XCEL: 객체지향 스프레드시트
(XCEL: Object-oriented Spreadsheet)

최 종 명 † 유 재 우 ††
(Jong-Myung Choi) (Chae-Woo Yoo)

요 약 스프레드시트는 사용하기 쉽기 때문에 가장 널리 사용되는 프로그래밍 도구이지만, 아직까지 스프레드시트 응용프로그램 개발에 객체지향 기술을 적용하려는 시도는 상대적으로 적었다. 일반적으로 스프레드시트 응용프로그램은 내부에 많은 오류를 포함하고 있으며, 재사용성이 낮고, 유지 보수가 어려운 단점을 가지고 있다. 이러한 문제를 해결하기 위해서 본 논문에서는 XCEL이라는 객체지향 스프레드시트를 소개한다. XCEL은 스프레드시트 응용프로그램을 체계적으로 개발하기 위한 데이타 모델링 방법과 스프레드시트에서 객체지향 프로그래밍을 표현하기 위한 방법을 지원한다. XCEL은 데이타 모델링에서 XML과 플로우차트를 이용해서 클래스를 정의하고, 스프레드시트에서는 연속된 셀들의 영역을 이용해서 객체들을 표현한다. XCEL을 이용하면 스프레드시트 응용프로그램 개발에 객체지향 기술들을 그대로 적용할 수 있는 장점이 있다.

키워드 : 객체지향, 스프레드시트, XML

Abstract Spreadsheet is one of the most widely used programming tool because of its ease of use, however there have been few researches on applying object-oriented techniques in developing spreadsheet applications. Generally, spreadsheet applications contain a lot of errors, and spreadsheet has some drawbacks such as low reusability and hard maintenance. In order to solve these problems, we introduce an object-oriented spreadsheet system, named XCEL. It enables users to develop applications using systematic techniques with data modelling method, and allows users to apply object-oriented technologies to spreadsheet programming. The data modelling method allows to define classes using XML and flowchart, and the spreadsheet represents an object with contiguous cells. XCEL has advantage that users can apply object-oriented technology to spreadsheet programming.

Key words : object-oriented, spreadsheet, end-user computing, XML

1. 서 론

Visicalc[1]로부터 시작된 스프레드시트는 가장 널리 사용되는 응용프로그램들 중의 하나로서, 가장 성공적인 시각 프로그래밍 도구로 인식되고 있다. 그리드(grid) 형태의 GUI, 직접 입력하는(fill-in) 스타일의 편리한 사용자 입력 방식, 상호 대화적인 실행 등의 특성은 스프레드시트를 사용하기 쉽게 해준다. 스프레드시트는 사용하기 편리하고, 배우기 쉬우며, 기능이 뛰어나기 때문에 일반 사용자들도 회계, 통계, 분석 등의 분야에서 널리 사용하고 있다. 또한 스프레드시트의 이러한 장점 때문에 스프레드시트를 이미지 프로세싱[2], GUI 생성[3] 등의 분야에 적용하기 위한 연구도 수행되고 있다.

스프레드시트는 널리 사용되고 있음에도 불구하고, 대형 소프트웨어 개발에는 사용하기 어려운 치명적인 단점들을 가지고 있다. 첫 번째 문제점으로 스프레드시트로 개발된 응용프로그램은 많은 오류들을 가지고 있다는 것이다. Panko[4]의 연구에 의하면 모든 스프레드시트 응용프로그램의 20-40%가 오류를 가지고 있다. 이러한 문제는 응용프로그램이 커지는 경우에 더욱 심각해진다. Coopers[5]는 150행 이상의 스프레드시트 응용프로그램 중에서 90%가 오류를 포함하고 있다고 밝히고 있다. 따라서 대형 스프레드시트 응용프로그램의 경우에 거의 100% 오류를 포함하고 있다고 볼 수 있다. 프로그램의 오류는 단순히 계산상의 오류로 끝나는 것이 아니라 심각한 경제적인 손실을 가져온다. 실제로 영국에서 1992년 세금 계산을 위해 사용된 스프레드시트

· 본 연구는 숭실대학교 교내 연구비 지원으로 이루어졌음
† 학생회원 : 숭실대학교 컴퓨터학과
jmchoi@comp.ssu.ac.kr
†† 종신회원 : 숭실대학교 컴퓨터학과 교수
cwyoo@comp.ssu.ac.kr
논문접수 : 2002년 8월 21일
심사완료 : 2003년 6월 24일

프로그램에서 131개의 오류를 발견했으며, 오류당 196,603 파운드의 계산상 착오가 있었다[6].

스프레드시트의 오류는 기계적 오류(mechanical error), 논리 오류(logic error), 생략 오류(omission error)로 분류할 수 있다[7]. 기계적 오류는 숫자나 공식을 입력할 때 발생할 수 있는 입력 오류이고, 논리 오류는 개발자가 잘못된 알고리즘을 사용할 때 발생하는 오류이다. 생략 오류는 개발자가 중요한 사항을 고려하지 못해서 발생할 수 있다. 논리 오류와 생략 오류는 주로 문제 분석과 해결 방법에 대한 신중한 고려가 부족하기 때문에 발생하며, 기계적 오류보다 심각한 문제를 발생시킨다. 이러한 문제들 때문에 스프레드시트에 체계적인 개발 방법이 필요하다는 의견이 여러 문헌에서 언급되고 있지만[8], 아직까지 이러한 분야에 관한 연구가 상대적으로 적었다.

두 번째 문제로는 스프레드시트는 소프트웨어 재사용을 위한 기능이 미약하다는 것이다. 일반 프로그래밍 언어에서는 사용자가 라이브러리를 구축하고, 재사용할 수 있는 방법을 제공한다. 특히 객체지향 언어인 경우에는 라이브러리 구축은 물론 상속 및 컴포지션을 통한 재사용 방법도 지원한다. 그러나 스프래드시트는 제한적으로 라이브러리를 구축할 수 있는 방법을 제공하고, 주로 사용하는 재사용 방법은 copy-paste 방식이다. 이러한 copy-paste 방식은 제한적이고, 에러가 발생하기 쉽기 때문에 대형 프로젝트에는 적합하지 않다.

