creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

[UCI]I804:24011-200000276202

2014年 8月

碩 士 學 位 論 文

S-BIM기반 스마트그린 구조시스템의 설계자동화 구현

朝 鮮 大 學 校 大 學 院

建 築 工 學 科

金 鎭 香

S-BIM기반 스마트그린 구조시스템의 설계자동화 구현

Implementation of Automated Design Module for Smart Green Structural System based on S-BIM

2014年 8月 25日

朝 鮮 大 學 校 大 學 院

建 築 工 學 科

金 鎭 香

dCollection @ chosun

S-BIM기반 스마트그린 구조시스템의 설계자동화 구현

指導敎授　崔　宰　赫

이 論文을 工學 碩士學位申請 論文으로 提出함.

2014年　04月

朝 鮮 大 學 校　大 學 院

建 築 工 學 科

金　鎭　香

dCollection @ chosun

金鎭香의 博士學位論文을 認准함

委員長 朝鮮大學校 敎授 조 창 근 印

委 員 朝鮮大學校 敎授 최 재 혁 印

委 員 朝鮮大學校 敎授 조 규 만 印

2014年 5月

朝 鮮 大 學 校 大 學 院

dCollection @ chosun

차 례

ABSTRACT

제1장 서론

1.1 연구 배경 및 목적 ····· 1
1.2 기존의 연구 ····· 2
1.3 연구범위 및 방법 ····· 4

제2장 BIM 및 스마트그린 구조시스템의 개요

2.1 BIM의 개요 ····· 6
2.1.1 BIM의 정의 ····· 6
2.1.2 국내외 BIM 도입 현황 ····· 7
2.1.3 BIM 프로세스 및 정보호환 ····· 11
2.2 스마트그린 구조시스템 ····· 14
2.2.1 스마트그린 구조시스템의 개요 ····· 14
2.1.2 스마트그린 구조시스템의 핵심 기술 ····· 16
2.3 OpenAPI ····· 19
2.3.1 .NET FRAMEWORK ····· 20
2.3.2 TEKLA OpenAPI ····· 20
2.4 소결 ····· 23

제3장 OpenAPI를 통한 스마트그린 구조시스템 설계자동화 모듈 개발

3.1 Tekla OpenAPI를 통한 설계자동화 모듈 개발 환경 ····· 24
3.1.1 Tekla OpenAPI의 클래스와 객체 ····· 24

dCollection @ chosun

3.1.2 개발언어인 C# 프로그램의 체계 ····· 25
3.2 에너지 흡수장치 설계자동화 모듈개발 ····· 26
3.2.1 전단형 마찰댐퍼의 설계자동화 모듈 개발 ····· 26
3.2.1.1 전단형 마찰댐퍼의 정의 ····· 26
3.2.1.2 전단형 마찰댐퍼의 Visual C# 입력 데이터 ····· 28
3.2.1.3 전단형 마찰댐퍼의 APPLICATION ····· 37
3.2.2 강재댐퍼 설계자동화 모듈 개발 ····· 37
3.2.2.1 강재댐퍼의 정의 ····· 37
3.2.2.2 강재댐퍼의 Visual C# 입력데이터 ····· 40
3.2.2.3 강재댐퍼의 APPLICATION ····· 40
3.3 수평저항기구 설계자동화 모듈개발 ····· 41
3.3.1 원형브레이스 설계자동화 모듈 개발 ····· 41
3.3.1.1 원형 브레이스의 정의 ····· 41
3.3.1.2 원형 브레이스의 Visual C# 입력 데이터 ····· 43
3.3.1.3 원형 브레이스의 APPLICATION ····· 43
3.4 신개념 구조 접합 설계자동화 모듈 개발 ····· 44
3.4.1 Flexible한 보-기둥 접합부 설계자동화 모듈 개발 ····· 44
3.4.1.1 Flexible한 보-기둥 접합부의 정의 ····· 44
3.4.1.2 Flexible한 보-기둥 접합부의 Visual C# 입력 데이터 ····· 45
3.4.1.3 Flexible한 보-기둥 접합부의 APPLICATION ····· 45
3.4.2 조립·해체가 자유로운 주각부 설계자동화 모듈 개발 ····· 46
3.4.2.1 조립·해체가 자유로운 주각부의 정의 ····· 46
3.4.2.2 조립·해체가 자유로운 주각부의 Visual C# 입력 데이터 ····· 48
3.4.2.3 조립·해체가 자유로운 주각부의 APPLICATION ····· 48
3.5 소결 ····· 49

제4장 BIM 객체정보 모델 데이터 활용

4.1 BIM 데이터 활용을 위한 스마트그린 구조시스템 3차원 건축 정보 모델링 ····· 50
4.2 BIM 기반 스마트그린 구조시스템의 부재 정보 관리 ····· 51

dCollection @ chosun

4.2.1 BIM기반 스마트그린 구조시스템의 물량 산출 ····· 51
4.2.2 기존 구조시스템과 스마트그린 구조시스템의 물량 비교 ····· 53
4.3 BIM 기반 스마트그린 구조시스템의 도면 출력 ····· 56
4.3.1 시스템상의 도면 출력 구성 ····· 56
4.3.2 스마트그린 구조시스템 핵심 기술요소 디테일 도면 출력 ····· 57
4.3.2.1 전체 구조물의 도면 출력 ····· 57
4.3.2.2 에너지 흡수장치의 도면 출력 ····· 58
4.4 소결 ····· 59

제5장 결론 ····· 60

참고문헌 ····· 61

부록

1. 면외좌굴구속 강재댐퍼의 Visual C# 입력데이터 ····· 64
2. 원형강관댐퍼의 Visual C# 입력데이터 ····· 69
3. 원형 브레이스의 Visual C# 입력데이터 ····· 75
4. Flexible한 보-기둥 접합부의 Visual C# 입력데이터 ····· 80
5. 조립·해체가 자유로운 주각부의 Visual C# 입력데이터 ····· 86
6. 부재 요소별 물량 데이터 ····· 90
7. 자동 도면 생성 ····· 92

dCollection @ chosun

표 차 례

표 1.1 연구범위 및 방법 ····· 5

표 2.1 BIM의 정의 ····· 7
표 2.2 해외 주요 국가의 BIM 도입 및 적용 현황 ····· 10

표 4.1 기존 구조시스템 및 스마트그린 구조시스템 물량 비교 ····· 54

dCollection @ chosun

그 림 차 례

그림 2.1 BIM의 개념도 ····· 6
그림 2.2 BIM 건설시장 ····· 8
그림 2.3 조달청 보도자료 ····· 8
그림 2.4 국토교통부 보도자료 ····· 9
그림 2.5 BIM을 이용한 상호운용성 ····· 12
그림 2.6 한국의 온실가스 배출 의무 감축량 ····· 14
그림 2.7 스마트그린 구조시스템 개념 ····· 15
그림 2.8 DFD(Design For Disassembly)의 개념 ····· 15
그림 2.9 분리 MECHANISM ····· 16
그림 2.10 스마트그린 구조시스템의 구성요소 ····· 17
그림 2.11 Flexible한 보-기둥 접합부 ····· 18
그림 2.12 조립·해체가 자유로운 주각부 접합부 ····· 18
그림 2.13 원형브레이스 ····· 19
그림 2.14 전단형 마찰댐퍼 ····· 19
그림 2.15 .NET 프레임워크 구조도 ····· 20
그림 2.16 Tekla OpenAPI ····· 21

