Realizarea interfețelor Web interactive folosind componente SVG

Sergiu Dumitriu
Universitatea „Al. I. Cuza” Iaşi
Strada Mesecăniş, nr A35A, ap 34, 500294 Braşov
sdumitriu@info.uaic.ro

Marta Gîrdea
Universitatea „Al. I. Cuza” Iaşi
Strada Oancea, nr 36, bl D1B, ap 24, 700350 Iaşi
marta@info.uaic.ro

Cătălin Hrițcu
Universitatea „Al. I. Cuza” Iaşi
Bulevard Prefect Gavril Tudoraş 19, bl B5, sc C, ap 3, 720264 Suceava
Catalin.Hritcu@gmail.com

REZUMAT
Această lucrare prezintă SvgUP, o platformă open source pentru dezvoltarea de interfețe Web interactive, care respectă standardele actuale şi sunt realizate folosind componente grafice predefinite.

Categorii şi descriptori ai subiectelor
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User interfaces, Software libraries*;

H.5.2 [**Information Interfaces And Presentation**]: User Interfaces – *Ergonomics, Graphical user interfaces, Standardization*;

H.5.3 [**Information Interfaces And Presentation**]: Group and Organization Interfaces – *Web-based interaction*.

Termeni generali
Design, Standardization, Human Factors.

Cuvinte-cheie
Web Interfaces, Rich Internet Applications, GUI Components, SVG, JavaScript.

1. INTRODUCERE
În evoluția sa spectaculoasă, Web-ul se transformă din ce în ce mai mult, dintr-un simplu mediu de distribuție a informației într-o platformă universală pentru dezvoltarea de aplicații. Pe măsură ce popularitatea sa creşte, Web-ul trebuie să devină mai accesibil, chiar şi pentru utilizatorii neexperimentați. Oferirea unor interfețe interactive, estetice şi uşor de utilizat devine astfel, în multe cazuri, o necesitate stringentă.

În această lucrare vom descrie *SvgUP* [19], o platformă open source pentru dezvoltarea de interfețe Web, construită pe standarde deschise existente. Platforma oferă clase extensibile pentru comunicarea client-server, inițializarea unei aplicații, încărcarea dinamică de clase, aspect uşor de modificat pentru interfață, localizare facilă. La baza acestei platforme stă *SvgWT*, o colecție de componente de interfață realizate folosind *SVG* şi *JavaScript*.

2. APLICAȚII WEB
2.1 Avantaje
Succesul de care se bucură deja aplicațiile Web este consecința avantejelor deosebite pe care acestea le oferă pentru dezvoltatori, şi mai ales pentru utilizatorii finali, acestea ajungând să reprezinte un contracandidat de temut pentru aplicațiile clasice.

Larga răspândire a browserelor care respectă standardele Web face ca fiecare aplicație Web să poată ținti o gamă practic nelimitată de dispozitive, de la stații de lucru la calculatoare personale şi dispozitive mobile. În plus, distribuirea unei astfel de aplicații este instantanee şi, indiferent de numărul de utilizatori existenți, ei vor folosi întotdeauna cea mai nouă versiune.

Utilizatorul unei aplicații Web este scutit de sarcina, uneori dificilă, a instalării aplicației. Există posibilitatea adaptării interfeței în funcție de dispozitiv, dar şi a personalizării acesteia pentru a satisface cerințele fiecăruia dintre noi [3]. Profilul utilizatorului poate fi reținut pe server şi folosit pentru a oferi o experiență consistentă între diferite sesiuni de lucru.

2.2 Interactivitate
In ciuda avantajelor evidente oferite de aplicațiile Web, tehnologiile Web existente nu sunt bine adaptate pentru aplicații ce necesită interfețe robuste şi cu un grad ridicat de interactivitate. Cauza principală a acestei probleme este chiar modelul de interacțiune care stă la baza Web-ului, un model gândit inițial nu pentru aplicații interactive ci, mai degrabă, pentru distribuția de documente (pagini Web).

Aplicațiile Web tradiționale permit utilizatorilor să urmeze legături sau să completeze formulare pe care browserul le va traduce în cereri către serverul Web. Acesta va prelucra informațiile din cerere şi va trimite drept răspuns o nouă pagină pe care browserul o va afişa. Se iroseşte astfel lățime de bandă deoarece cea mai mare parte a paginii noi era prezentă în pagina deja încărcată. Interactivitatea aplicației are astfel mult de suferit, deoarece utilizatorii sunt nevoiți să aştepte încărcarea noii pagini, ca să nu mai vorbim despre diminuarea productivității. Mai mult decât atât, atunci când trebuie efectuate operațiuni complexe şi neliniare, organizarea informației în pagini este adeseori

neintuitivă şi navigarea devine o experienţă frustrantă, mai ales pentru un utilizator neexperimentat.