세 번째 문제로 스프레드시트는 유지 보수가 어렵다는 것이다. 스프레드시트는 데이타와 공식이 혼합되어 사용되고 있으며, 캡슐화와 추상화가 지원되지 않는다[9]. 사용자는 고수준의 자료형이나 함수를 정의할 수 없기 때문에 작성된 스프레드시트 응용프로그램의 전체적인 구조를 파악하기 위해서는 모든 셀들을 방문하면서 저수준의 공식과 데이타를 파악해야 한다. 또한 캡슐화가 지원되지 않기 때문에 유지 보수시에 불필요한 데이타들이 모두 화면에 나타남으로써 프로그램을 이해하기 어렵게 하는 요인이 되고 있다. 이러한 문제를 부분적으로 해결하기 위해서 데이타들의 의존 관계를 그림으로 표현하는 방법들이 연구되어왔다[10]. 그러나 이러한 데이타들의 의존 관계는 개별 데이타들의 의존성을 표현하기 때문에 매우 복잡하고, 의미를 파악하기 어렵다. 이밖에도 스프레드시트 응용프로그램에는 문서화가 적다는 것도 유지 보수의 어려움이 되고 있다.

스프레드시트의 문제점들을 해결하기 위해서 본 논문에서는 스프레드시트에서 객체지향 프로그래밍을 지원할 수 있는 방법과 클래스를 XML과 플로우차트를 이용해서 지원할 수 있는 데이타 모델링 방법을 소개한다. 또한 제안된 방법에 따라 스프레드시트에서 객체지향 프로그래밍을 수행할 수 있는 XCEL이라는 객체지향 스프레드시트를 소개한다. XCEL은 클래스 디자이너와 스프레드시트로 구성된다. 클래스 디자이너는 일반 사용자도 쉽게 배울 수 있도록 XML을 이용해서 클래스를 정의하고, 메소드는 공식과 플로우차트를 이용해서 기술할 수 있는 방법을 제공한다. 스프레드시트는 클래스 디자이너에서 정의된 클래스들을 이용해서 동적으로 객체들을 생성하고, 메소드를 호출할 수 있는 방법을 제공한다. XCEL은 클래스와 객체를 지원함으로써 캡슐화와 추상화를 제공하고 높은 재사용, 쉬운 유지 보수 등의 장점을 갖는다. 또 다른 장점은 스프레드시트 응용프로그램 개발에서 객체지향 분석과 설계가 가능해진다는 것이다. 즉, UML과 같은 객체지향 분석, 설계 도구들을 사용할 수 있다. 이러한 객체지향 분석과 설계는 그 동안 스프래드시트에서 부족했던 체계적인 개발 방법을 제공할 것이다.

본 논문은 2장에서 XML과 플로우차트를 이용해서 클래스를 정의하는 데이타 모델링 방법을 소개하고, 3장에서는 클래스와 객체를 스프레드시트에 적용하기 위한 방법을 소개한다. 4장에서는 본 논문에서 구현한 XCEL 시스템의 구조를 소개하고, 5장에서는 기존의 연구들과의 관계성을 밝힌다. 마지막으로는 6장에서는 결론과 향후 연구 과제를 밝힌다.

2. 데이타 모델링

스프레드시트 모델링에 구조적 방법론을 적용하기 위한 연구가 있었지만, 객체지향 방법론을 적용하기 위한 연구는 상대적으로 적었다. 객체지향 방법론은 높은 재사용성과 생산성, 유지 보수의 용이함 등과 같은 많은 장점을 가지고 있다. 본 논문에서 제안하는 객체지향 스프레드시트의 기본적인 목적은 객체지향 프로그래밍의 장점과 스프레드시트의 장점을 결합하는 것이다.

XCEL은 일반 사용자가 클래스를 정의할 수 있도록 클래스 디자이너를 제공한다. 스프레드시트에서 사용되는 데이타들은 일반적으로 엔티티 데이타들이다. 엔티티 객체들은 데이타를 표현하는 속성들이 중심이 되고, 메소드들은 복잡한 기능을 포함하지 않기 때문에 간단한 계산식 혹은 간단한 함수로 표현할 수 있는 특징이 있다. 엔티티 객체들은 연관성있는 데이타들을 구조화시키고, 데이타들의 관계를 계산식으로 표현함으로써 쉽게 정의할 수 있다.

XCEL의 클래스 디자이너는 데이타들을 구조화시키고, 클래스를 정의하기 위해서 XML을 사용한다. XML의 태그명을 사용자가 임의로 정의할 수 있기 때문에 사용자가 손쉽게 데이타들을 표현할 수 있다. 클래스 디자이너는 사용자가 데이타를 모델링하기 위해서 XML 예

제 문서를 작성하도록 한다. 사용자는 XML 예제 문서를 작성하면서 문제 영역의 객체들을 파악하고, 데이타들을 구조화시키게 된다. 예를 들어, 학생 성적을 관리하기 위한 스프레드시트 응용프로그램을 객체지향 방법으로 작성한다고 가정해보자. 첫 번째 할일은 학생을 위한 XML 예제 문서를 작성하는 것이다. 사용자는 태그 이름을 임의로 결정할 수 있으며, 태그 내용도 문서 내용에 맞게 임의로 넣을 수 있다. 사용자는 XML 예제 문서를 작성하는 동안에 객체들을 파악하고, 데이타들을 구조화시키게 된다. 또한 사용자는 실제로 문서를 작성하기 때문에 문제와 해결책에 대한 구체적인 정보를 갖게 된다. 이러한 구체적인 XML 문서 내용은 사용자로 하여금 문제를 보다 쉽게 생각할 수 있도록 하고, 편안하게 느끼도록 한다. 만약 계산이 필요한 경우에는 계산을 위한 수식을 태그 내용으로 추가한다. 그림 1에서 학생의 평균 성적(avg)은 수학과 영어 점수를 합한 것을 2로 나눈 것이다. 태그 내용중에 '=' 문자가 포함된 경우에 클래스 디자이너는 수식이 포함된 것으로 간주한다.

```
<Student>
  <name>Gil D. Hong</name>
  <math>85</math>
  <eng>90</eng>
  <avg>
    avg = (math + eng) / 2
  </avg>
</Student>
```

그림 1 Student 예제 문서

만약 수식이 복잡한 경우에는 플로우차트를 사용해서 수식을 지정하는 것도 가능하다. XCEL의 클래스 디자이너는 수식 혹은 함수를 정의하기 위해서 플로우차트를 제공한다. 플로우차트는 프로그램 소스를 생성할 뿐만 아니라, 값을 제공하는 경우에 동적으로 실행할 수 있다. 플로우차트를 통해서 생성된 코드는 CDATA 섹션으로 XML 예제 문서에 포함되게 된다.

