韓國 컴퓨터情報學會 論文誌
第11卷 第5號, 2006. 11.

2006-11-5-1-16

재사용 단위 기반 소프트웨어 개발 방법에 의한 설계 자동화 시스템

이 미 라*

An Automatic Graphic Drawing System by Software Development Approach based on Reusable Units

Yi Mi Ra*

요 약

복잡하고 규모가 큰 소프트웨어 개발 요구가 증가하는 것에 비해 상대적으로 느린 소프트웨어 개발 속도의 대안으로 여러 가지 개발 방법론이 소개되어 왔으며, 이들 대부분의 핵심 개념으로 코드의 재사용성이라는 속성을 포함하고 있다. 이러한 코드의 재사용적인 측면에서 소개되는 소프트웨어 개발 방법론들은 시간이 지남에 따라 재사용될 코드의 단위가 점점 커져 왔다. 한편, 제조 공정 관련한 설계 자동화 시스템은 단순하게 제도판을 대신하는 기능을 넘어 설계 이후 공정에 활용 될 다양한 데이터 생성까지를 목표로 하고 있어 규모가 크고 복잡해지는 소프트웨어의 한 분야이다. 본 논문에서는 코드의 재사용성을 위한 여러 단위 - 객체, 컴포넌트, 모듈, 상업용 소프트웨어 - 기반의 소프트웨어 개발 방법이 실제로 어떻게 반영될 수 있는지를 금속제품을 위한 설계 자동화 시스템 개발 사례로 소개하고자 한다.

Abstract

The demands for the development of large scale software systems are being increased. Various software development methods have been introduced to meet these demands. The unit size of the codes that are reused is increasing in the development methods. These units reflect the concept of software reusability and can be identified as the object, component, and module. Recent trend in the development of a large scale software systems treats a commercial program as a unit to be reused. This approach lies along the same line as the identification of above three types of units. This paper shows how the above four types of units for enhancing the software reusability can be applied to the development of automatic graphic drawing System for a metal production.

▸ Keyword : 객체 지향(Object Oriented), 재사용성(Reusability), COTS, CAD

• 제1저자 : 이미라
• 접수일 : 2006.10.01, 심사일 : 2006.11.13, 심사완료일 : 2006.11.18
* 목포해양대학교 해양전자통신공학부

Ⅰ. 서 론

복잡하고 규모가 큰 소프트웨어 시스템에 대한 요구는 지속적으로 증가되고 있다[1,2]. 이렇게 증대하는 요구에 비해 더딘 소프트웨어 개발 속도의 대안으로 여러 가지 개발 방법론이 소개되어 왔으며, 그것들의 핵심 개념으로 코드의 재사용성을 포함하고 있다[1,2,3,4]. 코드의 재사용적인 측면에서 소개되는 소프트웨어 개발 방법론들은 시간이 지남에 따라 재사용 될 코드의 단위가 점점 커져 왔다. 이러한 코드의 재사용 단위를 표현하는 용어로는 객체(object), 컴포넌트(component), 모듈(module)이 있으며, 이들과는 좀 다른 형태로 목표 시스템의 한 구성요소로 사용되는 상업용 소프트웨어를 들 수 있다.

한편, 제조 공정 관련한 설계 자동화 시스템은 규모가 크고 복잡한 소프트웨어의 한 분야이다. CAD 시스템의 초기에는 제도판을 컴퓨터로 대신하는 형태의 제도 위주의 시스템들이 많았다. 하지만, 시간이 지남에 따라 단순한 제도판 기능에서 벗어나 사용자가 요구하는 조건에 부합되는 제품 스펙을 자동 선정하는 기능과 선정된 요소에 관한 그래픽 데이터를 자동으로 제도하고, 그려진 그래픽 데이터 스펙에 대한 관련 정보가 자동으로 이식되는 형태의 시스템을 CAD 시스템으로 인식하고 있다[5,6]. 더 나아가, 설계된 CAD 데이터를 분석하여 물량을 자동으로 산출하거나, CAD 데이터를 설계 이후의 여러 과정에 활용할 수 있도록 하는 FMS(Facility Management System)을 포함하는 CAD 환경 실현을 목표로 하고 있다. 즉, CAD 시스템은 초기의 단순 그리기 기능에서 출발하여 자동 설계로 발전하고 점차 MIS 및 CAM에 활용 가능한 형태의 문서 작성으로 그 자리를 굳혀가야 한다는 것이다[7].