그림 3.1 Tekla OpenAPI를 이용한 객체 생성 구조도 ····· 24
그림 3.2 C# 프로그램의 실행 모델 ····· 25
그림 3.3 C# 프로그래밍의 개발 흐름도 ····· 26
그림 3.4 전단형 마찰댐퍼의 구성요소 ····· 27
그림 3.5 전단형 마찰댐퍼의 개발 프로세스 ····· 27
그림 3.6 좌표계 설정 변경 입력데이터 ····· 28
그림 3.7 상부 T형 부재 입력데이터 ····· 29
그림 3.8 하부 T형 부재 입력데이터 ····· 29
그림 3.9 상부 T형 부재 슬롯 입력데이터 ····· 30
그림 3.10 내부 마찰재 입력데이터 ····· 31

dCollection @ chosun

그림 3.11 커버 플레이트 입력데이터 ····· 32
그림 3.12 상・하부 볼트 입력데이터 ····· 33
그림 3.13 중앙부 인장 볼트 입력데이터 ····· 34
그림 3.14 부동 소수점 형식변수 double의 데이터 ····· 35
그림 3.15 원형강봉의 입력데이터 ····· 36
그림 3.16 전단형 마찰댐퍼의 APPLICATION ····· 37
그림 3.17 면외좌굴 구속 강재댐퍼의 구성요소 ····· 38
그림 3.18 원형강관댐퍼의 구성요소 ····· 39
그림 3.19 강재댐퍼의 개발 프로세스 ····· 39
그림 3.20 면외좌굴구속 강재댐퍼의 APPLICATION ····· 40
그림 3.21 원형강관댐퍼의 APPLICATION ····· 41
그림 3.22 원형 브레이스의 구성요소 ····· 42
그림 3.23 원형 브레이스의 개발 프로세스 ····· 42
그림 3.24 원형 브레이스의 APPLICATION ····· 43
그림 3.25 Flexible한 보-기둥 접합부의 구성요소 ····· 44
그림 3.26 Flexible한 보-기둥 접합부의 개발 프로세스 ····· 45
그림 3.27 Flexible한 보-기둥 접합부의 APPLICATION ····· 46
그림 3.28 조립・해체가 자유로운 주각부의 구성요소 ····· 47
그림 3.29 조립・해체가 자유로운 주각부의 개발 프로세스 ····· 47
그림 3.30 조립・해체가 자유로운 주각부의 APPLICATION ····· 48

그림 4.1 스마트그린 구조시스템 3차원 건축 정보 모델링 ····· 50
그림 4.2 스마트그린 구조시스템 전체 물량 산출표 ····· 52
그림 4.3 스마트그린 구조시스템 보 물량 산출표 ····· 53
그림 4.4 세대 단위별 철골 자재 물량 비교 ····· 55
그림 4.5 모델하우스 강재의 누적 사용량 비교 ····· 55
그림 4.6 도면 Layout 구성 ····· 57
그림 4.7 전체 구조물의 도면 ····· 58
그림 4.8 전단형 마찰댐퍼 도면 ····· 59

dCollection @ chosun

Abstract

Implementation of Automated Design Module for Smart Green Structural System based on S-BIM

Kim, Jin Hyang
Advisor : Prof. Choi, Jae hyouk,
Department of Architectural Engineering,
Graduate School of Chosun University

A conventional 2D-based design approach has a limitation specially in a large scale construction projects, during formal and informal design changes. To come up with this limitations, a 3D-BIM based modeling has been implemented. In BIM modeling the limitations such as design and contractibility review, automatic quantity calculation, structural health monitoring in architectural engineering construction and facility management including large scale construction has solved. Thus, BIM (building information modelling) has become an integral part of construction projects. BIM-based software is a commercially available in varies type. In a large scale projects, BIM software updates all features that are modified in the design. However, the library of this technology do not provide a special structure. But it has an option to add new features developed by a group of professionals. In this paper, a special shape automated design module of smart green structure was developed using BIM modeling techniques. In

dCollection @ chosun

this structure, a new library is developed (by adding special structures) using .NET environment and optimized in C# using Tekla OpenAIP. In addition, the quantities of the developed smart green structure was calculated automatically. The automatically calculated quantities was compared with the quantities of existing structure.

dCollection @ chosun

제1장 서 론

1.1 연구 배경 및 목적

최근 건설프로젝트의 대형화, 다양화, 형태의 비정형화 등의 변화에 의해 기존 2D기반의 정보 교환 방식으로는 디자인, 시공성 검토(부재간섭 등), 건전성 진단 및 유지관리, 자동물량산출 등 AEC/FM(Architecture, Engineering, Construction and Facility Management) 산업 전반에서 생산되는 방대한 건축 정보를 통합 관리하기에는 뚜렷한 한계가 있다. 이에 대응하기 위하여 Building Information Modeling(이하, BIM)은 필수적인 요소가 되었다. 건설 생산성의 향상을 위해 파라메트릭 기술을 기반으로 한 BIM 및 IPD(Integrated Project Delivery)가 도입되고 실무 적용 연구가 이루어지고 있다. 또한 BIM 기술은 건설 프로세스 뿐만 아니라 이를 지원하는 기술적인 부분과 정책적인 부분이 상호 협력하여 발전을 거듭하고 있다. 특히, Structural BIM(이하, S-BIM)은 파라메트릭기법(Parametric Method)을 기반으로 하여 구조해석 및 설계, 골조 모델링, 도면생성, 물량산출 등의 업무를 통합 환경에서 원활하고 신속하게 수행 할 수 있는 유연한 환경을 제공한다.[1~5,10~14,22]

BIM 기반의 소프트웨어는 시중에 여러 가지 종류가 존재한다. 대형 소프트웨어 벤더사의 BIM 프로그램의 경우, 정기적으로 버전을 업데이트 하여 새로운 기능을 추가하는 것은 물론, 전문 개발자 그룹(Third Parties)에 의해 여러 기능의 플러그인이 상업적으로 제공된다. S-BIM을 적용한 철골 구조분야에서 구조물을 모델링 할 때, 접합 디테일에 대하여 기존 라이브러리의 조합으로 보, 기둥, 슬래브, 벽체, 계단, 지붕 시스템이 쉽게 표현이 가능하다. 하지만, 신기술, 신공법 및 특수구조형상은 기존 라이브러리에는 제공되지 않는 경우, BIM 벤더사가 라이브러리를 제공하지 않는 한 이를 모델링하기는 매우 제한적이다. 특히, AEC/FM 산업분야에서 현재 사용되는 상용 BIM 어플리케이션에서 기본적으로 제공하는 기능으로는 이러한 요구사항을 충족시키기에 한계가 있다. 따라서 새로운 특수구조체 설계와 실제 건축물에 적용을 위한 세부적인 프로세스에는 여러 가지 특화된 문제에 대한 해결 방안이 요구된다.[17-33]

따라서, 본 논문에서는 제한적인 라이브러리의 한계를 극복하기 위해 설계단계부터 해체를 고려한 신개념 시스템인 스마트그린 구조시스템을 대상으로 조립·해체가 자유로운 Flexible한 접합부, 수평저항장치 및 에너지 흡수장치 등 주요 구조요소를 .NET

dCollection @ chosun

프레임워크에 최적화되어 있는 C#을 사용하여 OpenAPI를 통해 개발하고자 한다. 사용자가 구현하고 싶은 형상의 개발에 제한이 없는 OpenAPI 환경에서 스마트그린 구조 시스템의 특수구조 디테일 모듈을 다차원 설계자동화 모듈로 구현하는 방법을 제안함으로써 기존 라이브러리 부재에 따른 한계를 극복하고, 나아가 새로운 구조 요소의 설계 프로세스를 개발하고자 한다. 또한, 구조 BIM 모델링을 완성한 뒤 부가 작업 없이 자동 물량 산출 및 디테일 도면을 출력할 수 있고, 모든 정보를 담고 있는 객체 모델의 정량적인 물량 데이터를 이용하여 강재의 사용량을 분석할 수 있다.

1.2 기존의 연구

현재 국내에서는 BIM기반 3차원 구조설계를 위한 기술 지원 확대가 활발히 진행되고 있으며, 다양한 프로그램의 기능 조합과 분야별 협업을 통해 BIM을 더욱 효율적으로 사용하기 위하여 연구가 활발히 진행되고 있다.

Razvi(2010)[12]는 BIM 기술은 건설 프로세스뿐만 아니라 이를 지원하는 기술적인 부분과 정책적인 부분이 상호 협력하여 발전을 거듭하고 있다고 밝혔다.