작성된 XML 예제 문서는 클래스 디자이너에 의해서 또 다른 XML 형태인 TCML(Tiny Class Markup Language) 문서로 변환된다. TCML 문서도 예제 문서와 유사하지만, 각 태그는 type 속성을 가지고 있다. type 속성은 문서의 내용을 보고 유추하게 된다. type 속성 값이 "abc"인 경우에는 문자열을 의미하고, "99"와 "99.99"는 각각 정수와 실수를 의미한다. 만약 type 속성이 생략된 경우에는 타입 이름이 태그 이름과 동일하다고 가정한다. readonly 속성은 값을 외부에서 변경할 수 없는 경우에 사용된다. 사용자는 생성된 TCML 문서가 사용자의 의도에 맞게 생성되었는지 체크한다. 만약 변경이 필요한 경우에는 TCML 문서를 사용자의 의도에 맞게 변경할 수 있다. 그림 2는 Student 예제 문서로부터 생성된 TCML 문서의 예이다.

```
<Student>
  <name type="abc"/>
  <math type="99"/>
  <eng type="99"/>
  <avg type="99.99" readonly="yes">
    avg = (math + eng) / 2
  </avg>
</Student>
```

그림 2 Student의 TCML 문서

상속은 객체지향 프로그래밍에서 코드를 재사용하기 위한 방법이다. TCML은 kindof 속성을 이용해서 상속을 지원한다. 그림 3은 Human 클래스가 name 속성을 갖고 있고, Student는 Human으로부터 상속받는다는 것을 보여준다. 사용자가 작성하는 예제 문서에서 type, kindof, number 속성을 기술할 수 있으며, 이러한 속성들이 있는 경우에 클래스 디자이너는 생성되는 TCML 문서에 이것들을 그대로 전달한다.

```
<Human>
    <name type="abc"/>
</Human>
<Student kindof="Human">
   ...
</Student>
```

그림 3 상속 관계

컴포지션은 재사용을 위한 또 다른 메카니즘이다. 컴포지션은 여러 개의 객체들을 조합해서 새로운 객체를 작성하는 것을 의미한다. TCML에서 컴포지션은 태그의 type 속성을 다른 클래스 이름으로 지정함으로써 가능하다. 그림 4에서 보여주는 ClassRoom 클래스는 많은 수의 Student들로 구성되어 있으며, 학급 평균을 위한 avg 멤버 필드를 포함한다. 다음 예에서 Student 태그에는 type 속성을 기술하지 않는다. 이렇게 type 속성이 없는 경우에는 데이타 타입은 태그 이름과 동일하다. number 속성은 컴포지션될 때 객체의 수를 지정하기 위해서 사용된다. 클래스 디자이너는 집합적인 데이타들을 처리하기 위해 sum(), numberof() 등의 함수들을 제공한다. sum() 함수는 모든 내용들을 합하기 위해서

사용되고, numberof() 함수는 개체의 수를 알아보기 위해서 사용된다.

```
<ClassRoom>
  <Student number="many"/>
  <avg type="99.99">
    avg = sum(Student.avg) /
          numberof(Student)
  </avg>
</ClassRoom>
```

그림 4 컴포지션 관계

TCML은 클래스 디자이너에 의해서 유효한 TCML 문서로 변환된다. 유효한 TCML은 DTD를 가지고 있으며, 유효성을 체크할 수 있다. 그림 5는 Student 문서를 유효한 TCML 문서로 변환한 것이다.

```
<?xml version='1.0' encoding='euc-kr' ?>
<!DOCTYPE class SYSTEM "tcml.dtd">
<class name='Student'>
  <field name='name' type='String'/>
  <field name='math' type='int'/>
  <field name='eng' type='int'/>
  <field name='avg' type='double'
                    readonly='yes'>
   avg = (math + eng) / 2
  </field>
</class>
```

그림 5 유효한 TCML 문서

그림 6은 유효한 TCML 문서를 위한 DTD를 보여준다. 유효한 TCML 문서의 가장 상위에는 class 태그를 사용하고, class 태그는 여러 개의 field를 포함한다. field 태그의 내용으로 수식을 정의할 수 있다.

유효한 TCML 문서는 클래스 디자이너에 의해서 자바 클래스로 변환된다. 변환된 클래스는 각 멤버 필드에 따라 setter와 getter 메소드를 갖게 된다. 그러나 readonly 속성이 있는 경우에는 getter 메소드만 만들어진다. XML 태그의 내용으로 수식을 기술한 경우에 이 내용들은 getter 메소드의 내용으로 포함된다. number 속성 때문에 집합적인 데이타를 갖는 경우에 클래스 디자이너는 자동적으로 add(), remove(), search(), len() 등의 메소드를 생성하게 된다. 예를 들어, Student의 avg 태그는 그림 7과 같은 getter 메소드를 생성하게 된다.

그림 8은 XML 예제 문서가 클래스 디자이너에 의해서 자바 프로그램으로 변환되는 과정을 보여준다. 스프

```
<!ELEMENT class (field*, method*)>
<!ATTLIST class
        abstract  NMTOKEN  #IMPLIED
        name      NMTOKEN  #REQUIRED
        kindof    NMTOKEN  #IMPLIED>
<!ELEMENT field (#PCDATA)>
<!ATTLIST field
        name      NMTOKEN  #REQUIRED
        type      CDATA    #IMPLIED
        number    NMTOKEN  #IMPLIED
        readonly  NMTOKEN  #IMPLIED>
<!ELEMENT method (#PCDATA)>
<!ATTLIST method
        name      NMTOKEN  #REQUIRED
        type      CDATA    #REQUIRED
        args      CDATA    #IMPLIED>
```

그림 6 TCML의 DTD

```
public  double  getAvg() {
    avg = (math + eng) / 2.0;
    return avg;
}
```

그림 7 생성된 코드

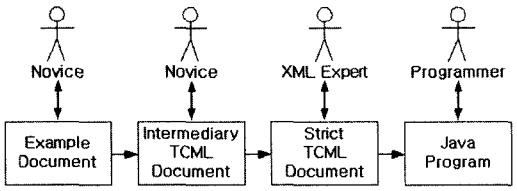

그림 8 XML 문서 변환 과정

레드시트 사용자는 자신의 능력에 맞게 각 단계에서 클래스를 정의할 수 있다. 예를 들어 전문가는 직접 자바 프로그램을 작성할 수 있고, XML에 익숙한 사용자는 유효한 TCML을 이용해서 클래스를 정의할 수 있다. XML이나 자바 프로그램에 전문적인 지식이 없는 사람의 경우에는 예제 문서나 느슨한 TCML을 이용해서 클래스를 정의할 수 있다.