본 논문에서는 소프트웨어 개발에 있어 객체, 컴포넌트, 모듈, 애플리케이션 단위의 재사용성이 어떻게 반영될 수 있는지를 그레이팅이라는 금속제품을 위한 설계 자동화 시스템 개발 사례로 소개하고자 한다.

2. 연구 배경

2.1 관련 연구

객체(Object)는 시스템에서 다루는 데이터와 그 데이터를 조작하는 함수를 함께 묶어 정의한 단위를 말한다[1,8]. 프로그램 작성자는 정의된 객체의 인스턴스(instance)를 이용하여 다수개의 객체를 손쉽게 만들어 사용하여 좀 더 효율적인 개발이 가능하다.

컴포넌트(component)는 시스템의 구성요소로서 유사 기능들의 묶음 단위의 프로그램을 말한다[1,4,9]. 이러한 컴포넌트들이 모여 유기적인 연관성을 갖고 하나의 독립된 기능을 수행하는 프로그램을 생성 할 수 있고, 해당 기능이 필요한 다른 프로그램의 컴포넌트로도 재사용된다.

위의 객체와 컴포넌트는 그 개념이 명확한 것에 비해 모듈(module)은 여러 가지 의미로 쓰여 특정 단위로 정의하기가 쉽지 않다. 그러나 모듈은 전체 시스템의 독립적인 특징을 갖는 한 부분으로서 논리절차나 데이터구조를 갖는 단위이며, 완성될 시스템을 구성하는 작은 덩어리라는 일반적인 정의는 가능하다[10]. 이는 한 프로그램을 독립된 단위로 분할하여 복잡한 문제를 작고 간단한 문제로 나누어 개발하고 테스트까지 할 수 있어 프로그램 개발에 효율적이다[1,2].

더 나아가 최근에는 상업용 소프트웨어를 전체 시스템을 위한 하나의 구성요소로 하여 목표 시스템을 개발하는 경우가 늘어가고 있는데[3,11,12] 앞의 세 가지와는 다소 차이가 있지만, 이 또한 실행코드의 재사용성이라는 측면에서는 일직선상에 있다 하겠다.

본 연구에서는 상대적으로 '단위'에 제약이 적은 개념인 '모듈' 용어를 컴포넌트보다 큰 코드 단위로 사용하여, 객체, 컴포넌트, 모듈, 상업용 소프트웨어의 재사용 접근 방법을 보이고자 한다.

2.2 그레이팅 생산 시스템

본 연구에서 다루는 문제의 대상 영역은 그레이팅이라는 금속제품을 생산하는 제조공정에서의 복잡하고 반복적인 설계 과정이다. 그레이팅이란 도로변의 배수로를 덮거나, 공장 또는 선박에서 바닥재로 사용되는 격자무

늬의 금속품을 일컫는다. 그림 1은 이러한 그레이팅의 모양 및 사용 예를 보인 것이다.

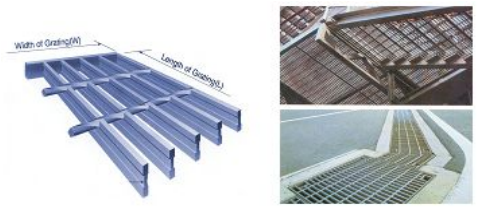


그림 1. 그레이팅 모양 및 용례
Fig. 1. Grating Shape (left) and Usage (right)

설계 작업은 크게 세 과정으로 구분되는 단계를 거치며 이루어지는데, 이 세 과정은 철골 배치(frame allocation) 단계, 그레이팅 배치(grating allocation) 단계, 도면자료 생성(drawing data generation) 단계이다. 철골배치 단계에서는 그레이팅이 배치될 구획을 따라 그레이팅을 지지할 철골이 설치될 도면을 그리는 것이고, 이를 기준으로 구획별로 그레이팅들을 어떤 모양으로 설치할 것인지를 그리는 것이 그레이팅 배치 단계이며, 마지막으로 한 도면 내에 그려진 그레이팅들의 통계적인 정보와 개별적인 그레이팅 품목의 상세 정보를 또 다른 형태의 도면(제작에 사용될 단품도면, 제품 검사에 사용될 테스트 도면)으로 그리는 작업이 도면자료 생성 단계이다.

이러한 과정에서 설계자들은 다음과 같은 문제점들을 갖는다.

