제약 사항을 적용한 DML(Diagram Markup Language)

윤태희°, 최종명, 유재우
숭실대학교 컴퓨터학부
oinee2k°@ss.ssu.ac.kr

DML(Diagram Markup Language) applying Constraints

Tae-Hee Yoon°, Jong-Myung Choi, Chae-Woo Yoo
School. of Computing, Soongsil University

요 약

다이어그램을 이용하는 시스템에서 다이어그램 컴포넌트는 객체의 특성을 갖기 때문에 속성(attribute)과 행위(behavior)로 구성된다. 속성은 다이어그램 컴포넌트의 위치, 크기, 또는 색깔 등의 형태(shape)정보를 나타내며, 행위는 최소크기 제한, 최대크기 제한 또는 크기변경 등의 형태정보에 대한 제약사항(constraints)과 다이어그램 컴포넌트가 응용프로그램에서 가지는 의미(semantics)를 말한다. 따라서 형태 정보만을 XML로 나타내는 방법에 대한 기존의 연구로는 다이어그램 컴포넌트가 가져할 객체 특성을 나타내기에는 적합하지 않다. 이러한 문제점을 해결하고자 본 논문에서는 XML을 이용해 다이어그램 컴포넌트가 객체의 특성을 가질 수 있도록 제약 사항과 의미로 구성된 행위 원소를 추출하여 정의하고, 행위를 표현할 수 있는 방법과 행위 원소간의 영향범위를 지정할 수 있는 방법 그리고 이를 관리할 수 있는 DML 시스템을 소개한다.

1. 서 론

다이어그램은 문서작성, 설계 등의 분야에서 널리 사용되지만 서로 다른 방법으로 구현됨으로써 응용프로그램간의 다이어그램 컴포넌트 공유가 불가능하다. 이러한 문제점을 해결하고자 XML[1]을 사용하여 그래픽 정보를 나타내는 방법들이 연구되고 있다[2, 8].

다이어그램 응용프로그램에서 다이어그램 컴포넌트는 정적인 형태 정보뿐만 아니라 동적인 행위를 갖고 있다. 즉, 화면에서의 위치, 크기 또는 색깔 등의 정적 정보뿐만 아니라 사용자의 속성 변경에 대한 제약 사항이나, 다이어그램 컴포넌트 자체의 의미와 같은 동적 정보를 갖기도 한다. 따라서, 기존의 XML을 이용한 그래픽 표현은 정적인 형태 정보만을 표현하기 때문에 동적인 객체의 특성을 갖는 다이어그램 컴포넌트를 표현하기에는 부족하며, 행위 속성을 정의할 수 있는 새로운 방법을 필요로 한다.

이 논문에서는 XML을 이용하여 다이어그램 컴포넌트가 객체의 특성을 표현할 수 있도록 행위에 대한 원소를 정의한 DML 문서를 제시한다. 또한 이 원소들이 서로 영향받는 관계와 이를 관리할 수 있는 방법에 대해 소개한다.

본 논문은 2장에서는 관련 연구를 소개하고, 3장에서는 DML 설계와 다이어그램 제약 사항 표현 및 흐름에 대해 소개한다. 또, 4장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 Scalable Vector Graphics(SVG)

SVG[2]는 2차원 벡터 그래픽 및 혼합된 벡터/래스터 그래픽을 표현하기 위해 W3C에서 웹 표준으로 정한 XML 응용프로그램이다. SVG는 경로정보로 표현되는 벡터 그래픽, 이미지 그리고 텍스트 세 가지 그래픽 객체가 있다. 각 그래픽 객체는 그룹으로 묶어 재사용이 가능하고 객체를 공유할 수도 있다. 하지만 SVG는 그래픽 정보만을 저장할 뿐이고 다이어그램이 가져할 규칙 및 의미정보 등을 포함하지는 않는다.

2.2 DiaGen

DiaGen[3]은 다이어그램 편집기 생성기로서 다이어그램 컴포넌트 정의, 리듀싱 규칙, 문법 및 의미를 기술하여 자동으로 free hand 또는 syntax directed 다이어그램 편집기를 생성한다. 하지만 *DiaGen*은 *Hypergraph* 문법을 사용함으로 복잡하고 확장성이 낮은 단점이 있다.

2.3 속성 문법(Attribute Grammar)

속성 문법[4]은 의미 규칙을 추가한 문맥 자유 문법으로 속성은 다음 두 가지 타입으로 구분한다. 첫째는 상속 속성(inherited attribute)으로 파스 트리에서 부모노드에서 자식노드로 속성 값이 전달되는 것을 말하며 둘째는 합성 속성(synthesized attribute)으로 자식노드에서 부모노드로 속성 값이 전달되는 것을 말한다.

다이어그램에서 속성문법을 사용하는 것은 2차원 공간에서 다이어그램 컴포넌트를 트리 형태로 정의한 수 있고 다이어그램 컴포넌트의 속성 값 변경이 해당 컴포넌트의 외부 혹은 내부 컴포넌트에 의해 발생할 수 있기 때문이다.

