Objectsheet: 객체 지향 스프레드 쉬트

최종명, 박권, 신경희, 유재우
숭실대학교 컴퓨터학과
jmchoi@it.soongsil.ac.kr

Objectsheet: Object-oriented Spreadsheet

Choi Jongmyung, Park Kweon, Shin Kyounghee, Yoo Chaewoo
Dept. of Computing, Soongsil Univ.

요 약

스프레드쉬트는 사용하기 쉽기 때문에 가장 널리 사용되는 프로그래밍 도구이다. 그러나, 스프레드쉬트로 개발되는 프로그램들은 비 구조적인 방법으로 개발되기 때문에 대형 프로젝트에 적용하기 어렵고, 프로그램에 논리적인 오류들이 많이 포함되어 있다. 또한 스프레드쉬트로 개발된 프로그램은 읽기 어렵고, 디버깅 및 유지 보수하기 어려운 단점이 있다. 이러한 문제를 해결하기 위해서 본 논문에서는 스프레드쉬트에 구조적인 개발 방법을 적용하기 위해 클래스와 객체 개념을 추가하고, 프로그램의 이해 및 개발을 쉽게 하기 위해서 데이터플로우 개념을 지원한다. 스프레드쉬트에서 객체 개념은 객체지향 분석, 설계 및 프로그래밍을 가능하게 하고, 데이터플로우 개념은 데이터의 의존 관계 및 데이터 흐름을 시각적으로 보여주기 때문에 프로그램의 개발 및 유지 보수를 쉽게 한다.

1. 서론

스프레드쉬트는 사용의 편리성과 상호 대화적인 특성 때문에 회계, 통계, 이미지 프로세싱[9], GUI 생성[10] 등 다양한 분야에서 사용되고 있다. 본 논문에서는 스프레드쉬트를 이용해서 개발되는 프로그램을 스프레드쉬트 프로그램이라 부르기로 한다.

스프레드쉬트를 이용한 프로그램 개발은 상대적으로 용이하지만 이에 따른 문제점들도 발생하고 있다. 첫째로 스프레드쉬트 프로그램이 대형화되고 있는데 비해 프로그램 개발이 비 구조적으로 진행된다는 점이다. 두 번째 문제는 많은 스프레드쉬트 프로그램들이 내부적으로 오류를 포함하고 있다는 것이다. Panko[3]의 연구에 의하면 모든 스프레드쉬트 프로그램의 20-40%가 오류를 가지고 있고, Coopers[4]는 150행 이상의 스프레드쉬트 프로그램의 90%가 오류를 포함하고 있다고 밝히고 있다. 셋째로 스프레드쉬트 프로그램은 읽기 어렵고, 유지 보수하기 힘들다.

이러한 문제점들을 해결하기 위해서 여러 가지 방법들이 연구되어왔는데 이 중에서 가장 주목할 만한 것은 스프레드쉬트에 객체지향 기술을 적용하는 것이다. 그러나, 현재까지 연구된 방법들은 스프레드쉬트에 직접 적용하기 어려운 단점들을 가지고 있다. 이러한 문제들을 해결하기 위해서 본 논문에서는 스프레드쉬트에 객체지향과 데이터플로우 패러다임을 결합한 방법을 소개한다.

스프레드쉬트에 객체지향 기술을 적용함으로서 구조적인 개발 방법을 적용할 수 있고, 대규모 프로그램 개발을 용이하게 지원한다. 또한 스프레드쉬트에서 객체 데이터의 흐름과 연산을 적용함으로서 프로그램을 쉽게 작성할 수 있고, 읽기 쉽게 표현할 수 있다. 데이터플로우 패러다임의 적용은 또한 프로그램의 에러를 줄일 수 있다는 장점을 가지고 있다.

본 논문은 2장에서 관련 연구들을 소개하고, 3장에서 objectsheet에 대해서 기술한다. 4장에서는 결론를 밝힌다.