- 그레이팅 배치 과정의 복잡성
- 세부 도면 작업에서 그레이팅의 반복 설계
- 빈번한 전체 도면 변경시 세부도면 재작업
- 전체 도면과 세부 도면의 불일치 오류
- 설계 기간의 장기화로 인한 납기지연

본 논문은 위의 문제점들에 대한 설계자들의 해결 요구에 따라 자동화된 도면 설계 시스템(GDS; automatic Graphic Drawing System)[13,14]의 개발 과정을 재사용 단위 기반 접근 측면에서 다룬다. 개발된 GDS를 통해, 설계자는 철골 배치도를 그린 후, 그레이팅을 배치하고자 하는 영역에 특정 조건들을 입력하여 자동으로 그레이팅이 배치되도록 하고, 그레이팅 배치 결과를 기준으로 간단한 명령을 줌으로써 도면의 통계적인 정보 및 개별 그레이팅의 상세 도면을 얻을 수 있다.

3. 재사용 단위 기반 설계

GDS는 큰 단위의 시스템 관점에서 점차 작은 단위로 설계 과정이 이루어진다. 설계 과정을 크게 네 단계로 살펴보면, 상업용 소프트웨어와 전체 목표 시스템의 연동, 자체 개발 모듈 설계, 각 모듈들을 구성하는 기능 단위의 컴포넌트 설계, 마지막으로 데이터 접근 및 여러 구성 요소들을 객체화 하여 다루기 위한 클래스 설계 단계이다.

3.1 상업용 소프트웨어

GDS는 AutoCAD 기반의 자동 설계 프로그램이면서 특정 도메인에 맞도록 한 응용 프로그램이기 때문에 기본적으로 CAD와의 연동을 고려해야 한다. AutoDesk에서는 C언어 사용 개발자를 위해 AutoCAD Development System(ADS)를 제공하고, 이를 발전시켜 C++ 사용하는 CAD 기반 응용 프로그램 개발자를 위한 객체지향형 라이브러리인 ObjectARX를 제공한다[15]. GDS에서는 AutoCAD와의 연동을 위해 ObjectARX를 이용한다. 그림 2는 이러한 요소들의 관계를 보여준 것이다.

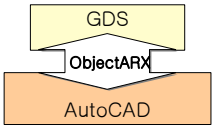


그림 2. AutoCAD 와 GDS
Fig. 2. AutoCAD and GDS

3.2 모듈

그림 3은 GDS의 구성도를 나타낸 것이다. GDS 내에는 여러 개의 모듈 단위 구성요소가 있다. 우선 사용자와의 원활한 상호작용을 위한 *GUI* 모듈, 그레이팅을 배치하기 위한 이전 작업을 담당하는 *철골배치기(frame allocator)* 모듈, 철골 배치 이후 그레이팅 배치를 위한 *그레이팅배치기(grating allocator)* 모듈, 배치된 그레이팅 내용을 이용하여 여러 가지 도면 관련 데이터를 만들어내는 *도면자료생성기(drawing data generator)* 모듈이 있다. 각 모듈은 독립적으로 개발되며 테스트가

이루어진 후에 전체 시스템을 위해 통합된다. 각 모듈의 기능을 좀 더 살펴보자.

*철골배치기*는 그레이팅 설계를 하기 위한 기초도면 생성을 위한 모듈이다. 이 모듈은 설계자로부터 도면 정보를 입력 받아 기본적인 레이아웃(layout)이 생성된 도면이 초기 도면이 자동 생성하고, 생성된 도면에 GUI를 매개로 한 입력 값들을 통해 철골 하나 하나의 객체를 만들어 초기 도면에 추가함으로써 철골 배치도를 완성할 수 있도록 한다. 이 결과 그래픽 내용과 일반 데이터가 도면 자료에 저장된다.

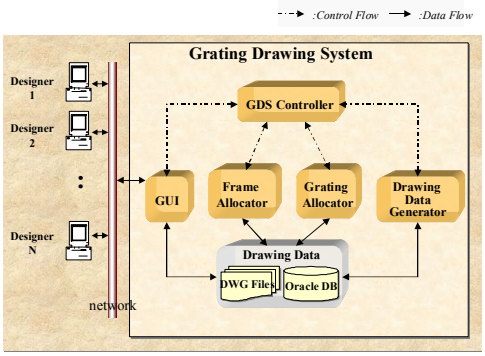


그림 3. GDS 전체 구성도
Fig. 3. Composition Diagram of GDS

*그레이팅배치기*는 그레이팅 철골 배치가 된 설계도면 내의 특정 구획(boundary) 내부에서 제작사양, 오픈(open-구획 내에서 그레이팅의 설치되지 않는 영역) 및 기타 그레이팅 배치에 영향을 주는 조건들을 고려하여 그레이팅 단품들을 적절하게 자동으로 배치해주는 모듈로서, 주요 기능에는 배치조정, 사각형 배치, 원형배치, 단품번호생성 등이 있다. 그레이팅 배치기는 배치 조건이 되는 정보들을 도면자료와 GUI로부터 전달 받아 그레이팅을 배치하며, 생성된 그레이팅 정보들을 도면자료에 저장한다.