3. 객체지향 스프레드시트

3.1 스프레드시트에서 클래스와 객체

XCEL은 스프레드시트 GUI에서 객체를 생성하고, 객체의 메소드를 호출할 수 있는 방법을 제공한다. 따라서 클래스 디자이너에서 정의한 클래스들은 스프레드시트에서 클래스의 인스턴스들을 생성하고, 메소드들을 호출함으로써 응용프로그램을 작성할 수 있다.

객체지향 스프레드시트에서 프로그램은 workbook으로 되어 있고, workbook은 클래스에 매핑된다. work-

book은 여러 개의 worksheet으로 구성되어 있으며, worksheet는 클래스의 메소드에 해당된다. worksheet는 그리드 형태에 데이타를 포함하는 블록들로 구성된다. 블록은 concept와 statement로 구성된다. 그림 9는 스프레드시트 프로그램의 문법을 보여준다.

```
workbook :: name attribute* worksheet+
worksheet :: name parameter result grid
grid :: block+
block :: [ concept ] statement
concept :: row
statement :: row+
row :: cell+
cell :: name expression
expression :: value | call | id
```

그림 9 스프레드시트 문법

정의 1. 클래스

클래스 C는 속성과 함수들로 구성되어 있으며, C = <name, A, F>의 튜플로 표현할 수 있다. A는 속성들의 집합이고, F는 함수들의 집합이다. 이름이 a인 속성을 C.A[a]라고 표현한다. 속성들의 개수를 len(C.A)이라고 한다. 속성 a의 값을 접근하기 위한 메소드는 C.F[a].get으로 표현하고, 속성 a의 값을 변경하기 위한 메소드는 C.F[a].set으로 표현한다. □

정의 2. 컨셉

스프레드시트에서 메타 데이타를 포함하는 셀들을 컨셉(concept)라고 한다. 셀 c의 위치를 c<i, j>라 하면, 컨셉에 포함되는 셀들의 집합은 Cell = { c | c<i, j+k>, $0 \le k \le n$, n은 컨셉의 데이타 수 }로 표현된다. 스프레드시트에서 컨셉에 매치되는 데이타를 포함하는 셀들을 컨셉 인스턴스(concept instance)라고 한다. □

정의 3. 클래스 선언

행 컨셉 A에서 i번째 셀 c_i의 이름을 c_i.name라고 하고, $0 \le i < len(C.A)$에서 $\forall c_i$.name에 대해서 C.A[c_i.name]가 존재하면, 행 컨셉 A를 클래스 선언이라고 한다. 클래스 선언은 연속적인 영역에 존재하게 되는데 이것은 서로 연관성이 높은 데이타들은 가까운 거리에 존재하기 때문이다[11]. □

객체지향 스프레드시트에서 클래스 선언은 컨셉 개념을 이용해서 셀들의 가로 방향으로 연속적인 영역에 표현된다. 또한 컨셉과 컨셉 인스턴스를 구별하기 위해서 클래스 선언 부분에는 굵은 실선으로 클래스에 포함된 셀들의 영역을 표시한다. 그림 10은 객체지향 스프레드시트에서 클래스 선언을 보여준다.

정의 4. 관계 ↠

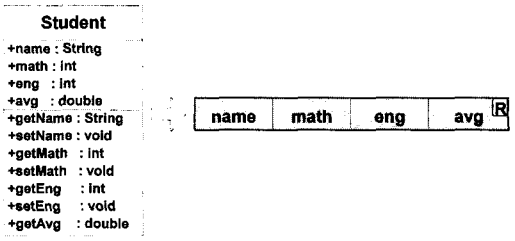

그림 10 객체지향 스프레드시트에서 클래스 선언

컨셉 P와 컨셉 인스턴스 R 사이에는 관계는 R ↠ P로 표현한다. 스프레드시트의 셀 리스트가 컨셉 인스턴스가 되기 위해서는 f(P, ↠) 연산이 수행되어야 한다. 클래스 선언과 객체의 관계도 ↠로 표현된다. 즉, 객체 O_c가 클래스 C의 인스턴스이면, 클래스와 인스턴스의 관계는 O_c ↠ C이다. 또한 f(C, ↠)는 클래스 C의 생성자 함수이다. □

객체지향 스프레드시트에서 컨셉과 컨셉 인스턴스는 연속적인 영역에 기술한다. 따라서, 클래스 선언과 클래스 인스턴스는 그림 11과 같이 표현된다. 그림 11에서 Student 클래스 타입의 객체가 2개 표현되어 있다. 하나는 name 속성이 "홍길동"이고, 다른 하나는 name 속성이 "김철수"이다.

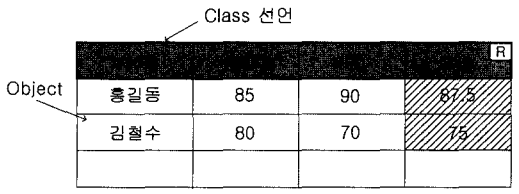

그림 11 클래스와 객체

클래스와 객체의 분리는 스프레드시트를 사실 매트릭스(facts matrix)와 솔루션 매트릭스(solution matrix)로 분리해야 한다는 DiAntonio[12]의 주장을 충실히 따르고 있다. 클래스는 속성과 메소드를 정의함으로써 솔루션 매트릭스의 역할을 수행하고, 객체는 실제 데이타를 가지면서 사실 매트릭스의 역할을 수행한다. 기존의 스프레드시트에서 분리된 사실 매트릭스와 솔루션 매트릭스는 각각 분리해서 보는 경우에 이해하기 거의 불가능하다는 문제점이 있다. 그러나 클래스와 객체의 분리는 DiAntonio의 주장을 따르면서도, 각각 분리해서 보더라도 쉽게 이해할 수 있는 장점이 있다.