최철호(2008)[17]는 초기의 BIM은 2D 도면을 3D로 재모델링하여 시공성 검토, 부재 간섭체크를 수행하는 등 그 적용성이 제한적이었으나, 최근 AEC/FM 산업계 전반에 걸쳐 생산되는 데이터를 통합 저장하는 매개체 역할로 그 적용 범위가 확대되고 있는 추세라고 정의했다.

신태송(2008)[18]은 건축구조설계에서 BIM의 도입은 단위 업무(구조설계, 상세설계, 도면작성)의 진행에 큰 변화를 가져왔고, IT 기술의 발달로 구조설계 단계에서는 BIM이 도입되기 이전부터 3차원 구조해석모델을 이용한 전산해석을 수행함으로써 일정 수준 이상의 설계 자동화가 이루어져 왔으나, 후속 업무인 상세설계 및 도면작성 업무는 여전히 2D 기반의 CAD를 이용한 수작업에 의해 모든 업무가 진행되어 업무의 효율성이 떨어지는 문제점을 안고 있다고 분석하였다. 또한 API를 이용하여 건축 구조설계 업무를 지원하는 구조기본모델(Revit Structures), 구조해석모델(MIDAS/GEN), 구조상세모델(Tekla Structures) 사이의 모델링 데이터를 양방향 호환 가능하도록 하는 프로그램으로써 구조형식(Steel, RC등)에 제한받지 않고 적용 가능한 범용적인 인터페이스 프로그램을 개발하였다.

dCollection @ chosun

이병해(2010)[21]는 BIM을 지원하는 소프트웨어 벤더들은 BIM 프로세스의 원활한 수행을 위해 최신의 기술들을 적용한 업데이트를 지속적으로 지원하고 있고, 블랙박스(BlackBox)화 되어 있던 자사의 원천 기술들을 오픈소스(Open Source)화 하여 지원함으로써 그 기술의 개발이 더욱 용이하도록 하였다. 이를 통해 사용자는 단순히 벤더에서 제공하는 기술들을 사용하는데 그치지 않고, 필요로 하는 기능이 적용 가능한 다양한 응용 프로그램을 개발할 수 있도록 함으로써 소프트웨어를 좀 더 유연하게 활용할 수 있고, BIM 적용성 또한 높일 수 있도록 하였다.

엄진업(2010)[24]은 현재 국내에서 통용되고 있는 철골 구조물을 대상으로 구조해석 모델과 BIM 기반 3D 건축정보를 담고 있는 기본상세 모델의 상호 연동이 가능한 인터페이스 모듈을 개발하였고, 또한 구조상세설계(Structural Detail Design) 자동화 시스템을 구축하기 위해 닷넷 프레임워크(.NET Framework) 개발환경인 Visual C#을 사용하여 접합부 자동 생성 모듈(Auto Connection Module)을 개발하였다.

엄진업(2011)[24-26]은 BIM 기반의 건축구조설계 업무 프로세스에서 활용 가능한 철골 구조물의 구조해석모델과 기본상세모델의 상호운용성을 기반으로 정보 교환을 수행할 수 있는 인터페이스 모듈을 개발하였다. 정보 교환 방식 중 직접 연결 방식을 적용하였으며, BIM 벤더에서 제공하는 OpenAPI를 통한 .NET Framework 개발환경을 이용하였다.

최중식(2011)[28]은 공통적으로 적용될 수 있는 업무를 대상으로 개방형 BIM 환경에서의 건축설계 품질향상을 위한 품질관리 기준을 개발하고, 요소 기술 및 시스템 개발로 향후 개방형 BIM을 통한 품질관리의 효율적 적용 방안을 제시하였다.

고아라(2013)[31]는 구조해석 및 설계 어플리케이션으로 구성된 각 어플리케이션간의 정보호환체계를 구축하고, 접합부 설계를 위한 외부모듈(SASS-EM: Steel connection Automaatic design System based on S-BIM-External Module)을 Tekla OpenAPI를 이용하여 Visual C# 환경에서 개발하였다.

김태형(2013)[32]은 강구조물의 실시설계 단계에서 브레이스 접합부의 구조설계 및 BIM 모델링의 효율성을 제고하기 위해, BIM 설계 도구와 연동하여 모델 데이터로부터 구조설계를 수행하기 위한 입·출력 데이터를 자동으로 생성하였다. 또한 구조설계 결과로부터 브레이스 접합부의 BIM 모델을 자동으로 생성할 수 있는 인터페이스 모듈을 개발하였다.

dCollection @ chosun

1.3 연구범위 및 방법

본 논문에서는 BIM 프로그램에서 제공하는 기본 라이브러리로는 모델링이 불가능한 특수구조형상의 설계 자동화 모듈을 개발하고, 이 설계 자동화 모듈을 이용하여 BIM 기반 스마트그린 구조시스템의 전체 모델링을 완성하고자 한다.

이를 위해, BIM의 개요 및 본 연구에서 적용한 정보 교환 방법 중 하나인 직접 연결 방식으로 적용되는 오픈소스(Open Source)에 대한 개념적인 이해와 더불어 Tekla Open API의 이론적 고찰을 진행하고, 스마트그린 구조시스템에 대한 전반적인 개념 및 세부 특성에 관하여 정의한다.

다음으로, Tekla Structures에서 제공하지 않는 구조상세접합 및 에너지 흡수장치 디테일 모델의 신규라이브러리를 개발하기 위해 프로그램 개발의 .NET환경에 최적화 되어있는 C#을 이용한여, 특수구조형상인 신개념 구조상세접합 (조립・해체가 자유로운 주각부, Flexible한 보-기둥 접합부), 수평저항장치(원형 브레이스) 및 에너지 흡수장치(전단형 마찰댐퍼, 면외좌굴구속 강재댐퍼, 원형강관댐퍼)의 설계자동화 모듈을 개발하고자 한다.

각 디테일 컴포넌트 DB와 신규 라이브러리 형상의 오브젝트를 효과적으로 구축하기 위해 구성요소별로 파라미터 분석을 선행한 후 개발된 에너지 흡수장치, 수평저항장치 및 신개념 구조상세접합 설계자동화 모듈을 적용함으로써 4층 규모의 건축물에 스마트그린 구조시스템 구조물을 완성한다.

BIM 기반으로 완성된 스마트그린 구조시스템에 대하여 자동물량산출 및 도면을 출력하고, 정량적인 물량 산출을 토대로 기존 구조시스템과 스마트그린 구조시스템의 물량을 비교・평가한다.

dCollection @ chosun

표 1.1 연구 방법

이론적 고찰

- BIM의 상호운용성(BIM Interoperability)
- 오픈소스(Open Source)

↓

대상 건물 선정(다차원 디자인 모듈)

- 모델하우스와 같은 기간지정 건축물 선정
- 신개념 접합, 수평저항장치, 에너지 흡수장치

↓

S-BIM기반의 특수 구조 형상 상세설계 프로세스 분석

- 스마트그린 구조시스템 디테일 형상 구조도 분석
- 스마트그린 구조시스템 디테일 형상 구성요소 분류 체계

↓

특수 구조 형상 상세설계 자동화 모듈 구축

인터페이스 모듈 개발	접합 모델링 자동화 모듈 개발

↓

개발 설계 모듈 대상건물에 장착

- UI(User Interface)를 통한 대상건축물 개발모듈 장착
- 프로세스 비교 분석

↓

도면 출력 및 정량적인 물량 산출

↓

기존 구조시스템과 스마트그린 구조시스템의 비교·분석

↓

결론 및 향후 연구방향 도출

dCollection @ chosun

제2장 BIM 및 스마트그린 구조시스템의 개요

2.1 BIM의 개요

2.1.1 BIM의 정의

그림 2.1과 같이 BIM은 Building Information Modeling의 약자로 초기 개념설계에서 유지관리 단계에까지 건물(프로젝트)의 전 수명주기 동안 다양한 분야에서 적용되는 모든 정보를 생산하고 관리하는 기술이라 할 수 있다. 따라서 BIM 기술을 적용할 경우 건축, 건축분야에서 생산되는 다양한 정보들을 좀 더 효율적으로 활용할 수 있으며, 다양한 장점들이 구체화되고 있는바 국제적 뿐만 아니라 국내에서도 다양한 접근을 통해 BIM 적용에 힘쓰고 있는 실정이다.[39]

