동적 GUI 생성을 위한 코드 생성 시스템

안중진 최종명○ 조용윤 유재우
숭실대 컴퓨터학과 목포대 컴퓨터공학과○
{jjahn, yycho}@ss.ssu.ac.kr, choijm@dreamwiz.com○, cwyoo@ssu.ac.kr

Design and Implementation of Dynamic GUI Code Generation System

Jung-jin ahn Jong-Myung Choi○ Yong-Yoon Cho Chae-Woo Yoo
Dept. Computer, Soongsil Univ. Dept. Computer-Engineering , Mokpo Univ. ○

요 약

본 논문은 실행시에 가변적인 형태의 사용자 인터페이스를 제공하기 위해서 GUI 코드 생성기 시스템을 구현한다. 이 시스템에서는 XML 문서를 이용하여 GUI구조를 기술한다. SAX를 이용하여 파싱한다. 파싱 결과물인 트리를 비지터 패턴을 이용하여 각 태그에 해당하는 코드를 생성하고 이 코드를 자이썬 인터프린터를 이용하여 사용자 인터페이스를 생성한다.

1. 서 론

컴퓨팅 환경이 발전하면서 GUI(Graphic User Interface) 사용이 많아지고 있으며, 이에 따라 GUI 개발은 점차 다양한 기기, 컴퓨팅 환경, 사용자의 요구에 많은 영향을 받는다. 특히 pervasive computing 환경에서는 기존의 환경보다 더욱 다양한 인터페이스를 요구하기 때문에 개발자들은 인터페이스 개발에 더욱 더 많은 비용과 시간을 소비하게 된다. 그러므로 개발자들은 생산성 향상을 위해 비주얼 스튜디오(Visual studio), 넷빈(netBean)등과 같이 쉽게 배우고, 사용할 수 있는 GUI 디자인 도구를 선호한다[1]. 그러나 이러한 GUI 디자인 도구를 사용할 때에는 다음과 같은 문제점이 있다. 새로운 인터페이스가 요구될 때 프로그래머는 위의 도구들을 사용하여 디자인하고 사용자 이벤트 처리 부분을 프로그래밍 언어를 통하여 기술해야 한다. 지속적으로 새로운 인터페이스가 요구되는 경우에 이를 분석, 설계하는데 많은 추가적인 시간과 비용을 소모하게 된다.

HTML을 확장한 XML[2]은 사용자가 원하는 형태의 태그와 속성을 정의하는 방법을 제공한다. XML 표준이 제정된 이후에 XML 스키마(schema)[3], 프로토콜 개발[4] 및 이를 이용하는 응용프로그래밍의 개발[5] 등의 연구가 진행됨과 동시에 XML을 이용한 사용자 인터페이스를 개발하려는 연구도 진행되고 있다[6,7,8,9]. XML을 이용해서 인터페이스를 개발하려는 가장 중요한 이유는 XML이 플랫폼과 프로그래밍 언어에 독립적이고, 배우기 쉽기 때문이다. XUL[6]과 UIML[7,8,9]은 인터페이스 개발을 위한 XML XML 기반 인터페이스 언어의 가장 대표적인 예이다. 그러나 현재까지 개발된 XML 기반 인터페이스 언어는 정적인 인터페이스 형태를 구성하는 것은 효과적이지만 실행시간에 가변적인 형태의 인터페이스를 구성하기는 어렵다. 본 논문에서는 이러한 문제점을 개선하기 위해 XML로 부터 가변적인 형태의 인터페이스를 위한 방법을 제안한다. 제안된 방법은 푸쉬 파서(Push Parser)인 SAX 파서[10]를 사용하여 문서 구조를 분석하고, 비지터 패턴(Visitor pattern)[11]을 사용하여 문서구조에 알맞은 코드를 생성한다. 이렇게 생성된 코드는 자이썬(Jython) 인터프리터를 사용하여 동적인 GUI를 생성한다.