정의 5. 관계 ≡

c_1과 c_2가 스프레드시트에서 셀의 내용이라고 가정하자. 만약 c_1과 c_2가 논리적으로 동일한 의미를 갖는다면, c_1과 c_2는 $c_1 > c_2$의 관계가 있다. C = (c_1, .., c_k)와 D

= (d_1, .., d_k)는 셀 내용들의 행 벡터일 때 c_i > d_i이고, $1 \le i \le k$ 이면, C > D이다. i는 행 번호이고, Γ는 행 컨셉 인스턴스이다. i.Γ는 i번째 행에서 Γ의 데이타 엔트리들의 튜플이다. i와 j가 행 번호일 때 (i.Γ > j.Γ) ∧ j = i + 1 인 경우에 i ⩾ j인 관계가 있다. 관계 ≡는 ⩾의 대칭적(symmetric)이고, 재귀적(reflective)이며, 이행적(transitive)인 클로져이다. □

정의 6. sob

Γ가 행 컨셉의 리스트일 때, Γ의 시작은 sob(Γ, i)로 표현되며, 최소값의 { j | j ≡ i}로 정의한다. □

사용자가 스프레드시트에서 클래스를 선언하고, 데이타를 입력하는 경우에 클래스 인스턴스들은 클래스의 생성자를 호출하고, 클래스의 set 메소드를 호출해서 데이타를 입력하여야 한다. 이러한 작업들은 알고리즘 1에 따라 이루어진다. 의존 그래프 DG는 현재 셀의 값이 변경됨으로써 변경해야 할 값들의 의존 관계를 그래프로 표현한 것이다.

알고리즘 1. 클래스 인스턴스 생성

```
현재 입력 셀 : c
컨셉 인스턴스 : CI
의존 그래프 : DG
BEGIN
    if c ∈ CI
        call Oc[c.name].set(c.value)
    else
        if PRE(c) ∈ CI
            sob(PRE(c), i)
            get concept P
            call f(P, →)
            add(c, CI)
            call Oc[c.name].set(c.value)
        else if PRE(c) ∈ P
            get concept P
            call f(P, →)
            add(c, CI)
            call Oc[c.name].set(c.value)
        end-if
    end-if
    call Oc[c.name].get()
    foreach node in DG
        call node.[x].get()
    end-foreach
END
```

현재 스프레드시트 시스템의 공식은 셀의 상대 주소와 절대 주소를 이용해서 계산을 수행하는데, 셀의 주소를 이용해서 계산하는 경우 프로그램은 이해하기 어려워지고, 에러 발생률은 높아진다. 만약 일반 스프레드시트 시스템을 사용한다면, 학생 평균 성적을 구하기 위해서는 다음과 같은 공식을 사용할 것이다.

= (B2 + C2) / 2 혹은 = ($B2 + $C2) / 2

이러한 공식은 의미 없는 셀의 주소를 사용하기 때문에 이해하기 어렵고, 입력 에러가 발생하기 쉽다. 예를 들어 엉뚱한 셀에 공식을 입력한다든가, 공식에 잘못된 셀의 주소를 입력하는 실수를 범하기 쉬워진다[13]. 절대 주소를 사용하는 경우에는 $ 심볼을 사용하는데, $ 심볼을 입력할 때 실수가 많기 때문에 절대 주소를 사용하는 경우에는 항상 공식이 올바르게 입력되었는지 체크해보아야 하는 부담이 있다[15]. 또한 셀의 주소를 사용하는 경우에 셀의 이동, 삭제, 삽입 등으로 셀의 위치가 변경되는 경우에 오류가 발생할 수 있다[7]. 그러나 객체지향 스프레드시트에서 알고리즘 1을 이용하면 학생의 수학과 영어 성적을 입력하면, 평균 점수가 자동적으로 계산된다. 따라서 사용자의 기계적인 오류를 줄여줄 수 있다. 또한 avg는 readonly로 설정되었기 때문에 사용자의 실수에 의해서 값이 변경될 염려가 없기 때문에 기계적 오류를 줄여줄 수 있다.

3.2 상속과 컴포지션

객체지향에서 상속은 재사용성 면에서 매우 중요한 의미를 가진다. 따라서 스프레드시트에서도 상속을 표현할 수 있는 개념이 필요하다. 스프레드시트 기반의 시각 프로그래밍인 Form/3 시스템은 유사성을 이용한 상속 개념이 소개하고 있지만[16,17], 본 논문에서는 기존의 객체지향 언어에서 사용되는 상속 개념을 일관되게 사용할 수 있도록 한다.

클래스의 상속을 속성과 오퍼레이션의 집합 측면에서 볼 때 자식 클래스는 부모 클래스의 속성과 오퍼레이션들을 모두 포함한다. 클래스의 속성은 스프레드시트의 셀에 표현되기 때문에 자식 클래스가 부모 클래스의 속성을 포함하는 것은 클래스를 표현하는 셀 범위의 포함관계로 나타낼 수 있다. 스프레드시트에서 상속은 역 삼각형 아이콘을 이용해서 표현한다. 그림 12는 Human 클래스로부터 상속받는 Student 클래스를 UML 클래스 다이어그램과 스프레드시트에서 표현하는 방법을 보여준다.

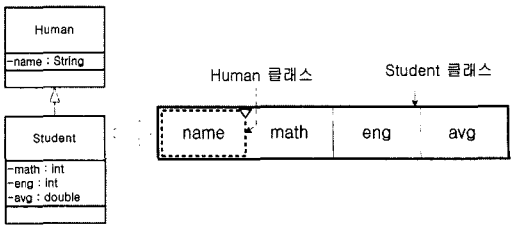

그림 12 클래스 상속

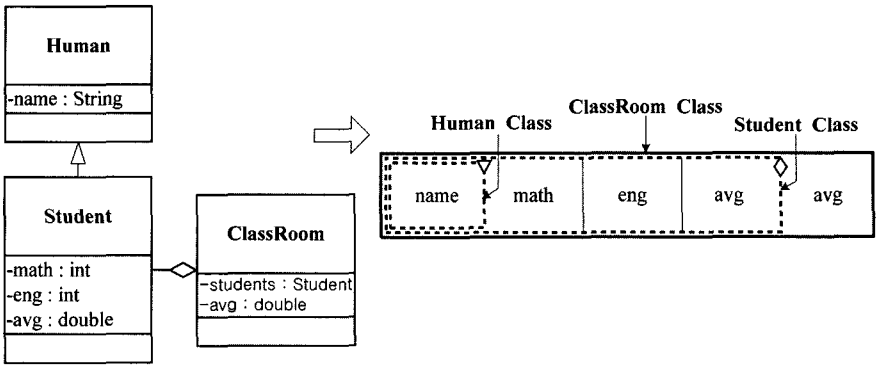

그림 13 클래스 컴포지션

객체들은 서로 연관성을 가지면서 서로 연결되어 있고, 복잡한 객체는 여러 개의 객체들로 구성되어 있다. 복잡한 객체는 하나 이상의 객체들로 구성되어 있다. 클래스의 컴포지션은 클래스의 속성이 다른 클래스 타입을 가지는 경우이다. 스프레드시트에서 컴포지션은 그림 13와 같은 형태로 표현한다. 컴포지션은 다이아몬드 형태의 아이콘으로 표현한다.