3 DML 설계 및 시스템 구현

3.1 DML 설계

다이어그램 컴포넌트는 사용자 이벤트에 의해 속성 값이 바뀔 수 있고, 바뀐 속성 값에 맞는 행위를 하기 때문에 속성과 행위를 정의하는 원소로 구성된다. 속성은 형태 정보 즉 화면상으로 다이어그램 컴포넌트가 나타내는 모습을 나타내고, 행위는 형태 정보의 변화에 의해 발생하는 제약 사항과 자체의 역할을 하는 의미 정보를 나타낸다.

그림 1은 DML에서 제시하는 DTD(Document Type Definition)이다. 형태정보, 제약 사항, 의미 정보 등을 나타내기 위한 원소(element)를 정의한다.

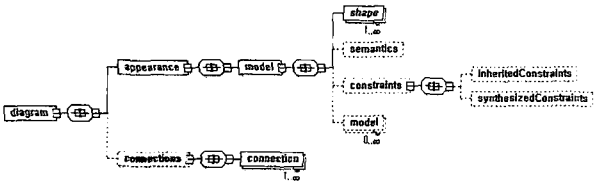

그림 1. DML의 DTD

diagram은 최상위 원소로 다이어그램 컴포넌트에 대한 형태 정보를 정의하는 appearance 원소와 다른 컴포넌트간의 연결 정보를 정의하는 connections 원소로 구성된다. 형태 정보를 갖는 appearance 원소는 여러 개의 다이어그램 컴포넌트에 대한 model 원소들로 구성되며, 하나의 model 원소는 실제 그림정보를 갖는 shape 원소와 그림에 대한 제약 사항을 갖는 constraints 원소, 다이어그램 컴포넌트의 의미 정보를 갖는 semantics 원소로 구성된다.

3.2 다이어그램 컴포넌트 생성

그림 3은 DML 그래픽 에디터를 통해 생성된 UML[7] 클래스 다이어그램 컴포넌트이다. 사용자는 우선 전체 크기의 사각형을 만든 후 클래스 이름, 속성 리스트, 메소드 리스트를 위한 각각의 사각형을 만들어 하나 또는 여러 문자열을 입력할 수 있는 컴포넌트를 해당 사각형 내부에 삽입한다.

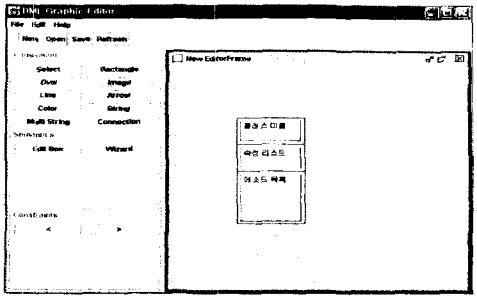

그림 3. 다이어그램 컴포넌트 생성

3.3 다이어그램 컴포넌트 관리

다이어그램 컴포넌트는 2차원 공간에서 여러 컴포넌트들의 집합으로 구성되어 있으며 공간에 대한 집합적 관계성을 부여함으로써 계층적 구조로 표현이 가능하다. 이렇게 생성된 트리 자료구조에서 제약 사항들을 추출하여 의존 그래프(dependency graph)를 생성함으로써 각 노드간의 영향 범위와 순서를 찾아내어 관리할 수 있다.

그림 2는 그림 3에 의해 생성된 자료 구조와 제약 사항에 대한 위치 및 폭과 너비 속성 값의 의존 그래프로서 속성 문법의 상속 속성과 합성 속성을 사용한다.

각 노드는 속성(x, y, width, height 등)과 행위({...})를 갖는 객체이며 전체 노드가 서로 유기적으로 묶여 하나의 다이어그램 컴포넌트를 구성한다. 상위 노드의 변경이 하위 노드의 속성 정보에 영향을 주는 것이 상속 속성으로 표현되었으며, 하위 노드의 속성 변화에 대해 상위 노드로 영향을 주는 것이 합성 속성으로 표현되어 있다.

속성 문법의 타입에 따라 두 개의 독립된 DAG(Directed Acyclic Graph)로 볼 수 있고 각 DAG의 위상 정렬(topological sort)에 의해 각 노드의 영향 범위에 따른 정렬이 가능해져 어떤 한 노드의 변형에 의한 다른 노드의 변형에 대한 정의가 가능해진다. 또한 정렬된 노드 정보를 이용하여 다이어그램 컴포넌트를 list, switch-case 또는 table-driven 방식으로 관리할

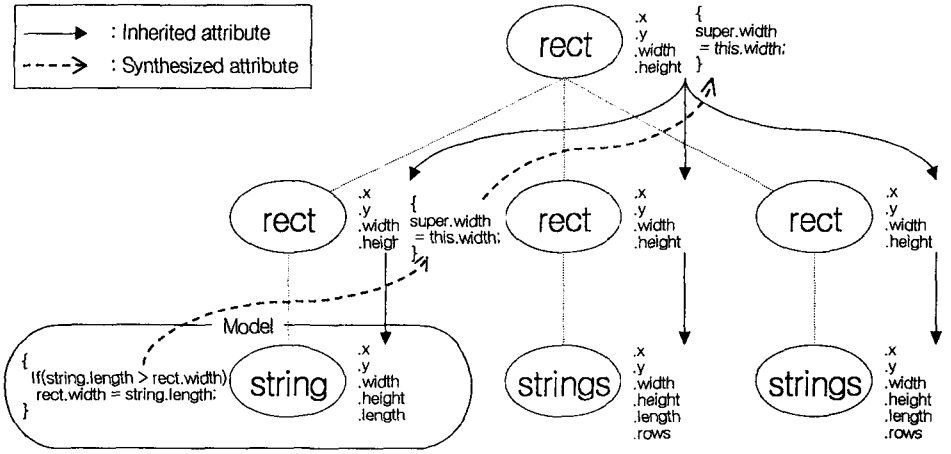

그림 2. constraints dependency graph

수 있다.

