de conectare a interfeţei la partea de procesare efectivă (*business logic*). Un aspect de interes este cel referitor la suportul acordat conceperii de interfeţe destinate persoanele cu anumite handicapuri.

Alte detalii sunt disponibile pe situl UIML aflat la adresa *http://www.uiml.org/*.

## STUDIU COMPARATIV

Pentru realizarea studiului comparativ – realizat în cadrul disciplinei *Interacțiune om-calculator* la Masterul de Ingineria Sistemelor Software, Facultatea de Informatică a Universității „A. I. Cuza” din Iaşi – am considerat

interacțiunea în cadrul unui joc electronic simplu de micro-management de resurse, similar *DopeWars* [7].

Descriem în continuare modalitățile de specificare a modelelor, în ambele abordări.

Din punct de vedere conceptual, domeniul jocului poate fi reprezentat prin clasele abstracte *GameController* (realizează derularea jocului şi facilitează salvarea şi încărcarea unui joc), *Game* (reprezintă o partidă), *Player* (modelează un jucător), *Product* (specifică un produs ce poate fi licitat), *Item* (reprezintă un artefact), *TradeStore* (modelează acțiunile de licitare), *Town* (specifică locația fizică a desfăşurării licitației).

Editarea se poate realiza cu un grad acceptabil de uşurință folosind instrumentul IdealXML [3].

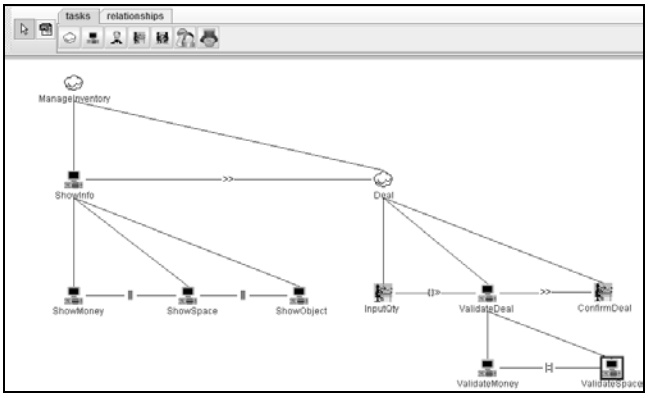


**Figura 1. Modelarea sarcinilor pe baza IdealXML**

Sarcinile ce pot fi efectuate în cadrul jocului se vor specifica în acelaşi program (Figura 1). De asemenea, am modelat şi interfața la nivel abstract, acest proces – din păcate – fiind destul de anevoios, deoarece instrumentul nu include posibilități de ajutor pentru novici, iar unele operații nu sunt intuitive.

Rezultatele obținute sunt în fapt documente XML generate automat de aplicația amintită mai sus.

Un exemplu este următorul, în care se pot observa informațiile privind sarcinile (*task*-urile) şi relațiile dintre acestea:

```
<taskmodel>
   <task id="st3task0" name="ChangeCity"
         type="abstraction">
      <task id="st3task1" name="ShowAllCitys"
            type="application" userAction="output"
            taskItem="element"/>
      <task id="st3task2" name="SelectCity"
            type="interaction"
            userAction="input" taskItem="element"/>
      <task id="st3task4" name="GoToCity"
            type="interaction"
            userAction="control" taskItem="operation"/>
   </task>
   <enabling>
      <source sourceId="st3task1"/>
      <target targetId="st3task2"/>
   </enabling>
   <enablingWithInformationPassing>
      <source sourceId="st3task2"/>
      <target targetId="st3task4"/>
   </enablingWithInformationPassing>
</taskmodel>
```

Dezvoltarea unei interfețe în UIML începe prin a se specifica maniera de prezentare a acesteia din punctul de vedere al elementelor conținute.

Pentru aceasta, s-a recurs la instrumentul LiquidApps – disponibil la *www.liquidappsworld.com* –, având la bază mediul de dezvoltare Eclipse, ceea ce asigură independența de platformă.

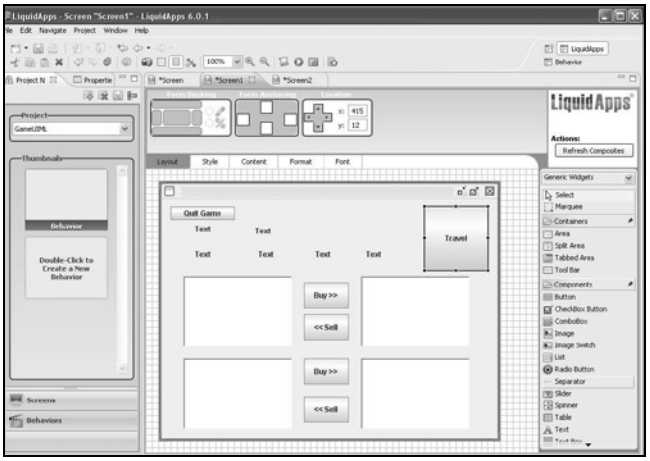


**Figura 2. Editarea elementelor de interacțiune în LiquidApps**

Este o abordare mai puțin abstractă, dar mai intuitivă pentru proiectant – a se vedea figura 2. Am constatat că aplicația este într-un stadiu avansat de dezvoltare, interacțiunea cu aceasta fiind facilă.