그림 13의 컴포지션 관계는 그림 14와 같이 간단하게 표현될 수 있다. Student 타입의 속성에 값을 입력하기 위해서는 별도의 윈도우나 풀다운 콤보 박스를 이용할 수 있다.

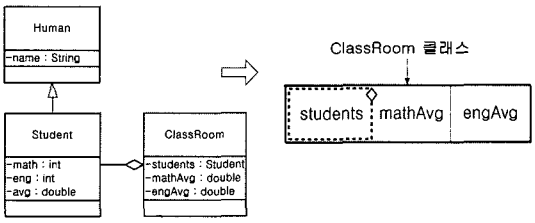

그림 14 간략하게 표현된 컴포지션

4. 시스템 구성

XCEL은 클래스 디자이너와 스프레드시트 부분으로 구성되어 있으며 자바와 Jython[24] 언어를 이용해서 구현되었다. 클래스 디자이너는 XML과 플로우차트를 이용해서 일반 사용자가 클래스를 정의할 수 있도록 지원한다. 클래스 디자이너를 이용해서 정의된 클래스들은 라이브러리로 구축된다. 그림 15는 클래스 디자이너의 모습이다. 사용자는 클래스 디자이너를 이용해서 XML 예제 문서로부터 자바 클래스 코드로 단계적으로 변환한다.

XCEL의 스프레드시트는 MVC(Model-View-Controller) 디자인 패턴을 따라 설계 및 구현되었다. 그림 16은 MVC 패턴에 따라 설계된 스프레드시트의 구조이다.

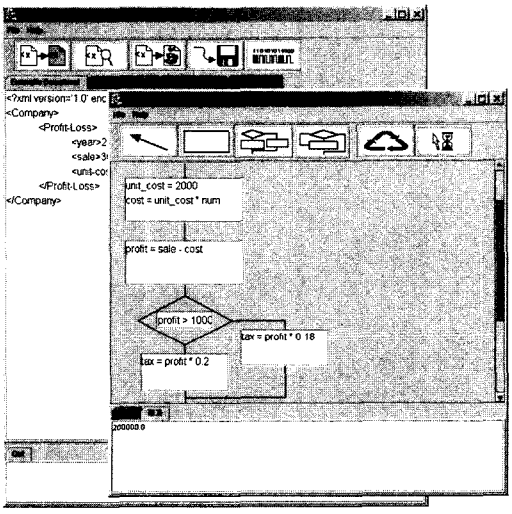

그림 15 클래스 디자이너

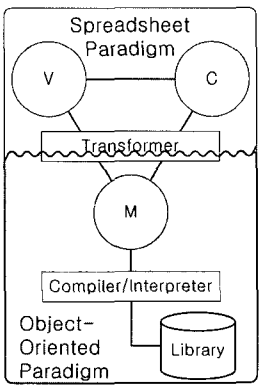

그림 16 객체지향 스프레드시트 구성

사용자와 상호 작용하는 뷰와 컨트롤러 부분은 일반 스프레드시트 시스템과 동일한 형태의 GUI를 제공함으로써 스프레드시트 프로그래밍 패러다임을 지원한다. 내부 로직과 데이타는 모델이 관리하게 되며, 실질적인 계산도 모델에서 이루어진다. 사용자가 입력한 데이타들은 뷰와 컨트롤러를 통해서 모델의 데이타에 영향을 주게 된다. 즉, 사용자가 입력된 데이타는 모델의 객체를 생성하고, 필요한 메소드를 호출하게 된다. 계산이 수행된 이후에 내부 상태는 뷰를 통해서 화면에 나타난다. 사용자가 스프레드시트를 이용해서 작성한 프로그램은 내부적으로 Jython 코드로 변환되고, Jython 인터프리터는 적절한 행동을 수행하게 된다. 그림 17은 스프레드시트로 작성된 프로그램 score가 어떻게 Jython과 JVM을 통해서 실행되는지를 tombstone 다이어그램[18]을 통해서 보여준다.

그림 18은 XCEL 시스템의 전체 화면을 보여준다. XCEL은 클래스 디자이너와 스프레드시트로 구성되어 있다.