*도면자료생성기*는 배치기를 통해 생성된 그레이팅들에 대한 통계적인 정보 및 상세한 도면 정보를 생성하는 모듈이다. 생성되는 정보는 BM-List, 단품도, 검사도이다. *BM-List*는 배치된 그레이팅 객체들에 대한 통계적인 정보 리스트가 그래픽 형태로 보여지는 것을 말하고, *단품도*는 개별적인 그레이팅에 대한 상세 정보를 알아보기 쉽도록 하여 제작 과정에서 참고할 수 있도록 한 도면이며, *검사도*는 공정과정을 통해 생산된 그레이팅 제품을 검사하기 위한 항목들이 표시된 상세도면이다.

*도면자료(Drawing Data)*는 도면에 관련된 모든 데이터를 체계적으로 저장 및 관리하기 위한 모듈로서, 두 가지의 자료 형태로 나뉜다. 하나는 AutoCAD에서 제공하거나 확장하여 재 정의한 그래픽 정보를 저장한 것이고, 다른 하나는 Oracle DBMS를 이용한 일반(non-graphical) 정보를 저장한 것이다.

각 모듈간의 통신(그림에서 화살표)은 도면관련 데이터를 공유하는 방식과 제어메시지를 교환하는 방식으로 이루어지는데, 전자의 경우를 위해 도면을 그리며 생성된 정보들을 관리하는 데이터베이스 모듈이 있고, 후자의 경우를 위해 제어 메시지를 발생시키며 전체적인 시스템의 흐름을 만들어 내는 *GDS 제어기(Controller)* 모듈이 있다.

3.3 컴포넌트

GDS에서 도면자료와 제어기를 제외한 모듈들은 모두 모듈 내에 세 가지 유형의 컴포넌트 (모듈 내의 지역 controller, 실질적인 기능 처리를 위한 computer, 데이터 접근과 관련한 container)로 구성된다. 그림 4는 그레이팅배치기 모듈에 대한 컴포넌트들의 구성 예를 보인 것인데, 다른 세 모듈도 이와 거의 유사한 요소들로 구성된다.