2. 관련연구

2.1 XUL(XML User Interface Language)

모질라(Mozilla) 프로젝트에서 개발된 사용자 인터페이스를 위한 XML 파생 언어로 모질라 브라우저(Mozilla browser)를 더욱 쉽고 빠르게 개발하기 위해 설계되었다. XUL은 다른 XML의 모든 장점을 지니고 있어 XHTML이나 MathML, SVG와 같은 다른 XML 파생 언어를 XUL에 삽입할 수 있고 XUL로 나타내는 언어는 적은 노력으로 다른 언어로 번역될 수 있다. 사용자 인터페이스는 내용(content), 스킨(skin), 장소(locale)로 구성되

어 있으며 내용은 윈도우와 다른 사용자 인터페이스 요소(script)들의 선언이 포함되어 있다. 스킨은 세부적 윈도우의 모습을 나타내는 스타일 시트, 이미지, 그리고 다른 특수 주제 파일들을 포함하고 있다. XUL은 정적인 GUI를 기술하기 위한 언어 이며 모질라 웹 브라우저를 위해서 개발된 언어이기 때문에 일반 응용 프로그램의 사용자 인터페이스 개발에는 적합하지 않다.

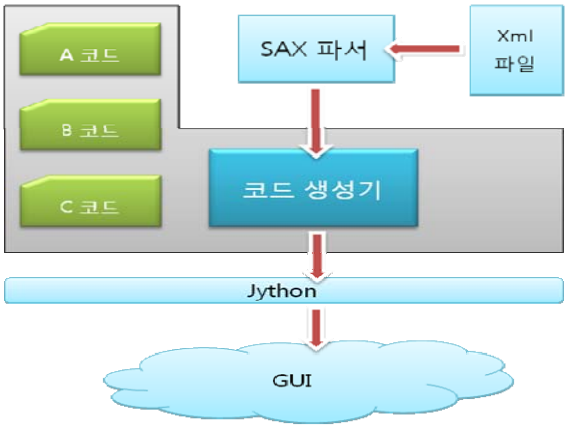
2.2 UIML

UIML은 주변 장치에 무관하게 사용자 인터페이스를 기술하기 위해 개발되었다. UIML은 타겟 디바이스에서 사용되는 언어(예: HTML, WML, VoiceML, C++, Java 등)에 자동적으로 매핑된다. UIML은 head, interface, peers, template이라는 4개의 원소로 구성되어 있다. head 원소는 문서에 대한 메타데이타(예: 저자, 버전 등)를 제공한다. interface 원소는 사용자 인터페이스를 구성하는 인터페이스의 구조(structure), 내용(content), 스타일(style), 행동(behavior)을 기술한다. structure 원소는 다양한 플랫폼을 위한 인터페이스와 인터페이스들 간의 구성을 기술한다. style 원소는 인터페이스와 관련된 다양한 속성 값들을 기술한다. content 원소는 사용자에게 전달할 정보(예: 텍스트, 음성, 이미지)를 가지고 있다. behavior 원소는 사용자가 인터페이스 위짓(widget)과 상호 작용할 때 어떤 액션이 수행되어야 할 지를 기술한다. peer 원소는 UIML의 각 속성과 이벤트이름을 UI 툴킷과 내부 로직에 매핑시키는 역할을 한다. template 원소는 UIML 문서의 일부분을 재활용하기 위해서 사용된다. 그러나 UIML은 내부 로직과 사용자 인터페이스를 연결하는 부분이 부족하고, 데이타 플로우 기능 및 UIML을 편리하게 개발할 수 있는 그래픽 편집기를 지원하지 않는다.

3. GUI 코드 생성기

3.1 시스템 개요

아래 그림 1은 본 논문에서 제안하는 동적 GUI 생성을 위한 코드 생성기 시스템 구조도 이다.

