GroovyMarkup 확장을 이용한 SWT Builder의 구현 및 성능 평가

이건우*°, 고동진*, 이동주*, 우균*, 김원영**, 최완**
*부산대학교 컴퓨터공학과
**한국전자통신연구원 디지털 홈 연구단
leoric99°@pusan.ac.kr, exsider03@yahoo.co.kr, {mrlee, woogyun}@pusan.ac.kr,
{wykim, wchoi}@etri.re.kr

Implementation and Performance Valuation of SWT Builder Using GroovyMarkup Extension

Gunwoo Lee*°, Dongjin Go*, Dongju Lee*, Gyun Woo*, Won-Young Kim**, Wan Choi**
*Dept. of Computer Engineering, Pusan National University
**Digital Home Research Division, Electronics and Telecommunications research Institute

요 약

Java 플랫폼 기반의 스크립트 언어인 Groovy는 GroovyMarkup을 이용하여 컴포넌트 및 콘테이너, 객체가 중첩된 구조로 이루어져 있는 GUI 프로그램을 간결하고 쉽게 작성할 수 있다. 본 논문에서는 GroovyMarkup을 확장해 고성능의 GUI 프로그램을 구현할 수 있는 SWT Builder를 구현하였다. 본 논문에서 구현한 SWT Builder의 성능 및 기능 평가를 위해 기존에 구현된 SWT Builder, Swing Builder와 비교하여 실험하였다. 테스트 한 결과, 구현한 SWT Builder는 GUI 구성 시간에 있어 기존 SWT Builder보다 1.4배 더 빠른 속도를 가진다. 또한 기능 부분에서는 Factory 클래스 소스 코드의 자동 생성, 클래스 기능 문서 자동 생성 등 으로 프로그래머에게 더 많은 편의성을 제공한다.

1. 서 론

Java 플랫폼 기반의 객체 지향 스크립트 언어인 Groovy는 Java의 특징과 스크립트 언어의 특징을 동시에 가지고 있어 보다 쉬운 프로그래밍 환경을 제공한다[1]. 특히 Groovy의 특징 중 GroovyMarkup은 XML문서와 같이 각각의 객체가 중첩된 트리구조를 다루는 응용프로그램을 쉽게 생성하거나 제어할 수 있어서 GUI(Graphic User Interface) 프로그램의 구성에 적합하다.

GUI 프로그램은 GUI 컴포넌트를 구성하는 부분과 컴포넌트 속성(Component Attribute), 이벤트 처리(Event Handling) 부분으로 구분할 수 있다. 이 중 컴포넌트와 컨테이너의 트리 구조를 가지는 GUI의 골격 부분은 GroovyMakup을 이용하여 간결하게 표현이 가능하며 MarkupBuilder를 통해 수행 가능한 Groovy GUI 프로그램으로 구체화된다. 현재 GroovyMarkup의 GUI Builder는 Swing을 지원하는 Swing Builder, SWT를 지원하는 SWT Builder(org)[1]가 공개 프로젝트로 개발되고 있다[2].

본 논문에서는 Groovy로 간결하며 좋은 성능을 내는 GUI 프로그램을 구현하기 위해서 기존에 설계된 SWT Builder[3]를 기반으로 SWT Builder를 구현한다. 또한 구현한 SWT Builder의 효율성을 입증하기 위해 SWT Builder로 작성한 Groovy SWT 프로그램과 SWT Builder(org)로 작성한 Groovy SWT 프로그램, Swing Builder로 작성한 Groovy Swing 프로그램, Java SWT 프로그램, Swing 프로그램의 성능을 비교하여 평가한다.

본 논문의 구성은 다음과 같다. 2절에서는 SWT Builder에 대해서 살펴보고, 3절에서는 SWT Builder의 설계와 구현 방법에 대해서 살펴본다. 4절에서는 SWT Builder와 SWT Builder(org), Swing Builder를 통해 작성된 Groovy 프로그램과 Java로 작성된 SWT, Swing 프로그램의 성능을 평가한다. 끝으로 5절에서는 결론을 내리며 향후 SWT Builder의 발전 방향에 대해서 살펴본다.

