OpenCad.org

Proiectarea unei aplicaţii CAD pentru arhitectură

Absolvent	Silviu-Georgian Aprozeanu
Profesor îndrumător	prof. dr. ing. Florica Moldoveanu

Bucureşti, 29 mai 2007

Cuprins

1 Introducere **1**

2 Obiective **2**

3 Tehnologii **3**
3.1 Eclipse Rich Client Platform 3
3.1.1 Eclipse Runtime 3
3.1.2 SWT 4

4 Arhitectură **5**

5 Implementare **6**

6 Concluzii **7**

Listă de tabele

3.1 Structura Eclipse RCP 3

Listă de figuri

Capitolul 1

Introducere

Capitolul 2

Obiective

Capitolul 3

Tehnologii

3.1 Eclipse Rich Client Platform

Proiectul Eclipse îşi are începuturile alături de compania IBM de care s-a separat în 2003; astăzi este printre cele mai cunoscute proiecte opensource, fiind recunoscut ca un foarte popular mediu de dezvoltare de aplicaţii Java şi nu numai.

Binecunoscutul mediu de dezvoltare Eclipse este dezvoltat pe baza Eclipse Rich Client Platform. Aceasta reprezintă suita minimă de plugin-uri necesare pentru dezvoltarea unei aplicaţii "rich client" [1]. Deşi platforma este destinată dezvoltării de aplicaţii tip IDE, prin înlăturarea anumitor componente ale platformei se pot dezvolta orice tip de aplicaţii desktop.

Structura de bază a acestei platforme poate fi rezumată astfel[2]:

Tabela 3.1: Structura Eclipse RCP

Componentă	**Descriere**
Eclipse Runtime	Suport pentru plugin-uri, puncte de extensie şi extensii. Construită peste framework-ul OSGi
SWT	Proiectat să ofere acces eficient şi portabil la facilităţile de interfaţă utilizator ale sistemului de operare pe care este implementat
JFace	Un framework de interfaţă utilizator, construit pest SWT, pentru tratarea multor sarcini de programare de interfeţe
Workbench	Construit peste toate componentele anterioare, aduce un mediu de lucru ultra scalabil, cu interfaţă publică ce suportă mai multe ferestre pentru administrarea vizualizărilor, editoarelor, perspectivelor, acţiunilor, preferinţelor şi multe altele.

3.1.1 Eclipse Runtime

Această componentă a Eclipse RCP este cea mai abstractă şi în acelaşi timp cea mai puternică componentă a sa.

Structura sa de bază poate fi împărţită în[1]:

- org.eclipse.core.contenttype - Suport pentru definirea şi administrarea tipurilor de fişiere
- org.eclipse.core.expressions - Un limbaj generic de expresii XML folosit în definirea punctelor de extensie.

[1] eng. *client bogat* – O aplicaţie desktop care beneficiază de un set de elemente de interfaţă evoluate ("bogate"), care oferă o putere mare de abstractizare şi în acelaşi timp control al funcţionalităţii programatorului, păstrînd o experienţă de utilizare plăcută pentru utilizator. În general orice aplicaţie desktop care foloseşte elemente de interfaţă predefinite pînă la un anumit punct poate fi considerată ca un rich client.

- org.eclipse.core.filesystem - Un API generic peste sistemul de fişiere
- org.eclipse.core.jobs - Infrastructură pentru programare paralelă în Eclipse
- org.eclipse.core.resources - Administrarea proiectelor, resurselor şi fişierelor
- org.eclipse.core.runtime - Istoric a fost baza platformei, acum a fost înlocuit de proiectul Equinox

3.1.2 SWT

SWT[2] este una din cele mai populare tehnologii dezvoltate de Fundaţia Eclipse. Această tehnologie foloseşte la crearea de interfeţe utilizator în mediul de dezvoltare Java, oferind programatorului capacitatea de a construi interfeţe utilizator portabile între diverse sisteme de operare. În prezent suportă toate sistemele de operare uzuale şi diverse arhitecturi maşină (e.g. x86, amd64).

Structura sa este similară cu cea a altor toolkit-uri similare, ca AWT şi SWING. Elementele fundamentale sunt Shell-ul şi Controlul. Un shell este abstracţia conceptului de fereastră şi reprezintă în general un container pentru controale. Controalele sunt elementele interfeţei utilizator. Butoanele, etichetele, arborii şi tabelele sunt toate controale şi utilizatorii sunt obişnuite cu ele din alte programe ale desktopului[3]. În principiu orice widget poate fi simplu sau compus (i.e. descendent al clasei Composite) şi întreaga lor ierharhie este controlată de relaţiile dintre clasele SWT.

Interacţiunile dintre controale şi utilizator cît şi controlul tranziţiei stărilor controalelor se face cu ajutorul Evenimentelor. Fiecare modificare de stare a unui control (cum ar fi apăsarea pe un buton) este semnalată de SWT către aplicaţie prin declanşarea unui eveniment. Orice Listener (ascultător înregistrat pentru un anumit eveniment) va primi o notificare (i.e. o metodă declarată în interfaţa de Listener va fi apelată) ce va conţine evenimentul ce a avut loc. Orice clasă poate deveni listener prin implementarea unor interfeţe specifice, SWT oferind astfel o mare flexibilitate în cadrul aplicaţiilor care îl folosesc.

Ca şi în AWT sau SWING, poziţia controalelor este stabilită cu ajutorul Layout-urilor. Un layout reprezintă un set de reguli de poziţionare pentru un control compus ce va decide pentru toate controalele incluse cum vor fi ele poziţionate pe ecran. Unele layout-uri asociază anumite date fiecărui control pentru a reţine poziţia în care va fi desenat. Poziţionarea manuală a tuturor controalelor este o problemă complexă şi dacă acel algoritm nu este scris eficient, el poate afecta performaneţele întregii aplicaţii. De aceea, SWT oferă nişte seturi predefinite de layout-uri ce serversc cele mai comune cazuri întîlnite de programatori.

[2]abrev. *Standard Widget Toolkit*

Capitolul 4

Arhitectură

Capitolul 5

Implementare

Capitolul 6

Concluzii

Bibliografie

[1] Eclipse rcp core component
http://www.eclipse.org/eclipse/platform-core/.

[2] Martin Oberhuber, Nick Boldt, Daniel Spiewak, Thomas Bonk, et al. Rich client platform frequently asked questions
http://wiki.eclipse.org/index.php/RCP_FAQ.

[3] Joe Winchester and Steve Northover. Swt - a native widget toolkit for java
http://java.sys-con.com/read/37463.htm.