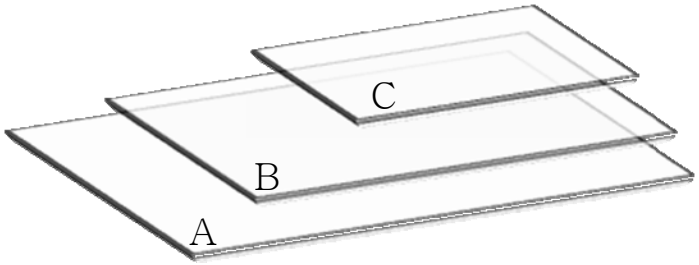


[그림 1] GUI 생성을 위한 코드 생성기 시스템 구조도

GUI 생성을 위한 코드 생성기 시스템은 크게 세 부분으로 나누어진다. 첫 번째는 XML 파일을 입력 받아 N-진 트리를 생성하는 SAX 파서 부분이고 두 번째는 코드 생성기 부분이다. Sax 파서로 부터 입력 받은 트리를 비지터 패턴을 사용하여 탐색하면서 각 XML 태그에 맞는 자이썬 코드를 생성한다. 세 번째는 코드 생성기에서 생성한 자이썬 코드를 인터프리터를 이용하여 GUI를 생성하는 부분이다.

3.2 GUI 코드 생성

애플리케이션에서 GUI 아래 그림 2 와 같이 중첩구조를 이용하여 표현한다. 이러한 특성은 XML 태그의 중첩구조로 표현 가능하다. 예를 들면 자바에서 frame panel, 버튼 순으로 중첩되어 있는 사용자 인터페이스는 frame, panel, 버튼 순으로 XML 문서에 기술한다. 이와 같은 방법으로 XML을 이용하여 GUI 표현이 가능하다.



[그림 2 사용자 인터페이스 구조]

SAX 파싱을 통해 그림 2 의 GUI구조를 표현한 XML 문서는 A, B 와 B, C 각각이 부모자식 관계를 갖는 트리를 생성할 수 있다. 이 트리를 비지터 패턴을 이용해 순회 하면서 각 노드에 해당하는 코드를 생성한다. 이때 루트 노드를 제외 한 다른 노드들은 부모 노드에 포함되는 코드를 추가적으로 넣어주어 XML 문서에서 기술한 사용자 인터페이스의 중첩구조를 코드에 반영한다.

4. 실행결과

그림 2는 프린터 제어를 위한 사용자 인터페이스를 만드는 XML 파일로 각 태그와 태그의 속성은 사용자 인터페이스 객체와 객체의 속성을 정의한다.

[표 1] 사용자 인터페이스 XML 파일

```
<?xml version="1.0" encoding="UTF-8"?>

<frame id="MyPrinterDlg" name="MyPrinter"
heigth="615" width="466" visible="1">
  <spilt id="printerSplit" orientation="vertical"
dividerLocation="350">
```

```
      <leftComponent id = "left">
      <scroll id="scroll1">
<textPane id="text"
editable="false"></textPane>
          </scroll>
       </leftComponent>

       <rightComponent id = "right">
          <panel id="panel">
             <flow align="left"></flow>
             <button   id="btn1"   name="Get
Status">
                <ImageIcon
src="img/info.jpg"></ImageIcon>
             </button>
             <button id="btn2" name="print">
                <ImageIcon
src="img/print.jpg"></ImageIcon>
             </button>
          </panel>
       </rightComponent>
   </spilt>
</frame>
```

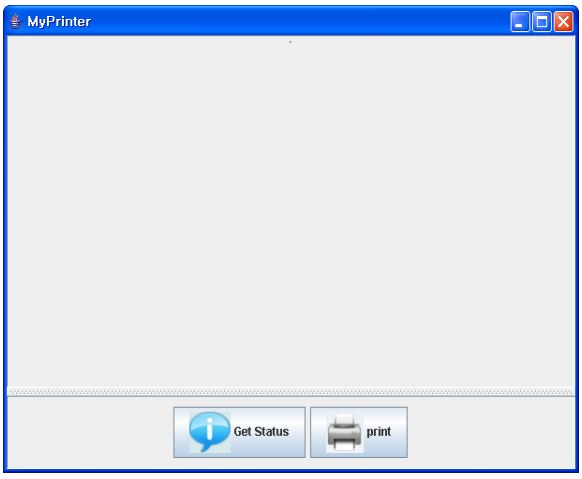
[그림 3] 사용자 인터페이스 XML 파일

그림 4는 그림 3의 XML 파일을 입력 받아 만든 자이썬 코드이다.