O soluţie la această problemă este utilizarea *JavaScript* [11], un limbaj care permite o mult mai bună interactivitate pentru clienţii Web. Până recent browserele nu au implementat *JavaScript* într-o manieră uniformă, dar lucrurile s-au îmbunatăţit semnificativ în ultima perioadă, mai ales datorită creşterii importanţei standardelor Web. Folosind *JavaScript*, clientul poate trimite o cerere serverului pentru a primi doar datele de care are nevoie la un moment dat. Răspunsul serverului, reprezentat de cele mai multe ori în format *XML* [4], produce modificarea în mod corespunzător a interfeţei. Acest model de interacţiune reduce simţitor cantitatea de informaţie transferată, ceea ce duce la reducerea semnificativă a timpului de răspuns, diminuând în acelaşi timp şi încărcarea serverului. Acest model a fost folosit recent în aplicaţii Web de mare succes cum ar fi *GMail* [20], *Google Groups* [21] sau *Personalized Google* [22], fiind uneori numit şi *Asynchronous JavaScript and XML* (*AJAX* [1]).

2.3 Interfeţe grafice complexe

Realizarea interfeţelor grafice complexe folosind *HTML* [5] este anevoioasă, pentru rezultate satisfăcătoare folosindu-se adeseori tabele, cadre, imagini raster, datele fiind amestecate cu informaţile referitoare la prezentare. Problema a fost în mare parte rezolvată, odată cu introducerea foilor de stiluri [8] şi, pentru multe din aplicaţiile simple, acestea reprezintă o soluţie foarte bună.

Pentru grafică avansată însă, *HTML* este din păcate depăsit. Se pot folosi doar imagini raster care nu pot fi scalate satisfăcător şi sunt în general stocate în fişiere de dimensiuni mari, ducând la o creştere inutilă a timpului de aşteptare. Mai mult decât atât, orice modificare efectuată asupra unei asemenea imagini sau a datelor care au generat-o presupune reîncarcarea ei în totalitate. Pentru a atenua această problemă, în cazul hărţilor de dimensiuni considerabile, *Google Maps* [23] împarte imaginile în zone şi foloseşte *JavaScript* pentru a încarca dinamic doar acele zone de care este, la un moment dat, nevoie.

În asemenea cazuri, folosirea graficii vectoriale este o alternativă mult mai bună, pentru aceasta utilizându-se adeseori tehnologia *Flash* dezvoltată de *Macromedia* [24]. Dar deşi aceasta este o tehnologie stabilă şi cu o largă răspandire, ea are şi un dezavantaj major, fiind bazată pe un format binar proprietar, ceea ce este în totală discordanţă cu standardele Web.

Soluţia cea mai bună pe termen lung pare a fi *SVG* [6], un standard deschis pentru crearea de grafică vectorială avansată, care este disponibil pe o gamă foarte largă de dispozitive. Deoarece e bazat pe *XML*, *SVG* permite separarea formei de conţinut prin intermediul foilor de stil [8], precum şi interacţiuni complexe prin folosirea JavaScript.

Deşi SVG este o specificaţie matură, ea nu este destinată în mod special realizării de aplicaţii, şi de aceea, nu include componente standard, ci doar primitive grafice simple Folosind aceste primitive şi cod *JavaScript* se pot însă construi interfeţe oricât de complexe. Volumul de muncă necesar unei asemenea încercări reprezintă totuşi o provocare, chiar şi pentru un expert în *SVG* şi *JavaScript*, apărând astfel necesitatea realizării unor componente *SVG* avansate, care să fie reutilizabile cu uşurinţă de către dezvoltatorii de interfeţe Web. În secţiunile care urmează vom prezenta o soluţie open source care îşi propune să rezolve elegant această problemă.

3. PLATFORMA SVGUP

3.1 Introducere

La momentul actual nu există o platformă open source care să permită dezvoltarea rapidă de interfeţe Web interactive şi care să respecte standardele Consorţiului Web (*W3C*) [2]. Majoritatea soluţiilor actuale pentru design Web nu sunt gândite pentru realizarea de **aplicaţii** Web, ci de **pagini** Web, şi nu permit interacţiuni avansate între client şi server, cum ar fi de exemplu comunicarea asincronă, sau componente avansate de interfaţă.