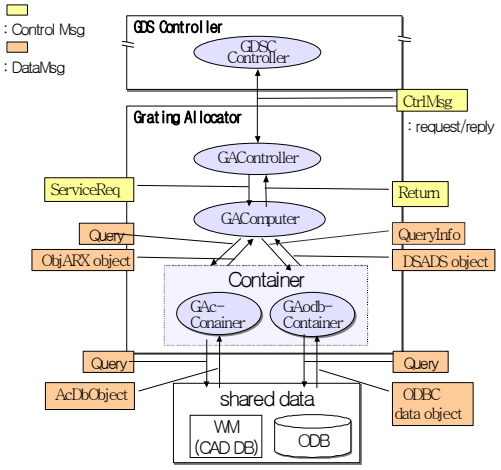


그림 4. 그레이팅배치기 모듈의 설계
Fig. 4. Components of Grainting Allocator Module

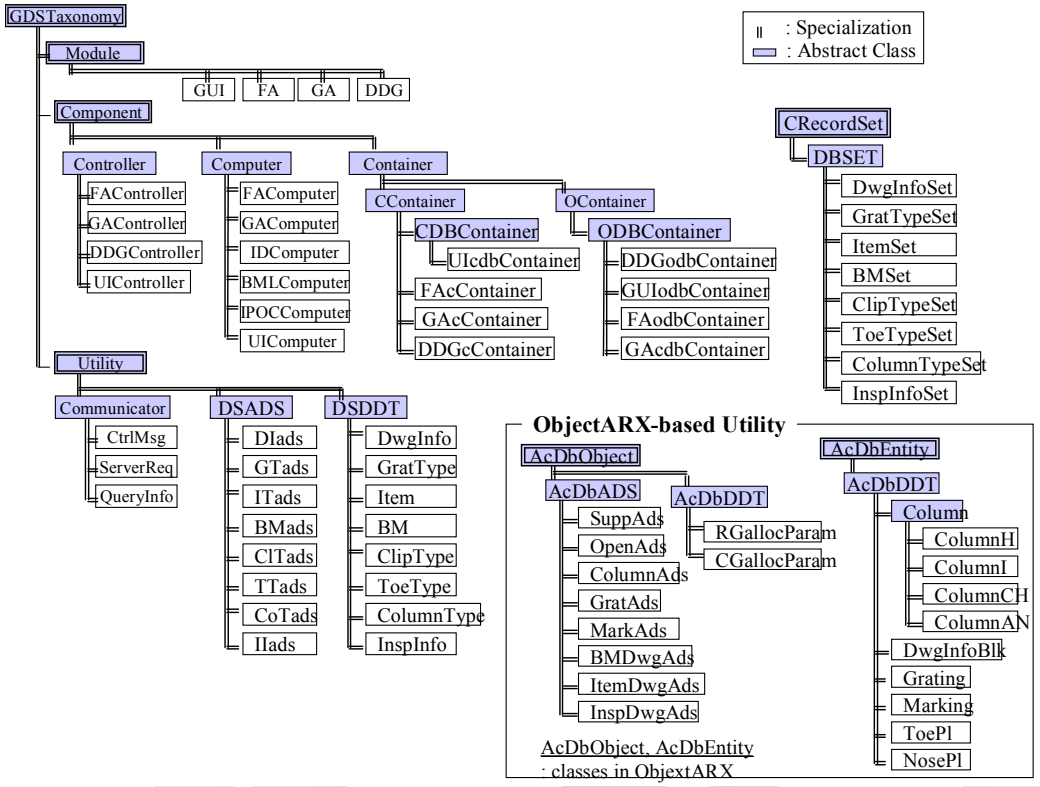


그림 5. 클래스 계층도
Fig. 5. Class Hierarchy

*Controller*는 다른 모듈들과의 연결을 목적으로 한 구성요소로서 특정 서비스를 요구하거나 서비스 요청을 받기 위한 메시지 생성, 분석, 전달 등의 기능을 수행한다. 모듈 단위의 서비스 요구를 받으면 해당 메시지를 'computer' 구성요소에게 전달하여 서비스 처리 할 수 있도록 한다.

*Computer*는 모듈의 실질적 프로세서 역할을 하는 구성요소이다. 주로 controller 에게 특정 기능 처리 요구를 받고, 그 일을 수행하기 위해 필요한 데이터들을 container에게 요청하게 된다.

*Container*는 해당 모듈에서 데이터(ODB-Oracle DB, CDB-CAD DB)를 접근하기 위한 구성요소이다. 접근하고자 하는 데이터의 종류에 따라, CDB관련 데이터를 접근하기 위한 CDBContainer와 ODB관련 데이터를 접근하기 위한 ODBContainer가 있다. CDBContainer에는 CDB를 각 모듈의 Computer에서 메모리 상의 공유 데이터를 사용하기 위한 버퍼 및 함수가 있고, 또 ObjectARX 함수를 이용한 파일 형태의 CDB를 메모리로 올리는데 필요한 버퍼 및 함수를 포함한다. ODBContainer에는 ODB를 각 모듈의 Computer에서 사용하기 위한 형태(모든 테이블의 레코드 리스트)로 유지하기 위한 버퍼 및 함수가 있고, ODBC API를 이용하여 ODB를 가져오기 위한 버퍼가 정의되어 있다. 여기에서는 3.4절에서 소개될 DBSET, DSDDT의 인스턴스들을 다루며, ODB를 Computer에서 사용할 수 있는 형태로의 변환 함수를 갖는다.

3.4 클래스 구조

그림 5는 GDS를 구성하기 위해 필요한 모듈, 컴포넌트 외에 데이터 접근 및 여러 구성 요소간 통신을 위한 유틸리티(utility) 객체들을 정의하기 위한 클래스 구조를 나타낸 것이다. 모듈과 컴포넌트 관련 클래스는 앞

절에서 언급되었던 내용들이 그대로 반영되어 있다.