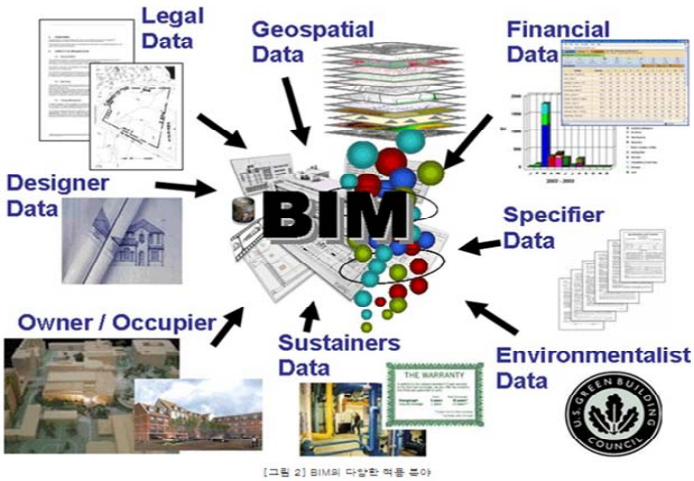


그림 2.1 BIM의 개념도

BIM은 대상(객체)의 의미로 쓸 수 있으며, Building을 건축물로만 해석하는 것이 아니라 '짓는 정보 모델'로 해석할 수 있다. BIM은 건축 분야뿐만 아니라 모든 건조환경을 대상으로 하기 때문에 유럽과 미국에서는 토목 및 플랜트 분야에서도 BIM이라는 용어가 일반화 되어있다. 국내외 대표적인 BIM의 정의를 확인해 보면, ISO/DIS 294

dCollection @ chosun

81-1 Building Information Models-Information Delivery Manual에서는 BIM을 “의사결정을 신뢰할 수 있는 기반을 형성하는 건물, 교량, 도로, 플랜트 등의 건설되는 개체의 물리적 및 기능적인 특성의 공유된 디지털 표현”이라고 정의하고 있다. 또한 2010년 국토교통부의 '건축 분야 BIM 적용 가이드’에서는 “BIM이라 함은 건축, 토목, 플랜트를 포함한 건설 전 분야에서 시설물 객체의 물리적 혹은 기능적 특성에 의하여 시설물 수명주기 동안 의사결정을 하는데 신뢰할 수 있는 근거를 제공하는 디지털 모델과 그의 작성을 위한 업무절차를 포함하여 지칭 한다”라고 규정하고 있다. 그 외에 국내외 기관 및 BIM 벤더사는 표2.1과 같이 BIM을 정의하고 있다.[1,4,5]

표 2.1 BIM의 정의

조직	정의
NIBS	좀 더 나은 가치를 실현하기 위한 비즈니스 의사결정을 알리기 위해 개방형 산업 표준을 사용하여 시설물과 그와 관련된 프로젝트. 생애주기 정보의 물리적 및 기능적인 특성의 컴퓨터적인 표현
AIA	프로젝트 정보의 데이터베이스와 연계된 모델 기반의 기술로서, 데이터베이스 기술에 대한 일반적인 의존도를 반영함
AEC Infosystems	단일한 구성 요소인 전자적 문서로 통합된 3차원-2차원 모델 기반 기술의 정보 이용, 재사용 및 교환
Graphisoft	그래픽 문서인 비그래픽 문서인 시방서, 일정 및 기타 데이터를 모두 포함하는 단일의 저장소
Bentley	데이터관리시스템의 전체 빌딩 생애주기의 모든 그래픽과 비그래픽의 모델링
Autodesk	빌딩 디자인과 문서화의 방법론은 설계와 시공에서의 빌딩 프로젝트에 대한 컴퓨터로 표현 가능한 정보의 생성, 조율된 사용, 내부적인 일관성에 의해 특성화됨

2.1.2 국내외 BIM 도입 현황

건설프로젝트가 대형화, 복잡화됨에 따라 관리 및 생산성 향상을 위한 노력들이 활발히 이루어지고 있다. 이러한 노력들 중 IT기술을 이용한 건축 정보 모델링(BIM)에

dCollection @ chosun

대한 관심은 가히 폭발적이다. 그림 2.2에서와 같이 미국의 연방조달청(GSA: General Services Administration), 공병단(US Army Corps of Engineers), 노르웨이의 공공발주처(Statsbygg), 싱가포르 건설청(Singapore BCA), 핀란드 공공발주기관(Senate Properties), 영국 등 전 세계적으로 BIM 기법을 도입한 건설프로젝트를 적극적으로 수행하고 있으며, BIM 건설 시장은 정책적으로 점차 확장되고 있다.[5]