SVG User Interface Platform (SvgUP) [19] a fost concepută special cu scopul de a crea interfeţe grafice pentru aplicaţii Web interactive. Ca tehnologii utilizate menţionăm: *SVG* [6], *JavaScript* [11], *DOM* [7], *CSS* [8], şi *RDF* [10].

3.2 Arhitectura unei aplicaţii *SvgUP*

SvgUP permite separarea codului unei aplicaţii Web în şapte părţi distincte: logică, descriere, conţinut, comportament, aspect, stil şi localizare (Figura 1).

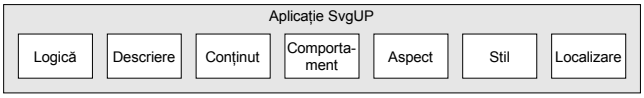

Figura 1: Arhitectura unei aplicaţii SvgUP

O asemenea arhitectură permite ca fiecare parte a aplicaţiei să fie dezvoltată separat, fără a afecta prea mult celelalte părţi. Astfel şi responsabilităţile pot fi mult mai uşor împărţite între membrii unei echipe de dezvoltatori, fiecare membru fiind în general specializat doar în câteva domenii.

Logica unei aplicaţii priveşte aspectul computaţional al acesteia, specificând cum funcţionează aceasta la nivel abstract**. Logica** unei aplicaţii Web tradiţionale este implementată în întregime pe server. Există însă multe cazuri în care are sens ca o parte a logicii aplicaţiei să fie executată de către client. O astfel de soluţie oferă un timp de răspuns mult mai bun, crescând interactivitatea aplicaţiei. Logica unei aplicaţii *SvgUP* poate să ruleze pe server şi să fie accesată de către client folosind servicii Web, local folosind *JavaScript* sau, cel mai adesea, printr-o combinaţie între procesare locală şi acces la distanţă. Este la latitudinea dezvoltatorului să aleagă locul procesării

datelor în funcție de specificul şi necesitățile aplicației proprii.

Descrierea textuală a aplicației este o componentă destul de importantă pentru aplicațiile *SvgUP*, doar prin intermediul ei putându-se asigura indexarea lor coresponzătoare de către motoarele de căutare. Aceasta deoarece aproape toate informațile sunt generate dinamic în timpul rulării folosind JavaScript. Descrierea aplicației se face folosind un fişier *RDF* folosind vocabularul *Description of a Project (DOAP)* [15], sau, pentru aplicații nesofisticate, printr-un simplu şir de caractere.

Pentru descrierea **conținutului interfețelor** se foloseşte pentru moment *XUL* (*eXtensible User interface Language* [16]), un limbajul bazat pe *XML* dezvoltat de *Mozilla Foundation*, şi folosit intensiv în aplicații ca *Mozilla Firefox* [25] şi *Mozilla ThunderBird* [26]. Urmează probabil ca ulterior să oferim şi posibilitatea folosirii altor limbaje pentru descrierea interfețe utilizator cum ar fi: *UIML* [14], *XForms* [9], *WebApplications* [13], sau *OpenLazlo* [18].

Gestiunea **comportamentului** componentelor revine platformei *SvgUP*, mai exact pachetului *SvgWT*. Pentru fiecare componentă se utilizează un fişier *ECC* (*Ecmascript Component Class* [12]) cu rol descriptiv, codul ei efectiv fiind scris în fişiere *JS* (*JavaScript*). O mare parte din efortul necesar dezvoltării unei aplicații este eliminat, deoarece în platforma *SvgUP* există o implementare implicită pentru comportamentul componentelor de interfață. Pentru crearea de componente noi este necesar un efort minim, presupunând cel mai frecvent doar adăugarea de noi funcționalități unei componente deja existente. Astfel, accentul în dezvoltarea aplicației se mută pe logica aplicației, şi nu pe comportamentul componentelor de bază.

Aspectul aplicației (*Look and Feel*) se referă la reprezentarea grafică a componentelor. Astfel, aspectul determină dacă butoanele sunt rotunjite, checkbox-urile sunt dreptunghiulare, iar ferestrele au bara de titlu în partea de sus. E necesar ca aspectul unei aplicații să poată fi schimbat cu uşurință pentru a permite particularizarea interfeței conform preferințelor utilizatorilor. Designerii pot crea noi aspecte pentru controalele standard în concordanță cu specificul aplicațiilor realizate. *Look and Feel*-ul care va fi folosit pentru o anumită instanță a unei interfațe este definit în fişierul *RDF* asociat unei aplicații sau în fişierul cu preferințele unui utilizator.