유틸리티는 컴포넌트 및 모듈(컴포넌트로 구성)을 연결하기 위한 통신관련 클래스와 DB 데이터 버퍼 및 접근 방법을 제공하기위한 클래스가 정의돼 있는 부분이다. *Communicator*는 다른 모듈 및 모듈내의 컴포넌트 간 통신을 위한 방법을 제공하기 위한 클래스들이고, 관련 클래스들은 세 종류의 메시지를 표현하기 위한 것이다. 첫째, 각 모듈의 controller에게 모듈 제어를 위한 CtrlMsg (제어 메시지), 둘째 모듈간 서비스 요청을 위한 ServiceReq (서비스 요청 메시지), 셋째 DBMS로의 데이터 요구 메시지인 QueryInfo (질의 정보 메시지)가 있다. 이들은 그림 4에 표현되어 있다. 또, ODB의 데이터 접근을 위한 세 종류(DSADS, DSDDT, DBSET)의 클래스가 정의되어 있다. *DSADS(Domain Specific Abstract Data Type)*는 각 모듈의 computer에서 사용 할 ODB 테이블별 레코드 리스트 접근을 위해, *DSDDT(Domain Specific Defined Data Type)*는 각 모듈의 computer에서 사용 할 ODB 테이블별 하나의 레코드 접근을 위해, *DBSET(Database record SET)*은 ODBC API를 이용하여 DB의 데이터를 가져오기 테이블 단위의 클래스이다. CDB 데이터 접근도 ODB 접근에서와 같이 DB접근 관련 클래스 종류가 있는데, 이는 ODB 접근과 구별하기 위해 클래스 이름에 접두어 'AcDb'를 붙여 정의하였다.

4. GDS 구조 및 결과

3장에서 설계한 내용을 바탕으로 한 GDS 시스템의 구조와 실행 예를 통해 설계내용을 확인 할 수 있다.

4.1 GDS 구조

그림 6은 그림 5에 정의된 클래스들의 인스턴스 객체들로 조합된 GDS 시스템의 구조이다. 최상위에 GDS 시스템이 있고, 이를 구성하는 모듈들이 있으며, 각 모듈 안에는 다양한 컴포넌트가 있다.

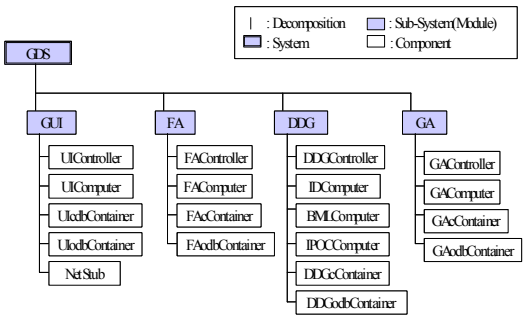


그림 6. GDS 구조
Fig. 6. GDS Structure

4.2 실행 예

이번 절에서는 앞 절의 구조로 완성된 GDS의 실행 예를 통해 각 모듈별 기능을 확인하고자 한다.

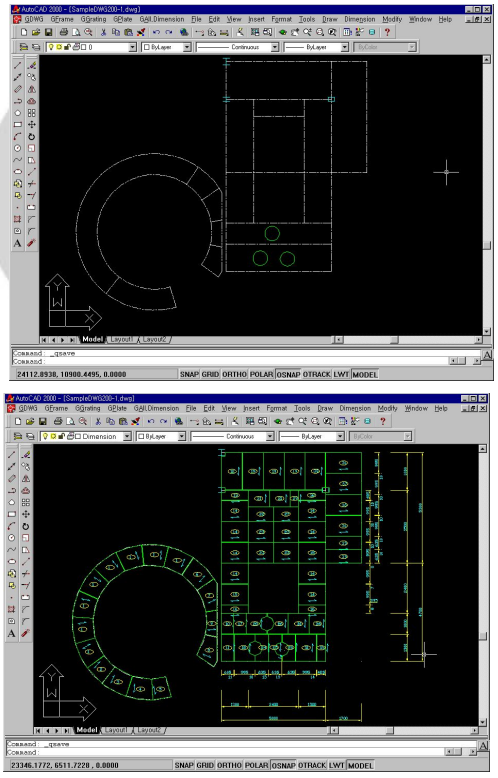
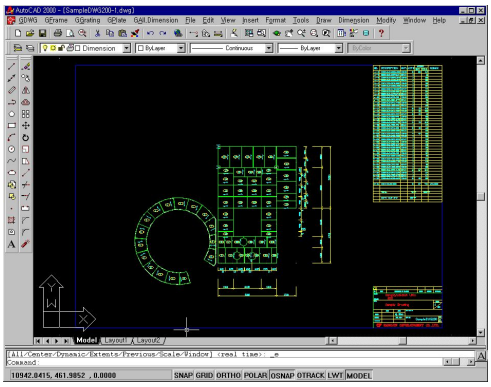