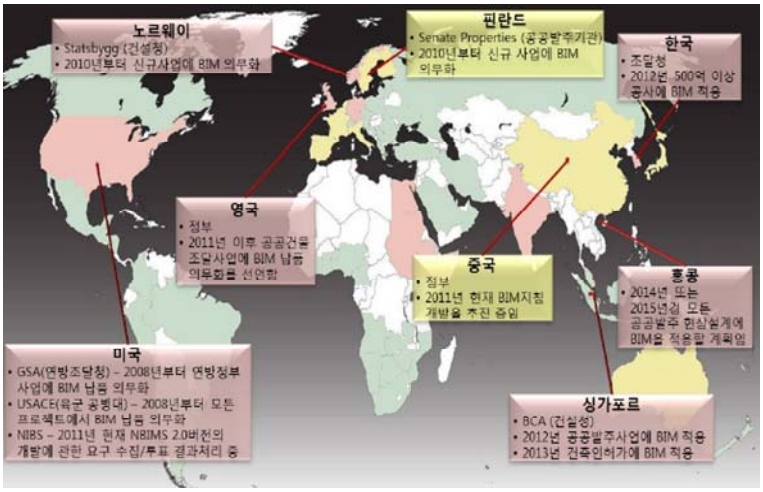


그림 2.2 BIM 건설 시장

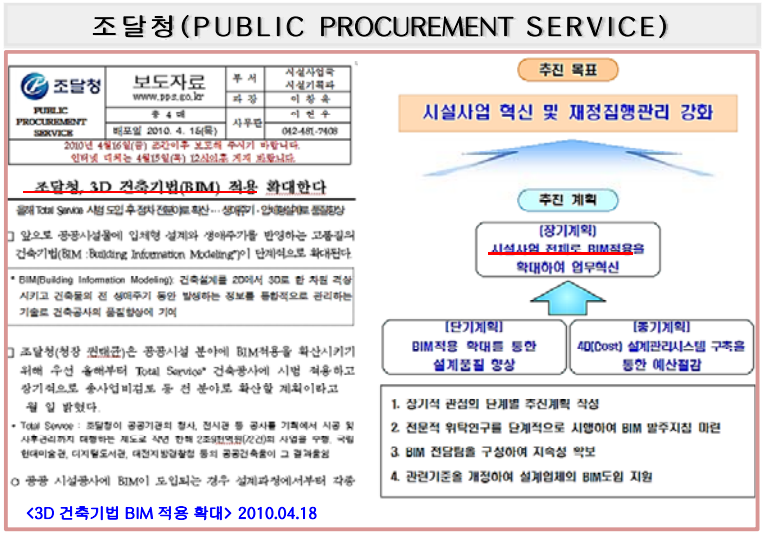


그림 2.3 조달청 보도자료

dCollection @ chosun

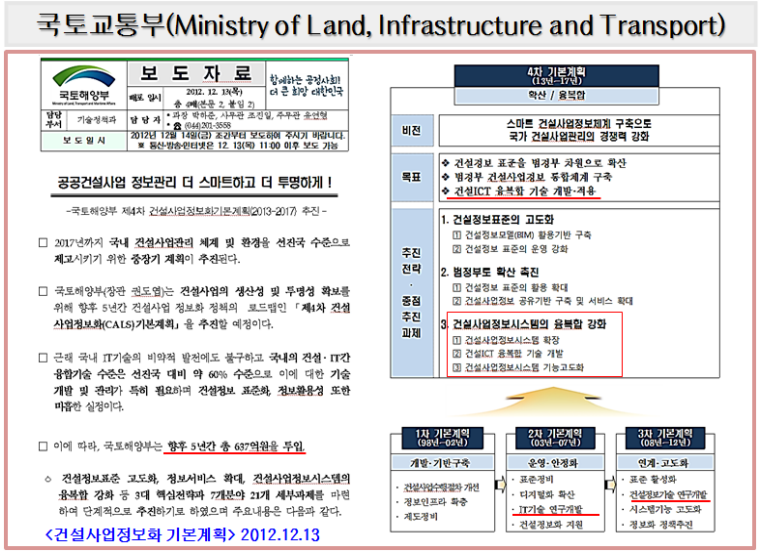


그림 2.4 국토교통부 보도자료

특히, 그림 2.3과 같이 국내의 경우 조달청에서 발주하는 500억 이상의 턴키공사(2012년)와 500억 이상의 일반공사(2013년) 건설프로젝트에 대하여 의무적으로 BIM을 도입하도록 하고 있으며, 2016년까지는 모든 일반 건설 공사에 BIM 의무 발주를 확대 적용하기로 공표함으로써 BIM의 정착은 이미 실현되었다고 볼 수 있다.[37]

또한, 그림 2.4와 같이 국토교통부(구, 국토해양부)에서는 건설과 IT융합기술에 향후 5년(2012-2017)간 637억원을 지원하기로 확정함으로써 건설 산업 분야의 국가 경쟁력을 높이기 위해 BIM의 도입은 불가피한 흐름이 되었으며, 국가 및 공공분야는 민간분야와 협력 체제를 구축하여 BIM기법의 확산에 적극 나서야 할 필요성이 대두되었다. 이렇듯 현대의 건설 산업은 관련 주체들 뿐만 아니라 이와 관련된 모든 분야에서 BIM을 적용한 프로세스의 도입 필요성이 고조되고 있으며 국내의 대형 건설 프로젝트에서는 이미 BIM을 적용한 프로젝트를 시범 적용하여 만족할 만한 성과를 올리고 있다.[38]

표 2.2에서와 같이 선진국을 필두로 해외의 주요 국가는 이미 BIM을 도입 및 적용하여 공공정책과의 연계를 통해 운영되도록 하고 있으며, BIM의 표준 포맷으로 납품을 받고 있다.[20]

dCollection @ chosun

표 2.2 해외 주요 국가의 BIM 도입 및 적용 현황

국가	주요내용	공공정책과 연계성
미국	• NBIMS을 제정함 • 연방조달청(GSA)이 개방형 BIM 포맷(BIM, IFC) 제출을 의무화 • 미육군 엔지니어링 사령부(USACE)는 로드맵을 통해 원가절감, 시간단축, 품질유지 및 개선을 위한 BIM 도입을 준비함.	• 2007년부터 연방정부 예산의 프로젝트부터 개방형 BIM 표준 포맷 제출을 의무화함
핀란드	• 2007년 10월부터 개방형 BIM 포맷의 채택과 IFC 요구(건축설계 분야: 필수, 구조 및 MEP 분야: 장려) • 공공부동산공사(Senate Properties)가 개발한 건축, 구조, MEP 등 전문 분야별 BIM 요구조건을 포함하는 'BIM Requirements 2007' 지침을 발간함	• 정부 산하의 건물설계 시공, 유지관리, 임대 등을 관장하는 공공 회사인 공공부동산공사가 개방형 BIM 표준 포맷으로 납품을 의무화함
노르웨이	• 건축 분야의 계획 심사에 개방형 BIM과 GIS 활용 중 • 건설산업에서 모든 정보 소통을 위한 표준으로 IFC를 점차 적용하고 있음(법규 기반의 디자인 체커 등) • BARBI와 LexiCon의 코디네이션을 위한 IFD 개발에 관해 STABU(네덜란드)와 함께 빌딩스마트 프로젝트의 동의서에 서명함. • 공공 발주 BIM 채택과 IFC 포맷 납품을 권고하기 위해 BIM-매뉴얼 버전 1.0을 발간함.	• 2010년부터 건설 프로젝트에 개방형 BIM을 전면적으로 채택 및 적용할 예정임. • 개방형 BIM 채택과 활용을 위해 Statsbygg는 2008년 1월 미국 GSA, 핀란드 Senate, DECA와 공동 협정을 맺음 • 싱가포르 e-플랜 체크 기술 도입을 위해 MOU를 체결했고, BYGGSØK시스템은 개방형 BIM과 IFG의 정보유통에 기반을 둔 자동 법규 케크 및 개발 영역 승인을 제공함.
덴마크	• 2007년 1월부터 BIM의 채택과 IFC를 요구함. • 개방형 BIM 기반의 '3D CAD-매뉴얼 2006' 지침을 개발함. • 모델 서버가 실제 건설 프로젝트에서 테스트베드될 예정임.	• NAEC (National Agency for Enterprise and Construction)가 주도한 디지털 건설 프로젝트에서 3D CAD-매뉴얼 2006 지침을 개발함
독일	• 빌딩스마트 GS 주도로 개방형 BIM 데이터의 유통 및 교환을 위한 'User Handbook Data Exchange BIM/IFC' 지침을 개발함. • 빌딩스마트 GS 주도로 시설물 관리에 대한 IFC/FM 뷰 구현, 개방형 BIM을 이용한 물량산출 프로젝트 수행, 구조분석 모델의 정보교환, 독일 ENEV/DIN18599에 따른 에너지 계산용 뷰 정의 프로젝트 개시, 그리고 IFC 2X3 지원 소프트웨어의 인증을 추진함.	• 제품 데이터 교환을 위한 연방 법규로 'BFR Gbestand'가 제정됨.
영국	• BS와 연계된 빌딩스마트 UK 표준화 활동을 추진 중 • BIM의 도입 및 적용을 위해 BS1192-CAD 표준의 확장, BIM과 상호운용을 위한 Bluffers지침을 개발 중 • 협력작업 지원을 위한 업무 사례 가이드 및 BIM 지침의 개발을 추진 중	• EU와 Avanti Programmes와 협조하여 정부 정책에 반영하고 있음 • 빌딩스마트 UK와 공공기관의 연대를 통해 정보 표준화를 추진함.
싱가포르	• 2002년에 설계도면 검토를 완전 전자화함. • 개방형 BIM에 의한 설계도면 자동 검토 및 웹 포털(CORENET시스템) 적용을 준비 중 • CORENET시스템에서의 실무 적용을 위해 개방형 BIM 데이터의 조달 및 납품을 위한 지침 개발 추진 중 • IFC2X2 Code Checking에 관한 자금 지원과 빌딩스마트 SG와 BCA(Singaporean Building and Construction Authority)간 MOU를 체결함	• BCA 정부기관 주도로 CORENET 시스템에서 개방형 BIM 납품의 의무화를 전개 중
호주	• 호주 건설 혁신을 위해 CRC(Cooperative Research Centre) 주도로 협력 모델 서버 플랫폼, National BIM Guidelines, 토목·조경 분야 및 건축물 객체 연구에 주력함	• 정부 출연 연구기관인 CISRO(Commonwealth Scientific and Industrial Research Organization)주도로 정책 개발을 추진 중
일본	• JACIC(Japan Construction Information Center)주도로 싱가포르의 CORENET시스템에 초점을 둔 개방형 BIM 연구 추진 중	• 정부 연구기관인 JACIC 주도로 개방형 BIM 연구 및 적용 추진 중
중국	• 가상 도시 모델링 프로젝트에서 IFC 적용 추진 중 • ArchiCAD, ETABS, SAP2000, Tekkla와 같은 IFC 지원 소프트웨어의 실무 적용 연구 프로젝트 추진 중 • 국가 11차 5개년 계획에서 협력 설계 프로젝트에 관한 IFC 적용을 제안함.	• 개방형 BIM을 국가 표준으로 제정함. • 개방형 BIM의 유통 및 납품을 위한 관련 지침 개발 추진 중
프랑스	• 개방형 BIM의 적용과 ICT 이용을 촉진하기 위해 eXpert프로젝트를 추진함. • GPEP Inventaire 프로젝트에서 기존 건축물의 개방형 BIM 적용이 진행됨.	• Bourgogne 구에서 두 개의 시범 프로젝트가 추진되고 있으며, Senate Properties사가 개발한 지침에 따라 개방형 BIM을 적용할 예정임.