2. 관련 연구

2.1 Model Master(MM)

Jocelyn[1]의 기본 아이디어는 스프레드쉬트의 셀에 직접 공식을 기술하는 대신에 프로그래머가 MM 프로그래밍 언어를 이용해서 모델을 기술하도록 하는 것이다. MM 컴파일러는 텍스트로 기술된 모델 명세를 스프레드쉬트의 공식으로 변환하는 작업을 수행한다. MM은 스프레드쉬트의 편리한 입력 기능을 사용할 수 없고, 사용하기 불편하다는 단점이 있다.

2.2 Spreadsheet 2000

Spreadsheet 2000[7]은 객체 기반(object-based)의 데이터플로우를 지원하는 시각 프로그래밍 언어이다. Spreadsheet 2000은 Prograph[8]를 이용해서 작성되었으며, Prograph의 영향을 받아 스프레드쉬트에 데이터플로

우 패러다임을 적용하고 있다.

3. Objectsheet

3.1 객체 추상화

프로그램이 대형화됨에 따라 비 구조적인 방법에 의해 개발되는 스프레드쉬트 프로그램은 개발 및 유지 보수에서 많은 문제들을 야기시킨다. 이러한 문제점들을 해결하기 위한 Objectsheet의 기본적인 아이디어는 스프레드쉬트의 데이터와 공식을 각각 별개로 처리하는 것이 아니라 식별성(identity)을 가지는 객체로 추상화시키는 것이다. 따라서 스프레드쉬트의 셀에 입력되는 데이터들은 더 이상 독립적인 데이터들이 아니라 한 객체의 속성 값으로 표현되고, 공식은 객체의 메소드로 표현된다.

한 객체의 속성들은 스프레드쉬트의 연속적인 영역에 표현되고, 객체의 속성은 읽기 전용인 경우에는 getter 메소드만, 읽기/쓰기인 경우에는 getter와 setter 메소드가 자동적으로 생성된다. 객체의 속성 값은 getter 메소드에 의해 셀에 반영되고, 역으로 셀에서 값의 변경은 setter 메소드를 이용해서 객체의 속성에 반영된다.

3.2 자료형 표현

스프레드쉬트의 데이터는 자료형을 기술하지 않는데, 이것은 프로그램의 유연성을 높여주는 장점이 있지만, 에러를 찾기 어렵게 하는 요인이 된다. Objectsheet에서는 이러한 문제를 해결하기 위해서 객체의 속성으로 사용되는 셀은 항상 자료형을 기술하도록 한다. 자료형은 셀의 우측 상단에 아이콘을 이용해서 표현된다.

3.3 이름에 의한 참조

스프레드쉬트의 공식은 현재 셀의 상대 주소와 절대 주소를 이용해서 계산을 수행한다. 이러한 경우에 셀의 이동, 삭제, 삽입 등으로 셀의 위치가 변경되는 경우에 오류가 발생할 수 있다[6]. 이러한 오류는 개발자나 유지 보수하는 입장에서 매우 찾기 어렵다. 이 문제를 해결하기 위해서 Objectsheet에서는 셀의 위치 주소를 사용하는 대신에 객체의 속성 이름을 사용한다. 이름을 사용하면 셀의 위치가 변경되더라도 오류가 발생하지 않게 된다.

3.4 가시성과 네임 스페이스

스프레드쉬트 프로그램에서 공식을 사용하는 경우에 임의의 셀에 있는 값을 사용할 수 있는데, 이것은 프로그램의 복잡도를 증가시키는 요인이 된다. 이것은 일반 프로그래밍에서 전역 변수를 많이 사용하는 것과 유사하다. 따라서 프로그램의 응집도를 높이기 위해서는 공식에서 사용할 수 있는 셀들의 네임스페이스를 제한하여야 한다. 이러한 문제를 해결하기 위해서 Objectsheet에서는 객체의 메소드는 객체의 속성과 특별히 지정한 다른 객체의 속성 값을 사용할 수 있도록 제한을 둔다. 외부에서 객체의 속성을 접근할 수 있도록 하기 위해서 다른 일반 객체지향 언어에서와 마찬가지로 Objectsheet에서는 가시성을 지원한다.