그림 7. 전체 도면: 철골 배치(위)와 그레이팅 배치(아래)
Fig. 7. Plan Drawing: Frame Allocation (above) and Grating Allocation (below)

그림 7은 자동배치의 전과 후의 도면을 나타낸 것이다. 위쪽이 철골만 배치되어 있는 상태이고, 이를 기준으로 구획별로 특정 조건을 주어 그레이팅배치기에 의해 수십 개의 그레이팅 품목들이 배치된 상태의 도면이 아래쪽 그림이다. 또, 그림의 아래쪽 화면은 자동배치 이후 품목번호를 매기는 기능까지 수행한 상태이다.

그림 8과 그림 9는 그림 7에서 그레이팅 배치된 후의 도면 내용을 기준으로 도면자료 생성기 모듈을 통해 만들어진 그래픽 자료들이다.

그림 8은 그레이팅 품목들을 분석하여 통계적인 정보인 BM-List를 그래픽적으로 생성하여 전체도면의 우측 상단에 자동으로 붙여 넣은 화면(그림의 위쪽)과 이를 확대시킨 내용(그림의 아래쪽)을 보인 것이다. 이는 해당 도면대로 그레이팅을 설치하기 위해 필요한 비용 계산에도 사용된다.



NO.	DESCRIPTION	MAT'L	QTY'S	WEIGHT UNIT	WEIGHT TOTAL	REMARK
n--1	SI25x5x3x1282x1021	SS400	8	52	419	
n--2	SI25x5x3x1035x1007	SS400	1		41	
n--3	SI25x5x3x1295x1041	SS400	1		53	
n--4	SI25x5x3x1285x1041	SS400	1		53	
n--5	SI25x5x3x899x1020	SS400	1		36	
n--6	SI25x5x3x1385x1028	SS400	1		57	
n--7	SI25x5x3x1443x1003	SS400	1		57	
n--8	SI25x5x3x1361x1007	SS400	1		54	
n--9	SI25x5x3x1278x540	SS400	1		27	
n--10	SI25x5x3x575x990	SS400	1		22	
n--11	SI25x5x3x605x1295	SS400	1		31	
n--12	SI25x5x3x994x1780	SS400	1		70	
n--13	SI25x5x3x485x1300	SS400	1		25	
n--14	SI25x5x3x995x1300	SS400	8	51	413	
n--15	SI25x5x3x994x1300	SS400	2	51	103	

그림 8. BM-List(아래)가 추가된 plan도면(위)
Fig. 8. Plan Drawing (above) and its' BM-List (below)

그림 9는 개별 그레이팅들에 대해 자동으로 생성된 단품도면와 검사도면의 예를 보인 것이다. 위쪽의 단품도면은 그레이팅 생성공정에서 실제로 그레이팅 제작에 필요한 상세 정보를 표시한 것으로서, 단품 아홉 개가 하나의 도면 단위 형태로 만들어지며 그레이팅 개수에 따라 여러 개의 도면이 생성된다. 또, 그림의 아래쪽은 그레이팅이 제작된 후 제품이 제대로 만들어졌는지를 확인하기 위한 검사 항목들이 표기되어 있는 검사 도면으로, 단품 여섯 개가 하나의 도면 단위 형태로 만들어지며 이 또한 그레이팅 개수에 따라 여러 개의 도면으로 생성된다.

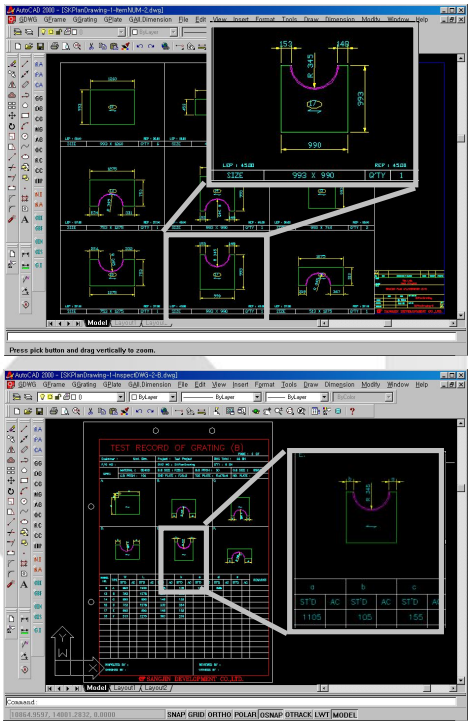


그림 9. 단품도면(위)와 검사도면(아래)
Fig. 9. Item Drawing (above) and Inspect Drawing(below)