2. SWT Builder

Groovy에서 GroovyMarkup을 이용하여 GUI 프로그램을 기술할 경우, GUI 프로그램의 3가지 구성요소인 컴포넌트, 컴포넌트의 속성, 이벤트 처리를 명백히 분리하여 표현할 수 있다. GUI 컴포넌트의 생성 즉, 골격의 구성은 GroovyMarkup을 이용하고 컴포넌트의 속성은 Map 타입의 객체를 이용한다. 그리고 컴포넌트의 이벤트 처

1) 본 논문에서 구현한 SWT Builder와 구분하기 위해서 Groovy 공식 사이트에서 구현하고 있는 SWT Builder를 SWT Builder(org)로 표기한다.

리는 클로저(closure)를 이용한다.

Java 플랫폼의 표준 GUI 라이브러리인 Swing을 Groovy에서 사용하려면 Groovy에서 제공하는 SwingBuilder와 GroovyMarkup을 이용하면 된다. 하지만 스크립트 언어의 특성상 Java 프로그램보다 느린 Groovy 프로그램과 플랫폼에 독립적인 Swing 라이브러리로 인하여 SwingBuilder로 작성한 GUI 프로그램은 비교적 낮은 성능을 낸다[4]. 따라서 본 논문에서는 Java 플랫폼에서 표준은 아니지만 Swing 라이브러리에 비해 좋은 성능을 내는 SWT 라이브러리를 GroovyMarkup으로 작성할 수 있도록 SWT Builder를 구현한다.

3. SWT Builder의 설계 및 구현

SWT Builder는 GroovyMarkup 형식의 Groovy 소스코드를 입력으로 받아 수행 가능한 SWT GUI 프로그램으로 구체화 시킨다. SWT Builder의 전체 구성은 그림 1과 같다.

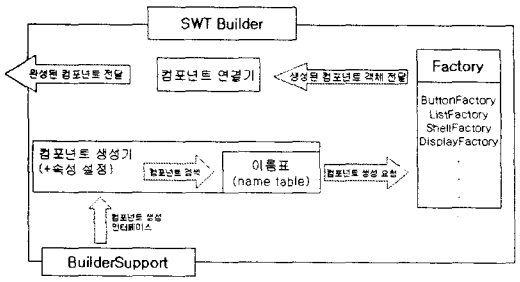

그림 1 SWT Builder의 전체 구성

BuilderSupport는 GroovyMarkup의 구현을 위한 기본적인 인터페이스를 제공한다. SWT Builder는 BuilderSupport 클래스를 상속받으며 노드 생성과 관계, 속성 설정에 해당하는 메소드를 재정의 함으로써 새로운 Builder를 구현한다.

SWT Builder는 BuilderSupport의 컴포넌트 생성 인터페이스를 통하여 컴포넌트 생성기에게 컴포넌트 객체의 생성을 요청한다. 컴포넌트 생성기는 컴포넌트 태그(tag) 이름과 컴포넌트 Factory 쌍으로 구성된 이름표(name table)에서 컴포넌트 Factory를 검색하여 해당 Factory에게 컴포넌트 객체의 생성을 요청한다. 이때 인터페이스로 전달된 컴포넌트 태그의 속성 값도 설정된다.

Factory에 의해 생성된 컴포넌트 객체는 컴포넌트 연결기에게 전달되고 컴포넌트 연결기는 생성된 각각의 컴포넌트 객체의 관계를 설정한다. 실제 이 부분은 SWT 라이브러리의 setParent 메소드를 이용하여 구현된다. 모든 설정이 다 끝나면 컴포넌트 연결기는 최종적으로 완성된 GUI 프로그램을 반환한다.