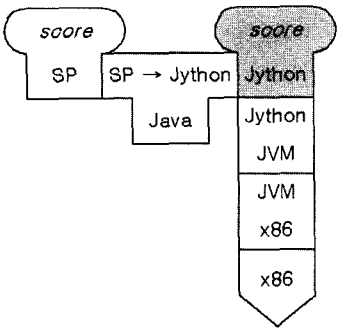

그림 17 XCEL의 tombstone 표현

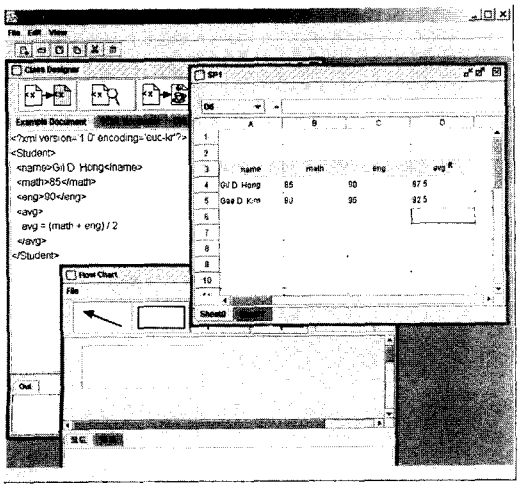

그림 18 XCEL 화면

5. 관련 연구

스프레드시트는 널리 사용되고 있으며, 스프레드시트의 단점을 해결하기 위한 연구도 많이 이루어져왔다. 스프레드시트 응용프로그램의 오류를 줄이기 위해서 체계적인 분석 설계를 적용하려는 연구[7,26]와 감사(監査)와 검증을 지원하는 연구[10,25]를 중심으로 수행되고 있다. 체계적인 분석 설계는 시스템의 논리와 생략 오류를 줄여주고, 응용프로그램의 질을 높여줄 수 있는 장점을 가지고 있다. 감사와 검증을 지원하는 방법은 스프레드에서 사용되는 공식의 의존 관계를 시각적으로 혹은 애니메이션을 이용해서 보여주는 방법이다. 이 방법은 세부적인 오류를 찾아 줄 수 있는 장점을 가지고 있다. 스프레드시트에서 사용되는 분석 설계 방법들이 구조적 방법론에 기초를 두고 있는데 비해서, 본 논문에서 소개하는 객체지향 스프레드시트는 객체지향 분석 설계에 기초를 두고 있다. XCEL은 클래스 디자이너를 통해서 스프레드시트 응용프로그램의 분석 설계를 수행하도록 지원한다.

스프레드시트의 언어적인 단점을 보완하고, 스프레드시트를 확장하기 위한 방법으로 스프레드시트와 다른 언어를 결합하기 위한 방법들이 연구되어왔다. 이 방법은 결합하는 언어의 강력한 컴퓨팅 능력, 풍부한 데이타 타입 등을 사용할 수 있는 장점을 가지고 있다. Michael[19]은 스프레드시트에 논리형 언어를 결합하기 위한 방법을 제시하고 있으며, Chris[20]와 Björn[21]는 스프레드시트에 함수형 언어를 결합하는 방법을 제시하고 있다. Spreadsheet 2000[22]은 스프레드시트와 데이타플로우 시각 프로그래밍을 결합하는 방법을 보여준다. 스프레드시트에 객체 지향 기술을 결합시킨 것으로는 Jocelyn의 Model Master(MM)[13,14]와 Smalltalk 기반의 ASP(Analytic Spreadsheet Package)[23]가 있다. MM은 스프레드시트의 셀에 직접 공식을 기술하는 대신에 프로그래머가 MM 프로그래밍 언어를 이용해서 모델을 기술하도록 한다. MM 컴파일러는 텍스트로 기술된 모델 명세를 스프레드시트의 공식으로 변환하는 작업을 수행한다. MM 언어는 객체와 상속을 지원하며, 상속을 통한 코드의 재사용이 가능하다. 하지만, MM은 스프레드시트의 편리한 입력 기능을 사용할 수 없고, 셀에 입력되는 공식이 다양해지면 실제로 사용하기 어렵다는 단점이 있다. ASP는 Smalltalk 언어를 기반으로 작성되었으며, 셀에 Smalltalk 객체를 포함할 수 있는 특징이 있다. ASP의 공식은 Smalltalk 프로그램으로 변환되고, Smalltalk 컴파일러를 통해 컴파일되는 점은 XCEL과 유사하다. 그러나 ASP는 클래스를 표현할 수 없고, 여러 개의 셀로 구성된 객체를 기술할 수 없는 단

점이 있다. 또한 Smalltalk를 사용하기 때문에 일반 사용자가 사용하기 어렵다.

현재 널리 사용되는 MS Excel은 사용자가 손쉽게 고 품질의 응용프로그램을 작성할 수 있도록 많은 기능들을 제공한다. 감사와 검증 기능은 물론 셀에서 사용되는 공식을 의존 관계 그래프로 표현하는 기능들을 지원한다. 또한 셀과 블록에 이름을 부여하는 기능과 셀에 입력을 막을 수 있는 기능들은 기계적 오류를 줄여줄 수 있도록 한다. 그러나 응용프로그램 개발 시 체계적인 분석 설계가 이루어지지 않는 경우에 논리와 생략 오류는 피할 수 없는 단점이 있다. 낮은 재사용성과 추상성, 유지 보수의 어려움과 같은 스프레드시트의 기본적인 단점들은 극복하지 못한다. 이러한 문제점을 해결하기 위해서 Excel에서 Visual Basic을 사용할 수 있는 방법을 제공하지만, 일반 사용자가 Visual Basic을 사용하기는 어렵다는 문제점이 있다.

6. 결 론

스프레드시트는 배우기 쉽고, 사용하기 쉽지만, 스프레드시트 시스템을 이용해서 개발된 응용프로그램은 에러를 많이 포함하고 있으며, 소프트웨어 재사용이 어렵고, 유지보수하기 힘들다는 단점을 가지고 있다. 이러한 문제점들 때문에 스프레드시트는 간단한 계산 프로그램 이외에 대형 소프트웨어 개발에는 적용되지 못하고 있다. 이러한 문제점들을 해결하기 위해서 본 논문에서는 스프레드시트에서 객체지향 프로그래밍이 가능한 XCEL이라는 객체지향 스프레드시트를 소개하였다. XCEL은 논리와 프리젠테이션을 분리하는 방법을 이용해서 설계 및 구현되었다. 프리젠테이션 부분은 일반 스프레드시트와 동일한 GUI를 지원하며, 내부적으로는 데이타를 객체지향 언어로 표현한다. 이러한 구분은 시스템을 간단하게 설계 및 구현할 수 있도록 한다.