5. 결론 및 향후 과제

점차 복잡한 시스템의 개발로 인한 효율적인 소프트웨어 개발 방법론이 요구되어지고 있고, 그의 대안으로 코드 재사용의 방향을 제시해주는 방법들이 다양하게 소개되어 왔다. 본 연구는 다양한 코드 재사용과 관련한 개

념들을 재사용 단위의 측면에서 객체, 컴포넌트, 모듈, 상업용 소프트웨어로 분류하고, 이러한 개념들이 실제 시스템을 개발하는데 어떻게 반영 할 수 있는지를 그레이팅 자동 설계 시스템(GDS) 개발을 예로 소개하였다. GDS는 여러(그래픽, 비그래픽) 유형의 데이터가 다양한 과정을 통해 가공되는 복잡한 시스템으로서, 이는 AutoCAD를 기반으로 하고 시스템 구성에서 필요한 요소들을 여러 단계로 나누어 정의함으로써 다양한 코드의 재사용적인 방법들을 반영하여 구현되었다. 그러나 본 논문에서는 그레이팅 설계 자동화라는 특정 도메인 내에서의 코드 재사용 방법을 적용해 본 것이며, 이러한 개발 접근법이 좀 더 일반성을 갖고 다양한 도메인에 쉽게 적용될 수 있도록 하는 것이 남겨진 과제이다.

참고문헌

[1] Pressman, R. S., Software Engineering A Practitioner's Approach, 4th edn, McGraw-Hill, 1997

[2] Hall, L., Hung, C. K., Hwang, C.,Oyake, A., Yin, S. J., "COTS-based OO-Component Approach for Software Inter-operability and Reuse, and reuse (software systems engineering methodology)", Aerospace Conference, IEEE Proceedings, Mar. 2001

[3] Janaki Ram, D., Sreekanth, M., "Reusable Integrated Components of Inter-related Patterns for Software Development", Proceedings of the Seventh Asia-Pacific Software Engineering, 2000

[4] 윤희환, CBD환경에서 컴포넌트의 재사용성 측정 메트릭스, 컴퓨터정보학회논문지, 10권 4호, 2005

[5] I. Zeid, CAD/CAM: Theory and Practice, McGraw-Hill, 1991

[6] Y. Cho, Construction of efficient CAD environment, 2000

[7] S.Szykman, Steven J. Fenves, Walid Keirouz, Steven B. Shooter, A foundation for interoperability in next-generation product development systems, Computer-Aided Design, 2001

[8] Zeigler, B. P., Objects and Systems, Springer-Verlag, 1997

[9] Al-Yasiri, A., Bamford, C., "DOOR-An Approach for Reusing and Retrieving Domain-Oriented Components", Real-Time Systems, Proceedings. 10th Euromicro Workshop, 1998

[10] C Szyperski, "Components vs. Objects vs. Component Objects", Proc. of OOP, München, Jan 2005

[11] Lee, S. C., Lin, J. M. & Jiau, H. C., "Constructing COTS-Based Software Components with Reusing COTS products" International Symposium on Multimedia Software Engineering, 2000

[12] Carnegie Mellon University, "COTS and Open Systems", http://www.sei.cmu.edu/str/descriptions/cots.html, 2005

[13] 이해영, 조대호, "향상된 그레이팅 배치를 위한 A*알고리즘의 적용", 한국시뮬레이션학회 춘계학술대회지, 2004

[14] 최문희, 조대호, "도면 자동 생성 시스템에서의 전문가 시스템 적용", 한국시뮬레이션학회 춘계학술대회지, 2004

[15] AcAuley, C., Programming AutoCAD 2000 Using ObjectARX, Autodesk Press, 2000

저 자 소 개

이 미 라

2005년 2월 : 성균관대학교대학원 전기전자및컴퓨터공학 박사

2005년 ~ 현재 : 목포해양대학교 해양전자통신공학부 전임강사

관심분야 : 모델링 및 시뮬레이션, 소프트웨어 개발 방법론