3.5 Objectsheet 예제

스프레드쉬트에서 객체 속성의 이름과 자료형을 지정하고, 연속된 영역을 선택해서 클래스로 선언한다. 그림 1은 Student 클래스를 지정하는 방법을 보여준다. Student 클래스는 문자열 타입의 name, 학생의 성적을 위한 정수형의 math, eng 속성과 평균을 위한 실수형의 avg 속성을 가진다. 클래스의 속성을 기술한 다음에 연속적인 영역을 선택하는 경우에 클래스의 기타 속성과 메소드를 기술할 수 있는 다이얼로그가 나타난다. 클래스를 선언한 뒤에 입력된 데이터들은 클래스의 인스턴스로 인식된다.

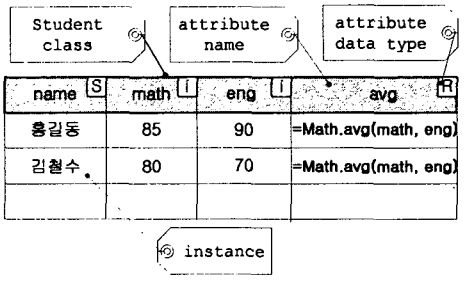

그림 1. 객체 표현

3.6 데이터플로우

스프레드쉬트의 공식은 다른 셀의 값을 이용해서 작업한다. 이러한 값의 참조는 의존 관계를 형성하지만, 스프레드쉬트에는 이러한 내용들이 표현되지 않기 때문에 프로그램 작성이나 유지 보수에 많은 어려움이 있게 된다. 이러한 문제점을 해결하기 위해서 Takeo[5]는 데이터의 의존성을 화살표를 이용해서 표현함으로서 프로그램의 유지 보수를 쉽게 하기 위한 방법을 지원한다.

Objectsheet는 스프레드쉬트에서 데이터의 의존 관계를 데이터플로우로 표현함으로서 프로그램의 이해 및 개발을 용이하게 한다. Objectsheet에서 데이터의 흐름은 화살표를 이용해서 표현된다.

그림 2는 Objectsheet에서 데이터플로우의 사용법을 보여준다. 좌측 상단의 =DB.sql("select ..")는 데이터베이스에서 SQL 질의어를 이용해서 결과를 추출한다. 추출된 결과는 데이터플로우에 의해 객체를 생성하기 위해서 사용된다. 회색으로 채워진 부분은 Student 클래스를 선언한 것을 보여준다. Student 클래스는 name, math, eng, avg의 속성을 갖도록 선언되어 있다. 이때 SQL로부터 추출된 데이터는 Student의 각 속성에 값을 할당하면서 객체들을 생성하고, 생성된 객체들은 배열로 관리

된다. 이처럼 배열 객체를 표현하기 위해서 여러 개 선을 클래스 선언 부분에 기술한다. 이렇게 생성된 Student 객체들은 Chart 클래스의 bar() 메소드를 이용해서 막대 그래프로 표현될 수 있고, Math 클래스의 sort() 메소드를 이용해서 내림차순으로 정렬될 수 있다. 정렬된 데이터는 다시 HTML 문서로 출력된다. 이렇게 데이터플로우는 프로그램을 간단하게 표현할 수 있도록 도와주고, 프로그램을 이해하기 쉽게 한다.