SWT Builder의 Factory 모듈을 자동으로 작성하기 위해 Factory 코드생성기를 작성하였다. Factory 코드생성기는 컴파일된 SWT 라이브러리를 입력으로 받아 모든 SWT 컴포넌트에 대한 Factory 모듈의 소스코드를 자동으로 생성한다.

Factory 생성기에 의해 작성된 SWT Builder는 Factory 모듈을 찾기 위해 리플렉션(Reflection)[2][5]을 이용하는 SWT Builder(org)와는 달리 Factory 코드 생성기에 의해 컴포넌트 클래스 별로 이미 작성된 메소드 호출을 사용하게끔 하여, 리플렉션의 사용으로 인한 오버헤드[6]를 줄일 수 있다. 또한 개발 측면에서는 모든 SWT 컴포넌트를 지원하기 위해 일일이 Factory 모듈을 수작업으로 작성하는 것보다 효율적이다.

Factory 코드 생성기는 다음 그림 2와 같이 5개의 모듈로 구성된다.

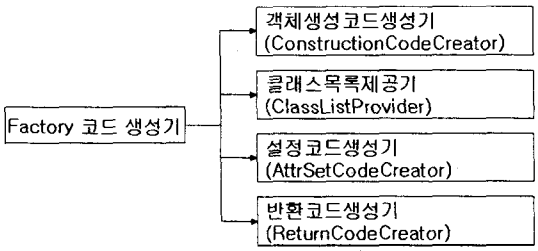

그림 2 Factory 코드 생성기의 구조

Factory 코드 생성기의 각 모듈의 기능은 다음과 같다. 객체생성코드생성기(ConstructionCodeCreator)는 각 컴포넌트의 객체를 생성하는 코드를 만든다. 클래스목록제공기(ClassListProvider)는 Factory 클래스 소스 코드를 생성할 때 Factory가 필요한 컴포넌트의 클래스 목록을 알려주며, 설정코드생성기(AttrSetCodeCreator)는 생성한 컴포넌트의 속성을 설정하는 메소드의 소스 코드를 만들어 준다. 마지막으로 반환코드생성기(ReturnCodeCreator)는 생성한 컴포넌트 객체를 반환해주는 소스 코드를 만들어준다.

SWT Builder는 컴포넌트의 속성을 설정하는 방법을 SWT Builder(org)와 다르게 구현하였다. SWT Builder(org)는 생성자 매개변수(constructor parameter)를 속성(attribute)의 표현과 유사하게 처리하는 반면, SWT

2) 리플렉션은 Java 프로그램에서 동적으로 클래스의 인스턴스를 생성하고 메소드를 호출할 수 있도록 해주는 Java API이다.

Builder는 맵(map) 기반의 속성 설정 구문과 생성자 매개변수를 모두 이용할 수 있도록 한다. 예를 들어 SWT의 Button 컴포넌트에는 Button(Composite parent, int style); 과 같은 생성자가 있는데, SWT Builder(org)에서는 Button을 생성할 때 button(text:"hello world", style:"PUSH"); 과 같은 형태의 문법에 따라 속성과 생성자 매개변수를 넘기지만, SWT Builder에서는 button(text:"hello world", SWT.PUSH); 과 같이 마지막의 객체를 생성자 매개변수로 처리하게 하여 메소드 사용시 속성과 생성자 매개변수의 구분이 용이하도록 구현하였다.

부가적으로 GUI 프로그램 작성시 프로그래머의 편의성을 위해 리포트 기능을 추가하였다. 리포트 기능은 Factory 코드 생성기를 이용하여 Factory 소스 코드 생성시 각 컴포넌트 클래스에 대해 사용 가능한 기능을 문서화하여 자동으로 PDF파일을 만들어 준다.