Pasul următor este stabilirea comportamentului acestor elemente, urmând apoi să le asociem părții de procesare (*business logic*) a jocului. Instrumentul a fost capabil să genereze codul Java al interfeței, pe baza modelului UIML creat. In figura 3 este prezentată ipostaza interfeței finale.

Alternativ, putea fi utilizat instrumentul UIML.NET pentru a realiza interfața în cod generat în limbajul C#.

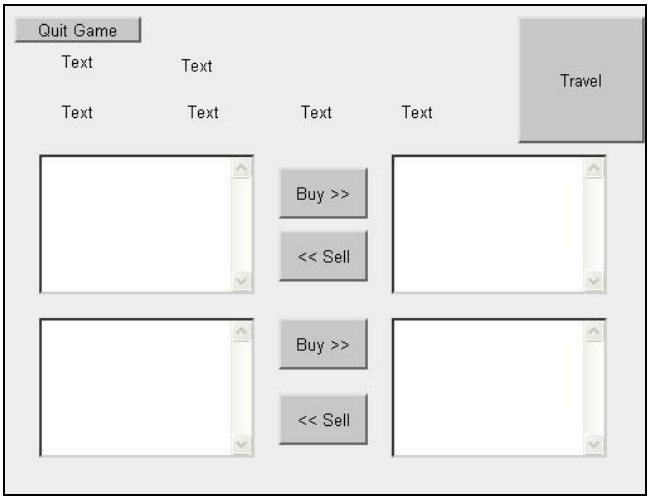


**Figura 3. Interfața finală generată automat**

Concluzionând, UsiXML pune la dispoziție o metodologie riguroasă de crearea a diverselor modele de specificare a interfețelor cu utilizator, pornind de la o abordare abstractă până la una apropiată de interfața concretă/finală. În ceea ce priveşte instrumentele de dezvoltare, acestea se află în

diverse faze de implementare, niciunul fiind suficient de matur pentru o utilizare efectivă. Structura documentelor XML generate este simplă şi clară, informațiile putând fi reutilizate şi în alte contexte.

UIML vizează în special modelarea componentelor de interfață şi generarea automată a unei interfețe concrete, urmând ca dezvoltatorul să definească – la nivel de cod – comportamentul fiecărei componente în parte. Instrumentul folosit prezintă un grad mare de utilizabilitate, fiind familiar în special programatorilor Java. Documentele XML generate sunt complexe din punct de vedere structural, informațiile stocate vizând aspecte concrete ale interfeței modelate.

## CONCLUZII

Lucrarea a descris două dintre cele mai importante limbaje descriptive de specificare la nivel abstract a interfeței – UsiXML şi UIML, ambele cu numeroase utilizări – în vederea realizării unui studiu comparativ care a vizat conceperea unei interfețe pentru un joc interactiv.

Cercetările au fost întreprinse luându-se în considerație metodologiile actuale de proiectare bazată pe modele.

Intenționăm pe viitor să considerăm cazuri mai complexe, care să implice şi interacțiuni neconvenționale precum interfețe vocale, haptice, bazate pe gesturi, tridimensionale.

## REFERINȚE


1. M. Abrams, J. Helms, *User Interface Markup Language (UIML) Specification version 3.1*, Technical Report, OASIS UIML Technical Committee, 2004.
2. Q. Limbourg *et al.*, „UsiXML: a Language Supporting Multi-Path Development of User Interfaces”, *9th IFIP Working Conference on Engineering for Human Computer-Interaction*, LNCS 3425, Springer, 2005.
3. F. Montero, V. Lopez-Jaquero, „IdealXML: An Interaction Design Tool”, în G. Calvary *et al.* (Eds.), *Computer-Aided Design of User Interfaces* V, Springer, 2007.
4. O. Pastor, „Generating User Interfaces from Conceptual Models: A Model Transformation based Approach”, în G. Calvary *et al.* (Eds.), *Computer-Aided Design of User Interfaces* V, Springer, 2007.
5. J. Vanderdonckt, „A MDA-Compliant Environment for Developing User Interfaces of Information System”, *Proceedings of 17th Conference on Advanced Information Systems Engineering*, LNCS 3520, Springer, 2005.
6. * * *, *Model-Driven Architecture*: http://www.omg.org/mda
7. * * *, *Situl Web al jocului DopeWars*: http://dopewars.sourceforge.net/