XCEL은 스프레드시트 시스템에서 객체지향 프로그래밍을 지원하기 때문에 여러 가지 장점을 얻을 수 있다. 첫째로 스프레드시트 응용프로그램 개발에 객체지향 분석과 설계를 적용할 수 있다. 따라서 기존에 널리 사용되는 UML과 같은 분석 설계 도구를 사용할 수 있다. 이러한 체계적인 분석 설계는 스프레드시트의 논리, 생략 오류들을 줄여줄 수 있다. 둘째로 클래스를 이용한 라이브러리 구축과 재사용이 가능해진다. 구축된 라이브러리들은 이후에 다시 사용할 수 있으며, 상속 및 컴포지션을 위해서도 재활용할 수 있다. 셋째로 프로그램의 유지 보수가 쉬워진다. 기존 스프레드시트는 프로그램을 이해하기 위해서 모든 셀들을 방문하면서 전체적인 구조를 파악해야 하는데, 객체지향 스프레드시트는 클래스와 클래스들간의 관계만 파악함으로써 프로그램의 구조를 파악할 수 있다. 넷째로 사용자 입력 오류에 의한 기계적 오류를 줄여줄 수 있다. 다섯째로 프로그램의 문서화가 쉬워진다. 기존 스프레드시트는 문서화가 거의 없는데 비해, 객체지향 분석 설계를 통한 개발은 보다 정확하고 많은 문서화가 가능할 것이다. XCEL은 장점은 물론 단점들도 가지고 있다. 첫 번째 단점은 디버깅이 어렵다는 것이다. 이것은 XCEL에 디버깅 기능이 없기 때문에 Jython과 Java에서 제공하는 기본적인 디버깅 기능을 사용하여야 하고, XML로부터 생성된 코드를 디버깅하여야 하기 때문에 에러를 찾기 어려운 문제점이 있다. 두 번째 문제점은 임으로 새로운 셀들을 삽입할 수 없다는 점이다. 만약 클래스와 인스턴스들의 데이타가 있는 부분에 새로운 셀을 삽입하는 경우에 기본 규칙이 깨지기 때문에 임의의 위치에 셀을 추가하는 것은 XCEL에서는 금지되어 있다.

참 고 문 헌

[1] D. Bricklin and B. Frankston, "VisiCalc: Information from its creators, Dan Bricklin and Bob Frankston", 1999. available at http://www.bricklin.com/visicalc.htm

[2] Ed Haui-hsin Chi et al, "A Spreadsheet Approach to Information Visualization," in *Proc. of Information Visualization Symposium*, pp. 17-24, 1997.

[3] Jeff A. Johnson et al, "Ace: Building Interactive Graphics Applications," in *Comm. of the ACM*, Vol. 36, No. 4, pp. 41-55, 1993.

[4] Panko R. R., Spreadsheet Research(SSR) Website (http://panko.cba.hawaii.edu/ssr/) Honolulu, Hawaii: University of Hawaii.

[5] Coopers & Lybrand in London. Description available at http://www.planningobjects.com/jungle1.htm

[6] Raymond J Butler, "Is This Spreadsheet a Tax Evader? How H.M. Customs & Excise Test Spreadsheet Applications," in *Proc. of the 33rd Hawaii International Conference on System Sciences*, 2000.

[7] Kamalasen Rajalingham et al, "Quality Control in Spreadsheets: A Software Engineering-Based Approach to Spreadsheet Development," in *Proc. of the 33rd Hawaii nternational Conference on System Sciences*, 2000.

[8] Panko R.R and Halverson R.P. Jr., "Spreadsheets on Trial: A Framework for Research on Spreadsheet Risks," in *Proc. of the 29th Hawaii International Conference on System Sciences*, 1996.

[9] Jorma Sajaniemi, Markku Tukianinen and Jarmo Vaisanen, "Goals and Plans in Spreadsheet Calculation," in *Technical Report A-1999-1*, Dept. of Computer Science, University of Joensuu, 1999.

[10] Takeo Igarashi et al, "Fluid Visualization of Spreadsheet Structures," in *IEEE Symposium on Visual Languages*, pp. 118-125, 1998.
[11] Tomas Isakowitz, Shimon Schocken and Henry C. Lucas, Jr, "Toward a Logical/Physical Theory of Spreadsheet Modeling," in *ACM Transactions on Information Systems*, Vol. 13, No. 1, pp. 1-37, Jan, 1995.
[12] A.E. DiAntonio, *Spreadsheet Applications*, Prentice-Hall, 1986.
[13] Jocelyn Paine, "Model Master: an object-oriented spreadsheet front end," in *Proc. of CALECO97*, 1997. available at http://www.ifs.org.uk/~popx/
[14] Jocelyn Paine, "Ensuring Spreadsheet Integrity with Model Master," in *Proc. of EuSpRIG*, 2001. available at http://www.ifs.org.uk/~popx/
[15] Hendry D.G. and Green T.R.G, "Creating, comprehending, and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model," in *International Journal of Human-Computer Studies*, Vol. 40, No. 6, pp. 1033-1065, 1994.
[16] M. Burnett and A. Amblerr, "Interactive Visual Data Abstraction in a Declarative Visual Programming Language, in *Journal of Visual Languages and Computing*," pp. 29-60, Mar., 1994.
[17] Walpole Djang et al, "Similarity Inheritance: A New Model of Inheritance for spreadsheet VPLs," in *IEEE Symposium on Visual Languages*, pp. 134-141, 1998.
[18] Jay Earley and Howard Sturgis, "A Formalism for Translator Interactions," in *Comm. of ACM*, Vol. 13, No. 10, pp. 607-617, 1970.
[19] Michael Spenke and Christian Beilken, "A Spreadsheet Interface for Logic Programming," in *CHI*, pp. 75-80, 1988.
[20] Chris Clack and Lee Braine, "Object-oriented functional spreadsheets," in *Proc. of the Glasgow Workshop on Functional Programming*, pp. 1-12, 1997.
[21] Björn Lisper and Johan Malmström, "Haxcel: A Spreadsheet Interface to Haskell," in *Proc. of International Workshop on the Implementation of Functional Languages*, pp. 206-222, 2002. available at http://www.mrtc.mdh.se/
[22] Steve Wilson, "Building a Visual Programming Language," in *MacTech*, Vol. 13, No. 4, 1997. available at http://www.mactech.com/
[23] Piersol, K. W., "Object Oriented Spreadsheets: The Analytic Spreadsheet Package," in *OOPSLA '86*, pp. 385-390, 1986.
[24] Jim Hugunin, "Python and Java: The Best of Both Worlds," in *Proc. of the 6th International Python Conference*, 1997. available at http://www.jython.org/
[25] Gregg Rothermel, et al, "A Methodology for Testing Spreadsheets," in *ACM Transactions on Software Engineering and Methodology*, Vol. 10, No. 1, pp. 110-147, 2001.
[26] Brian Knight, David Chadwick and Kamalasen Rajalingham, "A Structured Methodology for Spreadsheet Modelling," in *Proc. of EuSpRIG*, pp. 43-50, 2001.

최 종 명
정보과학회논문지 : 소프트웨어 및 응용
제 30 권 제 3 호 참조

유 재 우
정보과학회논문지 : 소프트웨어 및 응용
제 30 권 제 3 호 참조