Stilul interfeței se referă cel mai adesea la tema coloristică, dar şi la dimensiunea şi aliniearea textului, fontul folosit sau grosimea liniilor. Astfel, un anumit stil descrie că butoanele sunt gri, ferestrele au textul de pe bara de titlu aliniat la stânga, iar meniurile au fontul Arial. Stilurile asigură un aspect plăcut pe dispozitive diferite şi pot uşura interacțiunea pentru persoanele cu necesități speciale. Pentru specificarea stilului sunt folosite fişiere *CSS*, referențiate în descrierea interfeței.

Localizarea aplicației permite ca textele ce apar în cadrul interfeței, poziționarea elementelor acesteia, direcția textului sau chiar elementele ce apar în interfață, să fie particularizate pentru diferite limbi. Acest lucru este realizat prin utilizarea fişierelor *DTD* şi a entităților definite în acestea.

3.3 Modelul unei aplicații *SvgUP*

Într-o aplicație Web tradițională, browser-ul are doar rolul de a afişa paginile create de server. *SvgUP* schimbă acest comportament prin transformarea clientului static într-o gazdă pentru o aplicație dinamică (Figura 2).

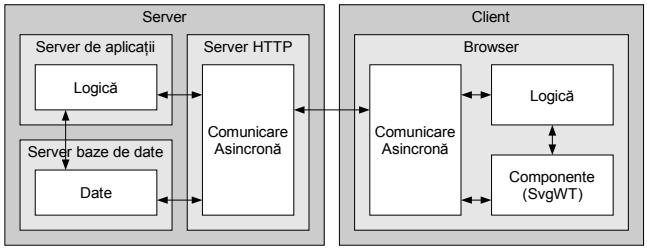

Figura 2: Modelul unei aplicații *SvgUP*

Cea mai mare parte a interacțiunii cu utilizatorul este controlată de *SvgUP* şi nu de browser. Cu ajutorul conexiunilor dinamice între client şi server, prin care sunt anunțate schimbările de stare ale acestora, dispare necesitatea reîncărcării unei pagini după efectuarea unei acțiuni. Interfața poate fi actualizată în mod dinamic pentru a reflecta schimbările cauzate de acțiunile utilizatorului.

3.4 Organizarea unei aplicații *SvgUP*

O aplicație *SvgUP* va fi încărcată prin deschiderea de către browser a unui fişier *SVG* special. Acesta va face referință la descrierea textuală a aplicației şi la clasa *SInitializer*, metoda statică *initialize* din această clasă fiind apelată la încărcarea fişierului.

Într-un fişier *RDF* utilizat în faza de încărcare (*bootstraping*) vor fi descrise modulele aplicației: clasa derivată din *SApplication* utilizată, fişierele ce descriu conținutul interfeței (de exemplu folosind *XUL*), fişierele *JavaScript* conținând comportamentul şi aspectul componentelor, fişierele *CSS* ce descriu stilul aplicației, fişierele *DTD* ce descriu localizarea aplicației. Inițial aplicația nu va încărca nici o componentă, urmând ca, pe măsură ce va avea nevoie de componente neîncărcate, acestea să fie automat transferate de pe server.

3.5 Structura *SvgUP*

Clasele platformei pot fi împărțite în următoarele categorii:

- Inițializarea platformei (*SInitializer*);
- Comunicarea client-server (*STransporter*, *SJavaScriptTransporter*, *SXmlTransporter*, *SSoapTransporter*);

- Încărcarea dinamică de clase (*SClassLoader*);
- Încărcarea aplicației şi crearea interfeței (*SApplicationLoader*, *SXulLoader*, *SUimlLoader*);
- Stilul şi aspectul aplicației (*SStyle*, *SLookAndFeel*);
- Biblioteca de componente, *SvgWT* (*SApplication*, *SMdiApplication*, *SSdiApplication*, *SWidget*, *SMenu*, *SButton*, *SListBox*, *SPanel*, *SWidgetFactory*, *SLayoutManager*, *SPersistence*).