4. SWT Builder의 성능 평가

본 논문에서 구현한 SWT Builder의 성능 평가를 위해서 Swing과 SWT 기반의 Java GUI 프로그램과 SWT Builder(org), Swing Builder로 작성한 Groovy GUI 프로그램의 전체 컴포넌트 구성 시간을 비교하였다.

성능 평가를 위한 실험 환경으로는 펜티엄 4 3.06GHz, 512M 사양을 갖춘 PC에 JDK1.5, Groovy1.0.6을 이용하였다.

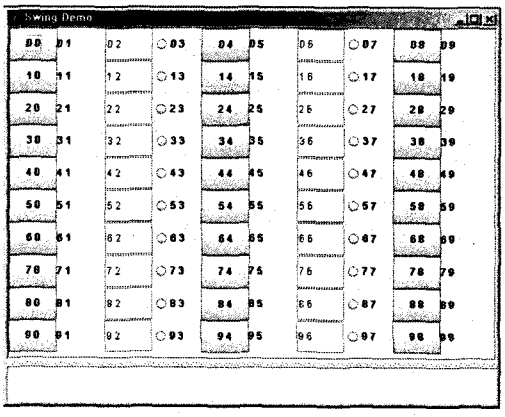

그림 4 Swing을 이용한 GUI 컴포넌트 구성

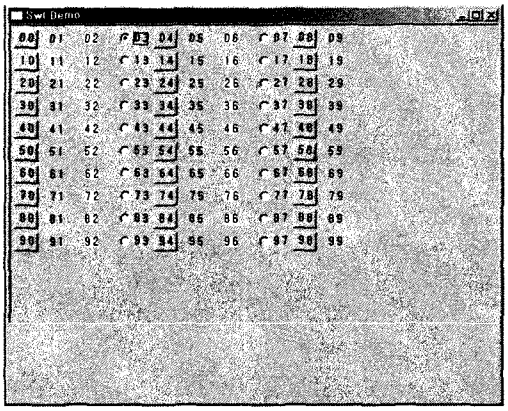

그림 5 SWT를 이용한 컴포넌트 구성

Java의 SWT, Swing와 Groovy의 Swing Builder와 SWT Builder(org), SWT Builder의 GUI 컴포넌트 전체 구성 시간을 측정하였다. 측정을 위해 가장 기본이 되는 GUI 컴포넌트인 버튼, 텍스트, 텍스트 에디터, 라디오 버튼을 그림 4, 5와 같이 100, 200, 400, 800, 1600, 3200개씩 만들어 각각을 100번씩 테스트하여 GUI 컴포넌트의 전체 구성 시간을 측정하여 각 평균 시간을 구하였으며 실험결과 수행 시간은 표 1과 같다.

(단위 : millisecond)

라이브러리 / 컴포넌트 수	Java		Groovy		
	Swing	SWT	Swing Builder	SWT Builder(org)	SWT Builder
100	263	362	2,539	1,391	1,113
200	328	462	2,701	1,586	1,309
400	423	575	2,943	1,859	1,561
800	678	779	3,425	2,453	2,111
1,600	1,032	1,200	4,332	3,748	3,303
3,200	1,438	2,248	5,837	6,440	5,741

표 1 GUI 컴포넌트 전체 구성 시간 측정 결과

Groovy로 작성한 GUI 프로그램의 컴포넌트 구성 시간은 Java의 Swing으로 작성한 GUI 프로그램 보다 5~10배 느렸고, SWT로 구현한 GUI 프로그램의 경우는 2~3배 정도 느린 속도를 보였다. SWT Builder를 사용하여 GUI 프로그램을 작성하였을 경우 전체 컴포넌트 구성 시간은 Swing Builder로 작성한 GUI 프로그램에 비해서는 평균 2배 정도 빠른 속도를 보였고, SWT Builder(org)로 작성한 GUI 프로그램에 비해서는 1.4배 정도 빠른 속도를 보였다. 실험시 컴포넌트의 수를 일정 크기의 개수를 늘리는 것 보다 배로 개수를 늘렸을 때 성능의 결과를 명확하게 비교할 수 있었다.