dCollection @ chosun

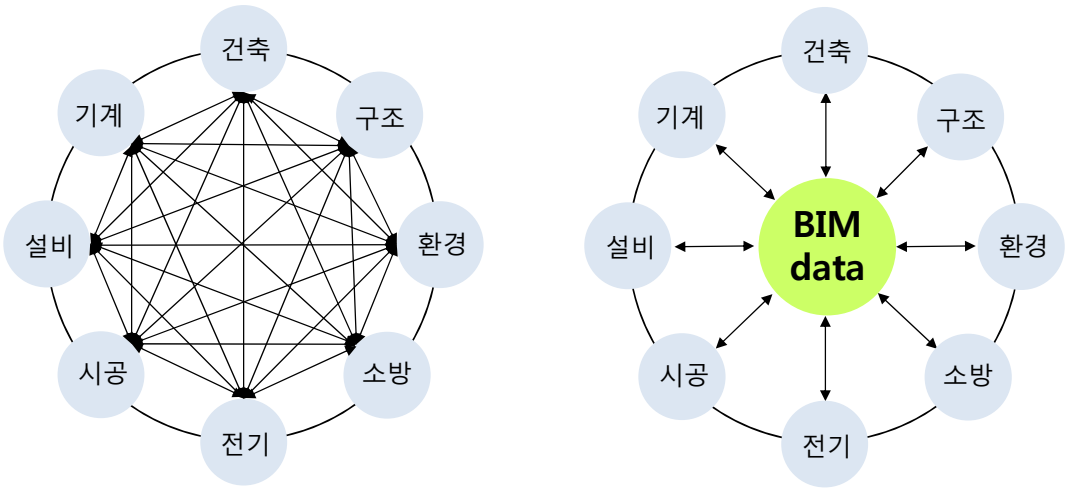
2.1.3 BIM 프로세스 및 정보호환[5,11]

건설 산업은 다양한 단계별 분야가 융합되어 구성된다. 프로젝트를 진행하기 위해 전 단계에서 산출되고 현재 단계에서 요구되는 정보가 융합되며, 단계를 거치면서 추가적으로 다양한 정보가 산출된다. 예를 들어, BIM 기술 적용 이전의 설계 단계에서는 2차원 CAD프로그램을 활용하여 2차원 도면(평면도, 입면도, 단면도 등)을 산출한 후 각 후속 단계를 진행하였다. 그러나 이러한 2차원 도면은 건물을 구성하는 빌딩 객체(벽, 문 등)의 표현을 단순 기하 정보(선, 호 등)로 인식하므로 빌딩 객체 내에 다양한 정보를 담기에는 한계가 있었다. 2차원 기반에서 진보한 3차원 CAD프로그램도 기하 정보의 3차원 표현으로 빌딩 객체 표현의 이해력을 증가시키기는 했지만 정보 표현의 한계는 여전하였다.

BIM은 파라메트릭 기술을 적용하여 지능적인 빌딩객체(벽, 슬래브, 창 등) 각각의 속성(기능, 구조, 용도)을 표현할 수 있고, 설계방식을 2차원에서 다차원으로 실현시키는 구조이므로 물량, 비용, 일정, 자재목록에 관한 정보를 제공할 뿐만 아니라 구조 및 환경을 고려한 데이터 분석을 가능하게 한다. BIM에 의한 건설정보 생산방식의 변화는 여러 측면에서 살펴볼 수 있다. 조립식 시공방식의 일반화, 건축물 유지관리 및 보수 문제 증가, 에너지·친환경 및 그린 건설 등의 이슈로 인해 건설 프로세스가 변화하고 있다. 미국 상무국의 조사에 따르면 다른 산업 분야의 생산성은 과거 대비 계속 성장세이나 건설 분야의 생산성은 매년 감소하는 추세이다. 하지만, BIM의 등장은 건설 프로세스 변화의 계기가 되고 있다.

BIM기반 프로세스는 초기 설계 단계부터 참여 주체 간 협업과 의사결정을 수반하기 때문에, 초기 단계에서 결정된 가치정보는 각 건설 단계별로 지속적으로 공유, 축적, 활용될 수 있다. 따라서 BIM 기반의 프로세스에서 정보는 자산 가치로서 활용 될 수 있으며, 건설의 마지막 단계인 유지관리 및 보수단계에서는 축적된 정보의 활용을 극대화 할 수 있다.

dCollection @ chosun



(a) 일대일 정보 교환 (b) BIM을 활용한 통합설계 모델

그림 2.5 BIM을 이용한 상호운용성

그림 2.5에서와 같이 BIM의 상호운용성이란 서로 다른 형식의 포맷으로 이루어진 소프트웨어 사이의 BIM 데이터를 교환하여 건물의 전 생애주기에 걸쳐 프로젝트에 참여하는 여러 전문가가 정보를 교환하고 의사결정을 하는 것을 말한다. 그림 2.5(a)와 같이 전통적인 건물 프로젝트에서 전문가 그룹간에 일대일 정보 교환이 이루어졌다면, 그림 2.5(b)에서는 BIM 활용을 통해 건물의 모든 정보를 포함하는 하나의 공유 모델이 바탕이 되어 각각의 전문가 또는 해당 그룹 간 정보교환이 이루어진다.

특히 건축설계 과정에서 BIM은 다양한 동적 시뮬레이션 모델을 이용하여 에너지 해석 결과물을 도출할 수 있으며, 이를 설계 단계에 반영할 수 있다. BIM 기반 건물 프로젝트는 기존 프로세스와 달리 정보를 교환할 때 필요한 중복작업 및 정보 입력을 위한 시간, 비용, 노동력 손실을 최소화할 수 있다. 예를 들어, 2D 기반의 CAD 시스템에서는 작성된 정보를 바탕으로 에너지 성능 분석 시뮬레이션 툴에 요구되는 3D 모델링 작업 및 다양한 추가 작업(벽, 바닥, 창, 지붕 등의 재료 정보, HVAC, 재실자, 조명, 기기의 운영 일정 등)이 필요하다. 또한, 기존 설계 업무 프로세스로는 시시각각 변화하는 설계 대안을 분석하는 것은 매우 어려운 일이다. 하지만 공유화된 정보 모델을 이용한 BIM 환경에서는 에너지 해석을 위해 필요한 Schema 정보가 포함되므로, 에너지 해석 프로그램과 연계하여 업무 효율성을 증가시킬 수 있다. 또한 설계 작업에 대해 실시간 피드백이 가능해지므로 에너지 분석 준비 단계가 매우 간편해진다.[11]

dCollection @ chosun

이상적인 BIM 환경은 설계 단계에서 환경 분석 프로그램을 신속하게 적용하여 보다 좋은 성능의 건물을 디자인할 수 있도록 하는 데 목적이 있다. 현재 서로 다른 프로그램 간의 데이터 중심 교환, Plug-in 형태의 정보 교환, 표준 포맷 활용 등 다양한 방법으로 상호 운용을 진행하고 있지만, 다양한 가정과 한계(BIM 데이터 매핑 자동화의 어려움, 시뮬레이션 툴의 불확실성, 모델링 과정에서 필수적으로 수반되는 현실의 단순화와 가정, 데이터의 변형과 누락)가 존재한다.[5]