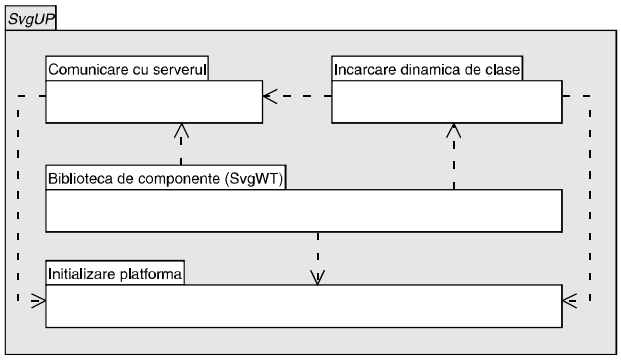

Figura 3: Arhitectura generală a unei aplicații *SvgUP*

4. CONCLUZII ŞI DIRECȚII DE VIITOR

În această lucrare am prezentat *SvgUP*, o nouă platformă *open source* pentru realizarea de interfețe interactive. Prin combinarea componentelor *JavaScript* şi folosind limbaje din familia *XML*, *SvgUP* permite dezvoltarea facilă de aplicații *Web* modulare, extensibile şi uşor de particularizat în concordanță cu cerințele utilizatorilor. SVG s-a dovedit a fi potrivit pentru interfețe Web complexe, un design sofisticat şi atractiv putând fi realizat cu uşurință prin utilizarea de primitive grafice şi combinații de efecte.

După finalizarea şi implementarea standardului *SVG* 1.2, componentele actuale vor fi modificate pentru a profita de beneficiile aduse de acest standard. Setul de componente de bază oferit de *SvgUP* urmează de asemenea a fi extins cu alte componente avansate pentru a aduce *SvgUP* mai aproape de nivelul platformelor actuale pentru crearea de aplicații desktop.

Deoarece *SvgUP* este destinată dezvoltării de aplicații Web, ar fi foarte util un mediu integrat de dezvoltare de aplicații (*IDE*). Acest mediu ar putea fi conceput în întregime ca o aplicație *SvgUP* şi ar putea conține şi un editor pentru *JavaScript*, care să faciliteze scrierea de cod orientat obiect.

5. REFERINȚE

[1] Garrett, J. J., *Ajax: A New Approach to Web Applications*. Adaptive Path. 2005: http://www.adaptivepath.com/publications/essays/archives/000385.php

[2] * * *, *World-Wide Web Consortium (W3C)*: http://www.w3.org

[3] * * *, *Web Accessibility Initiative (WAI)*: http://www.w3.org/WAI

[4] * * *, *Extensible Markup Language (XML), W3C*: http://www.w3.org/XML

[5] * * *, *HyperText Markup Language (HTML), W3C*: http://www.w3.org/MarkUp

[6] * * *, *Scalable Vector Graphics (SVG), W3C*: http://www.w3.org/Graphics/SVG

[7] * * *, *Document Object Model (DOM), W3C*: http://www.w3.org/DOM

[8] * * *, *Cascading Style Sheets (CSS), W3C*: http://www.w3.org/Style/CSS

[9] * * *, *XForms, W3C*: http://www.w3.org/MarkUp/Forms

[10] * * *, *Resource Description Framework (RDF), W3C*: http://www.w3.org/RDF

[11] * * *, *JavaScript, Standard ECMA-262: ECMAScript Language Specification, ECMA*: http://www.ecma-international.org/publications/standards/Ecma-262.htm

[12] * * *, *Standard ECMA-290, ECMAScript Components Specification, ECMA*: http://www.ecma-international.org/publications/standards/Ecma-290.htm

[13] * * *, *Web Applications, WHATWG*: http://www.whatwg.org/specs/web-apps

[14] * * *, *User Interface Markup Language, OASIS*: http://www.uiml.org

[15] * * *, *Description of a Project (DOAP), Useful Information Company*: http://usefulinc.com/doap

[16] * * *, *XUL Project, Mozilla Foundation*: http://www.mozilla.org/projects/xul

[17] * * *, *XUL Planet*: http://www.xulplanet.com

[18] * * *, *OpenLaszlo*: http://www.openlaszlo.org

[19] * * *, *SVG User Interface Platform (SvgUP)*: http://svgup.sourceforge.net

[20] * * *, *Google GMail*: http://gmail.google.com

[21] * * *, *Google Groups*: http://groups-beta.google.com

[22] * * *, *Personalized Google*: http://www.google.com/ig

[23] * * *, *Google Maps*: http://maps.google.com

[24] * * *, *Macromedia Flash*: http://www.macromedia.com/software/flash

[25] * * *, *Mozilla Firefox*: http://www.mozilla.org/products/firefox

[26] * * *, *Mozilla Thunderbird*: http://www.mozilla.org/products/thunderbird