본 논문에서 구현한 SWT Builder가 기존 SWT Builder(org)보다 빠른 이유는 앞서 구현부에서 언급한 바와 같이 Builder 실행시 리플렉션을 사용하지 않고 일반 메소드 호출을 사용하였고 이벤트 처리를 위해 별도의 클래스를 만들어 사용할 수 있게 하였기 때문이다.

5. 결론 및 향후 연구

본 논문은 Groovy에서 간결한 표현력으로 고성능의 GUI 프로그램을 작성하기 위한 개발환경으로 SWT Builder를 구현하였다. SWT Builder의 성능을 평가하기 위해 SWT Builder(org), Swing Builder, Java로 작성된 Swing, SWT GUI 프로그램과 비교하였다.

실험 결과, SWT Builder로 작성한 GUI 프로그램은 Swing Builder로 작성한 GUI 프로그램보다 2배 정도, SWT Builder(org)로 구현한 GUI 프로그램보다 1.4배 정도 더 빠른 수행 성능을 가졌다. 따라서 Groovy에서 간결한 표현으로 고성능의 GUI 프로그램을 작성할 수 있음을 SWT Builder를 통해 보였다. 또한 부가적으로 사용이 가능한 컴포넌트 목록을 제공함으로써 SWT Builder를 이용하는 프로그래머에게 편의성을 제공하였다.

본 논문에서 구현한 SWT Builder는 몇 가지 개선해야 할 점이 있다. 아직 org.eclipse.swt.SWT, org.eclipse.swt.dnd.DND 등 일부 컴포넌트를 지원하지 않은 상태이며, JFace 라이브러리의 Factory 클래스를 생성할 수 있는 Factory 코드 생성기는 구현하지 않은 상태이다.

향후 SWT Builder 구현에 사용한 Factory 코드 생성기를 다른 GUI 라이브러리를 지원 가능하게 확장한다면 다양한 Builder를 자동으로 작성하는데 응용 가능할 것이다.

참고문헌

[1] Rod Cope and James Strachan, The Groovy Programming Language: Let's Get Groovy, Presentation at 2004 JavaOne, Conference, 2004. (http://www.codehaus.org/~jstrachan/GroovyJavaOne-2004.pdf)

[2] Groovy Homepage, http://groovy.codehaus.org

[3] 이동주, 지정훈, 장한일, 우균, 김원영, 최완, "GroovyMarkup 확장을 이용한 SWT Builder의 설계", 한국정보과학회 2005 추계학술대회, 32권. 2호 pp. 976~978, 2005년 11 월

[4] Barry Feigenbaum, SWT, Swing or AWT: Which is right for you?, IBM developerWorks, 2006 : http://www-128.ibm.com/developerworks/java/library/os-swingswt

[5] Ken Arnold, James Gosling, David Holmes, "The Java™ Programming Language, Fourth Edition", Addison Wesley, pp. 397~446, 2005

[6] Dennis Sosnoski, "Java programming dynamics, Part 2 : Introducing reflection", 2003 : http://www.ibm.com/developerworks/library/j-dyn0603/#N102F6

[7] SWT : The Standard Widget Toolkit, http://www.eclipse.org/swt/

[8] Tutorial and Example of Eclipse and SWT : http://www.cs.umanitoba.ca/~eclipse/

[9] David Flanagan and Brett McLaughlin, "Java 5.0 Tiger: A Developer's Notebook", Morgan Kaufmann, 2006

[10] Kenneth Barclay and John Savage, "Groovy Programming: An Introduction for Java Developers", O'Reilly Media, 2004

[11] Jackwind Li Guojie, "Professional Java Native Interfaces with SWT/JFace (Programmer to Programmer)", Wrox, 2005