이와 함께 현재 건설 프로세스는 초기 설계 단계부터 다양한 이해 당사자가 참여하는 IPD(Integrated Project Delivery), 즉 통합프로젝트 발주 또는 프로세스로 변화하고 있다. 미국건축가협회(AIA)에서 처음 제안한 IPD는 그간 건설 프로세스에서 발생되는 협업, 품질, 의사소통 등의 문제를 해결하기 위해 시도되고 있다. IPD는 기존의 발주 방식, 설계·시공의 분절, 정보의 지속적인 관리 및 활용 체계의 미비, 초기 협업 부재 등의 문제를 해결하기 위한 통합프로세스 또는 통합발주 방식을 일컫는다. 즉 발주처, 설계사, 시공사, 엔지니어사, 컨설턴트 등이 협업체계를 구성하여 사업 구조 및 업무를 단일화된 프로세스로 통합하여 프로젝트를 수행한다. 모든 참여 주체가 책임, 역할, 성과 등을 공동 분배하는 발주 방식이라 할 수 있다. 이와 같이 현재 건설 프로세스는 IPD의 도입과 함께 변화하고 있다. 이는 BIM 기술과 같은 협업을 가능하게 하는 건축정보 모델이 개발되고 실무에 적용함으로써 가시적인 성과와 가치가 입증되고 있기 때문이다. 건설 산업의 생산성 하락과 그린 건설에 대한 인식 증대 등도 건설 프로세스의 변화 요인이 되고 있다.[5]

BIM 기반 프로젝트를 수행하기 위해서는 건설산업 각 단계에서 다수의 BIM지원 소프트웨어가 활용된다. 각 소프트웨어에서 산출되는 데이터는 표준적인 방법으로 호환 되어야 한다. 이때 BIM 국제 표준인 IFC와 같은 중립 포맷을 통하여 소프트웨어 간 데이터 호환이 가능한 BIM 환경을 개방형 BIM환경' 이라고 일컫는다. IFC는 소프트웨어 간 데이터 호환을 위해 건물의 생애주기를 다루는 표준 데이터 세트 개발과 실무적용이 동시에 이루어지고 있다. 건물 객체는 가상의 BIM 응용도구 환경에서 실세계를 표현하기 위해 BIM을 적용하여 빌딩 객체를 정의한다. 이와 비슷하게, IFC는 실세계에 존재하는 건물 객체에 대해 표준 데이터 세트로 정의된다. 표준 데이터 세트는 BIM 응용도구 간의 BIM 객체를 교환하기 위해 이용되며, BIM 객체는 IFC 객체 모델 인스턴스로 변환된다. 결국, IFC기반 BIM은 BIM 응용도구 간 BIM 데이터를 교환하기 위한 새로운 개념으로 정의할 수 있다.

dCollection @ chosun

2.2 스마트그린 구조시스템

2.2.1 스마트그린 구조시스템의 개요

최근, 전 세계적으로 CO_2배출 저감을 위한 많은 노력들이 이루어지고 있다. 그림 2.6과 같이 건축물의 생산과 운영과정에서 발생하는 지구 환경 부하는 세계 총에너지 소비 및 온실가스 방출량의 30~40% 정도를 차지하고 있다. 건설산업분야의 친환경·자원절약형 건설기술의 개발은 국가기술지도(NTRM)에서 제시하는 건설분야 발전전략의 핵심이라 할 수 있다. 건설 분야에서는 건축물 생산 시 발생하는 건설산업 폐기물을 절감하고 CO_2배출량 저감 등 환경부하를 줄이기 위하여 폐기물 저감(Reduce) 및 부재의 재사용(Reuse)이 용이한 건축시스템의 개발을 꾸준히 진행해 왔다.

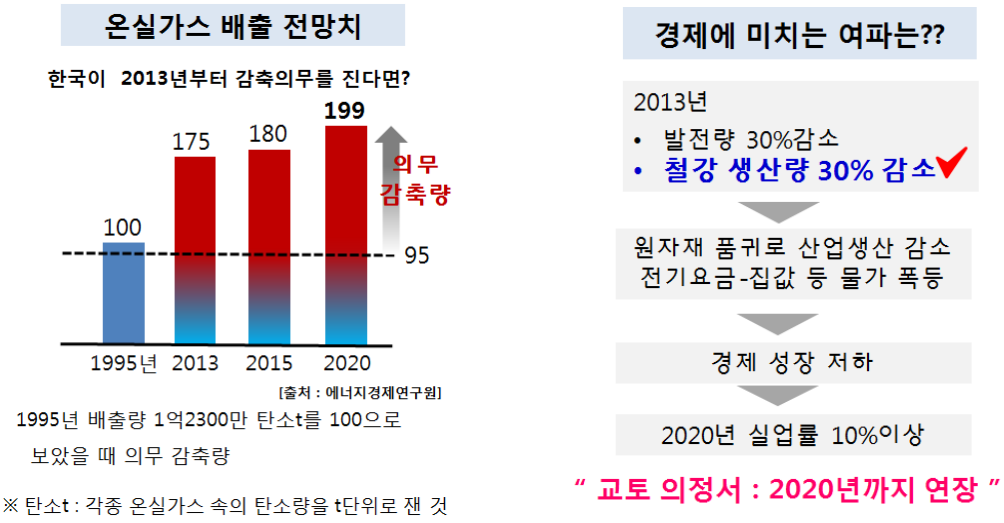


그림 2.6 한국의 온실가스 배출 의무 감축량

폐기물 관리법의 자원화 개념에서 재활용(Recycle)은 "발생되는 폐기물을 중간처리 과정을 거쳐 이를 원래의 용도 또는 타 용도의 원재료로 재사용하는 것"으로 정의하고 있다. 또한 재사용(Reuse)은 현 상태 그대로 또는 변형하여 원래의 용도 또는 타 용도로 재사용 하는 것"으로 정의하고 있다.[36]

그림 2.6, 2.7에서와 같이 철골 부재의 공장 제작과정, 건물 해체과정, 철강 재생산

dCollection @ chosun

과정에서 모두 적지 않은 양의 CO_2가 발생한다. 건설산업 중 강재분야에서 CO_2발생량을 줄이기 위해 그림 2.7과 같이 공장 제작, 건축물 생산, 건축물 해체, 철강 재생산의 Recycle 과정을 거치지만, 강재의 Recycle을 위해 막대한 에너지가 소비되며, 다량의 CO_2가 발생한다. 건축물의 해체 시 손상을 입지 않은 온전한 강재는 Reuse가 가능하나, Reuse를 고려하지 않는 구조설계와 강재의 손상여부를 판단할 수 있는 기술력의 한계로 실질적인 강재 Reuse는 미흡한 실정이다. 이러한 강재의 Recycle의 한계점을 극복하고, 강재 Reuse를 실현하기 위하여 해체를 고려한 구조시스템이 필요하다.