[표 2] 생성된 자이썬 코드

```
from java.awt import *
from java.awt.event import *
from javax.swing import *
from java.lang import System

class MyPrinterDlg (JFrame) :
def __init__(self, title="MyPrinter"):
JFrame.__init__(self, title)
self.size = (615, 466)
self.visible = 1
printerSplit = JSplitPane()
printerSplit.setOrientation(0)

printerSplit.setDividerLocation(350)

self.contentPane.add(printerSplit)
left = JPanel()

printerSplit.setLeftComponent(left)
scroll1 = JScrollPane()
left.add(scroll1)
text = JTextPane()
text.setEditable(0)
scroll1.add(text)
right = JPanel()

printerSplit.setRightComponent(right)
panel = JPanel()
right.add(panel)
panel.layout=FlowLayout(FlowLayout.LEFT)
btn1 = JButton('Get Status')
panel.add(btn1)

icon=ImageIcon('img/info.jpg')
btn1.setIcon(icon)
btn2 = JButton('print')
panel.add(btn2)
icon=ImageIcon('img/print.jpg')
btn2.setIcon(icon)
if __name__ == '__main__':
MyPrinterDlg().validate()
else:
print 'not __main__'
MyPrinterDlg().validate()
```

아래 그림 4는 위의 표 3의 코드를 실행 시킨 결과로 생성된 사용자 인터페이스 이다.



[그림 4] 생성된 사용자 인터페이스

5. 결론 및 향후과제

본 논문은 GUI구조를 기술한 XML 문서를 이용하여 실행 시 가변적인 형태의 사용자 인터페이스를 생성하는 시스템을 제안하였다. 그리고 이를 위해 XML 문서를 SAX를 이용하여 파싱한다. 파싱결과 생성된 트리는 비

지터 패턴을 이용하여 자이썬 인터프리터를 통하여 실행 시 가변적인 형태의 사용자 인터페이스를 생성하는 문제를 해결하였다. 하지만 본 논문에서 제안하는 시스템은 이벤트 처리 부분의 코드는 생성하지 않기 때문에 개발자가 직접 작성해야 하는 단점이 있다. 이는 편리성을 저하시키고 이는 곧 생산성 저하로 이어진다. 이는 향후 연구로 보완해야 할 사항이다. 그리고 현재 SAX를 사용하여 XML 문서를 파싱하기 때문에 사용할 수 있는 메모리가 제한적인 임베이디드 기기에서는 위의 시스템을 적용하기가 어렵기 때문에 향후 XML Pull Parser를 이용하여 임베이디드 기기에서 사용할 수 있도록 개선할 것이다.

6. 참고문헌

[1] Ben Shneiderman, Designing the User Interface Strategies for Effective Human-Computer Interaction, Addison Wesley, 1998.
[2] XML, http://www.w3c.org/xml
[3] XML Schema, http://www.w3c.org/XML/Schema
[4] Simple Object Access Protocol, http://www.w3c.org/TR/SOAP
[5] G. Badros, "JavaML: A markup language for Java Source code," in Computer Networks, Vol. 33, pp. 159-177, June, 2000
[6] Mozilla.org, "Introduction to a XUL (XML-based User Interface Language) Document," available at http://www.mozilla.org/xpfe/xptoolkit/xulintro.html
[7] Abrams, M., Phanouriou, C., Batongbacal, A. L. , Williams, S. M. , Shuster, J.E., "UIML: An Appliance-Independent XML User Interface Language, "WWWS Conf., May, 1999, available at http://www.harmonia.com/resources/papers
[8] Sumanth Lingam, "UIML for Voice Interfaces," UIML Europe 2001 Conf., Mar., 2001, available at http://www.harmonia.com/resources/papers/
[9] Marc Abrams, "Device-Independent Authoring with UIML," W3C Workshop on Web Device Independent Authoring, Oct, 2000, available at http://www.harmonia.com/resources/papers/

[10] SAX http://www.saxproject.org/
[11] JOHN VLISSIDES , Erich Gamma , Ralph Johnson , Richard Helm, *Design Patterns: Elements of Reusable Object-Oriented Software - Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1995