3.4 DML 시스템 구현

그림 4는 그림 3에 의해 생성된 형태와 제약사항을 갖는 다이어그램 컴포넌트의 DML문서이다. 이렇게 생성된 DML문서는 DOM[6] 파서나 XSLT[5]를 사용하는 변환기를 통해 원하는 언어로 변환된다.

```
<diagram type="UMLClass" width="100" height="200">
  <appearance>
  <model>
    <shape type "rect" id "a" x="0" y="0"
                          width="100" height="200"/>
    <model>
      <shape type="rect" id="b" x="0" y="0"
                          width="100" height="50"/>
      <model>
        <shape type="string" id="c"/>
        <constraints>
          <synthesizedconstraints>
            if(c.length > b.width) b.width = c.length;
          </synthesizedconstraints>
        </constraints>
      </model>
      <constraints>
        <inheritedconstraints>
          c.x = b.x;
          c.y = b.y;
          c.width = b.width;
          c.height = b.height;
        </inheritedconstraints>
      </constraints>
      ...
<connections>
  <connection type="point" value="center center"/>
</connections>
</diagram>
```

그림 4. 생성된 DML 문서

그림 5는 그림 4의 DML 문서를 자바 언어로 변환한 소스의 일부이다.

```
package dml.ss.ssu.ac.kr;
public class UMLClass extends dml.editor.Model {
...
Rect b = new Rect();
...
public UMLClass(DMLCanvas canvas){ ...
b.setX(0); b.setY(0); b.setWidth(100); b.setHeight(50); ...
public void doInheritedConstraints(int node){
        switch(node){
        case 1 :  { ... }
                  doInheritedConstraints(2); break;
        case 2 :  {
                  c.x = b.x;
                  c.y = b.y;
                  c.width = b.width;
                  c.height = b.height;
                  }
                  doInheritedConstraints(3);
                  break;
}
```

그림 5. 생성된 DML 문서

그래픽 객체 생성과 의존 그래프 관리에 의한 제약 사항 처리에 대한 소스를 볼 수 있다. 여기서는 switch-case 문으로 정렬된 그래프 노드를 관리한다.

그림 6은 변환기를 통해 생성된 다이어그램 컴포넌트를 다이어그램 에디터에서 실행시킨 결과이다.

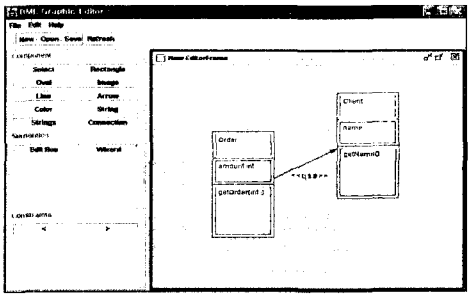

그림 6. 다이어그램 컴포넌트 실행 결과

4. 결론 및 향후 연구방향

본 논문에서는 XML을 이용하여 다이어그램 컴포넌트를 표현하기 위한 방법으로 DML을 제안했다. 즉, 다이어그램 컴포넌트는 객체의 특성을 갖기 때문에 형태 정보에 대한 속성 원소와 제약 사항 및 의미 정보에 대한 행위 원소로 구분하여 설명하였으며 행위 원소간의 영향 범위와 이를 관리할 수 있는 방법에 대해 소개했다.

앞으로, DiaGen과 같이 각 다이어그램 사이의 문법을 정의하여 사용자에게 올바른 다이어그램을 그릴 수 있도록 도와주는 문법 지향 에디터에 대해 연구할 계획이다.

5. 참고 문헌

[1] Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/2000/REC-xml-20001006.

[2] Scalable Vector Graphics, http://www.w3.org/Graphics/SVG/.

[3] M. Minas, O. Koth. "Generating Diagram Editors with DiaGen". in *Proc. of the Int'l Workshop with Industrial Relevance*, pp. 433-440, Sep., 1999.

[4] Jukka Paakki, "Attribute Grammar Paradigms -- A High-level Methodology in Language Implementation", in *ACM Computing Surveys*, Vol. 27, No. 2, pp. 197-255, 1995.

[5] XSL Transformations, http://www.w3.org/TR/xslt.

[6] Document Object Model, http://www.w3.org/DOM/.

[7] Booch, G., Rumbaugh, J. and Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

[8] Vector Markup Language, http://www.w3.org/TR/NOTE-VML, 1998.