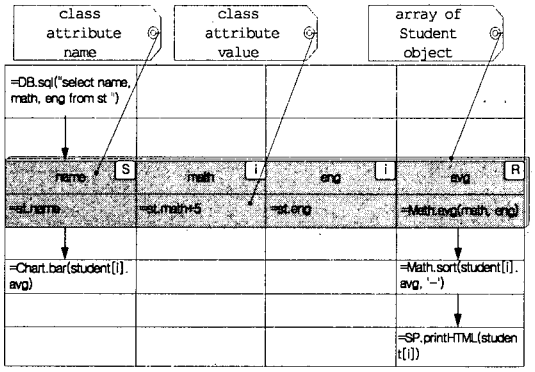

그림 2. Objectsheet에서 데이터플로우

4. 결론

스프레드쉬트는 배우기 쉽고, 사용하기 쉽지만 프로그램이 점차 대형화되어 가고 있기 때문에 여러 가지 문제들이 발생하고 있다. 첫째로 프로그램이 비 구조적으로 개발되기 때문에 대형 소프트웨어를 개발하기 어렵고, 소프트웨어 가체에 논리적인 오류들을 가지고 있다. 두 번째 문제는 스프레트쉬트는 데이터만 화면에 보여지고 공식은 보여지지 않음으로서 프로그램 이해와 디버깅 및 유지 보수를 어렵게 한다.

이러한 문제들을 해결하기 위하여 본 논문에서는 스프레드쉬트에 객체지향 기술을 적용하는 방법에 대해 설명하고 있으며, 이와 함께 데이터플로우 패러다임을 적용하는 방법을 보여준다. 스프레드쉬트에서 클래스의 정의와 객체의 사용은 스프레드쉬트 프로그램 개발에서 객체지향 분석, 설계 및 프로그래밍을 적용할 수 있는 방법을 제공한다. 또한 데이터의 의존 관계를 데이터 흐름으로 시각적으로 표현함으로 프로그램의 이해를 도와주고, 디버깅 및 유지 보수를 쉽게 도와준다. 스프레드쉬트에서 객체지향과 데이터플로우 개념의 도입은 기존의 비 구조적인 스프레드쉬트 개발의 문제점들을 어느 정도 해결할 것으로 보인다.

향후 연구 과제로는 Objectsheet에 객체지향과 스프레드쉬트에 적합한 폴리모피즘[11]과 상속성[12]을 지원하는 것이다.

5. 참고 문헌

[1] Jocelyn Paine, "Model Master: an object-oriented spreadsheet front end", In Proceedings of CALECO97, Sep. 1997. available at http://www.ifs.org.uk/~popx/mm.html

[2] Tomas Isakowitz, Shimon Schocken and Henry C. Lucas, Jr, "Toward a Logical/Physical Theory of Spreadsheet Modeling", ACM Transactions on Information Systems, Vol 13, No. 1, pp. 1-37, Jan, 1995.

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

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

[5] Takeo Igarashi, Jock D. Mackinlay, Bay-Wei Chang, Polle T. Zellweger, "Fluid Visualization of Spreadsheet Structures", IEEE Symposium on Visual Languages, 1998.

[6] Kamalasen Rajalingham et al, "Quality Control in Spreadsheets: A Software Engineering-Based Approach to Spreadsheet Development", in Proceedings of the 33rd Hawaii Intl Conf. on System Science, pp. , 2000.

[7] Casady & Greene, Inc. Spreadsheet 2000, Online HTML document, 1997. available http://www.emer.com/s2k/

[8] P. T. Cox, F. R. Giles, T. Pietrzykowski, "Prograph", in Visual Object-Oriented Programming, Manning Pub. pp. 45-66, 1995.

[9] Ed Haui-hsin Chi et al, "A Spreadsheet Approach to Information Visualization", ACM Proceedings, 1997.

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

[11] G. Wang, A, Ambler, "Invocation Polymorphism", in IEEE Sysmposium on Visual Languages, pp. 83-90, Sep. 1995.

[12] Walpole Djang et al, "Similarity Inheritance: A New Model of Inheritance for spreadsheet VPLs", in IEEE Symposium on Visual Languages, Sep. 1998.