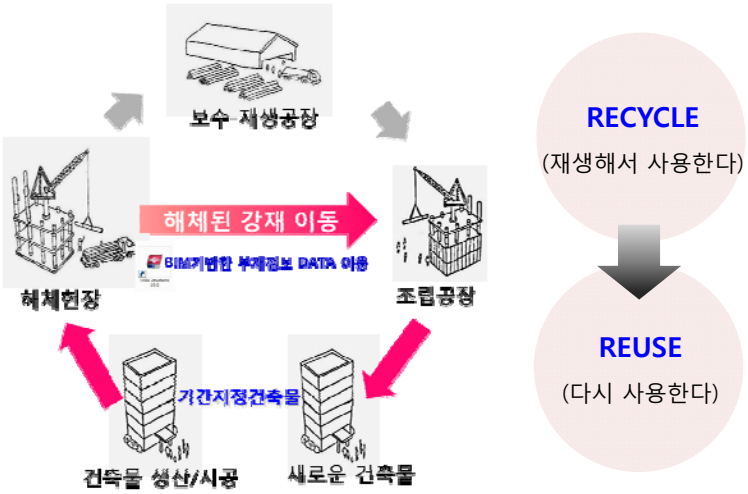


그림 2.7 스마트그린 구조 시스템 개념

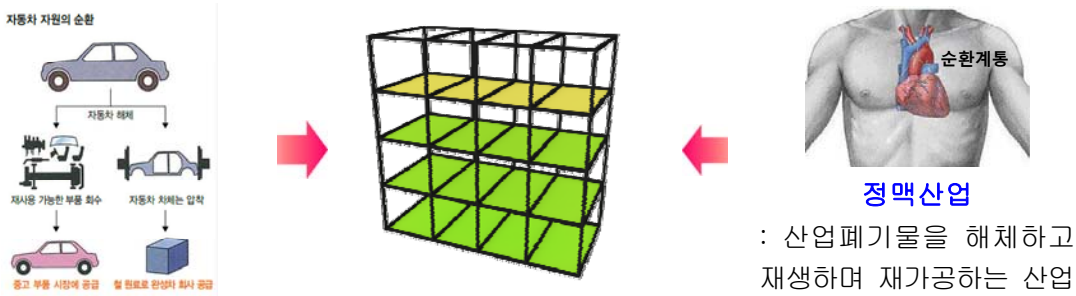


그림 2.8 DfD(Design for Disassembly)의 개념

dCollection @ chosun

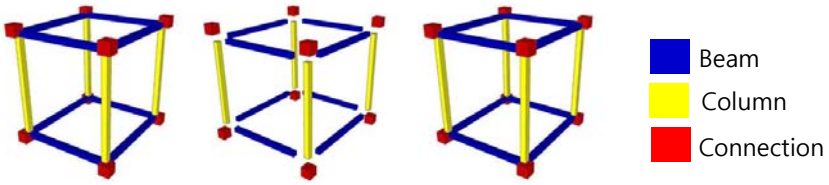


그림 2.9 분리 MECHANISM

한편, 기계공학 분야에서는 해체를 위한 설계의 개념으로 Design for Disassembly (이하, DfD)의 개념을 도입하여 활발히 연구를 하고 있다. DfD의 개념은 그림 2.8와 같이 재사용, 재활용이라는 폐기물의 재자원화를 구체적으로 실현하기 위한 방법이다. 즉, 리사이클링을 위한 분해 기술이라 할 수 있고, 이는 부품과 재료에 있어 복합적인 것을 배제하여 최대한 단일한 성격의 재료와 부품을 사용하도록 하는 것이다, DfD의 궁극적인 목적은 장기적 관점에서 제품디자인에 대한 연구와 실행을 통해 환경적 문제점을 개선하려는 노력이라 할 수 있다.[16,17]

이러한 강재의 Reuse를 실현하기 위하여 해체를 고려한 구조시스템이 스마트그린(Smart Green) 구조시스템이다. 스마트그린 구조시스템은 그림 2.9와 같이 설계단계부터 각 구조 요소의 조립·해체가 자유로운 구조시스템 공법의 집합체로 Flexible한 접합부, 수평저항장치 및 에너지 흡수장치 등의 Connection으로 구성된다.

2.2.2 스마트그린 구조시스템의 핵심 기술

스마트그린 구조 시스템의 구성 요소를 그림 2.10에 나타내었다. Reuse를 고려한 구조 시스템을 구축하기 위해서는 조립·해체가 자유로운 접합부를 구성할 필요성이 있다. 건축물의 초기 설계 단계에서 보-기둥 접합부 및 주각부는 Flexible한 접합부를 사용함으로써 구조물에 외력이 가해졌을 때 접합부로 손상을 유도하여 기둥과 보 등의 주 구조체의 손상을 최소화할 수 있다. Flexible한 접합부를 사용함으로써 부족해진 수평 내력은 수평저항장치(브레이스) 및 에너지 흡수장치(댐퍼)를 적용함으로써 신뢰성 높은 구조 시스템을 완성하였다.

dCollection @ chosun

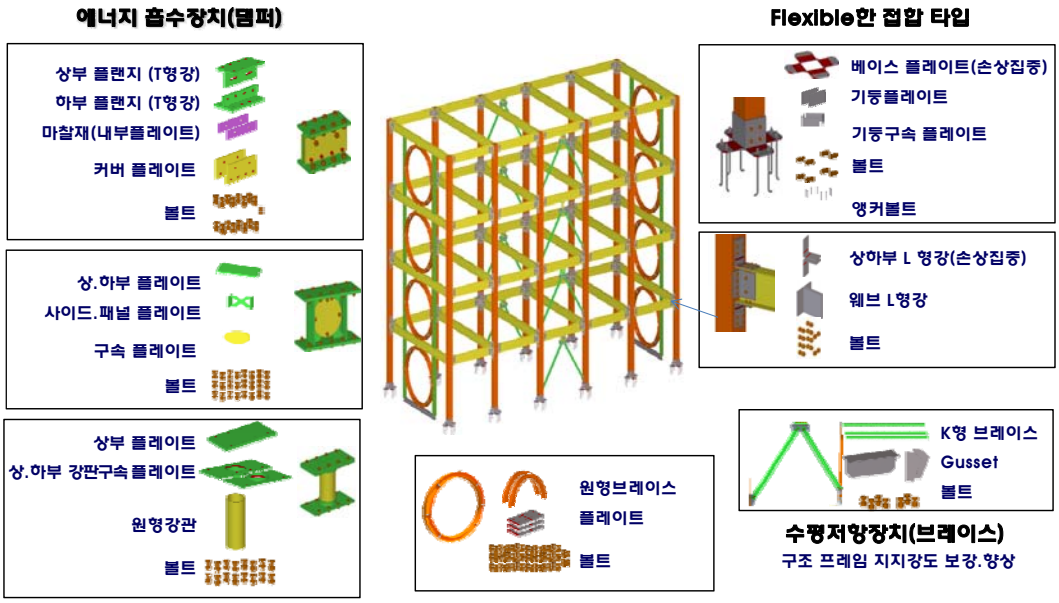
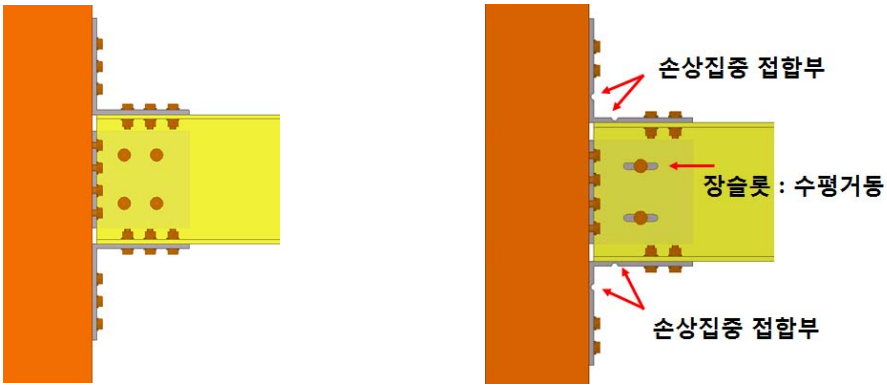


그림 2.10 스마트그린 구조시스템의 구성요소

주 구조체의 손상을 방지하고 접합부로의 손상을 유도하기 위한 보-기둥 접합부를 그림 2.11와 같이 구성하였다.

보-기둥의 접합부는 기존 반강접(Semi-rigid) 접합부 중 Top and Seat Angle 접합부를 기초로 하였다. 접합부에 손상을 유도하기 위해 상하부 앵글 내에 임의적으로 단면적을 감소시켜 손상을 집중 시켰다. 수평하중이 작용하여 수평변형이 발생하였을 때 기둥, 보 부재의 전단접합부에서 손상을 방지하기 위해 보의 웨브 접합부분에 장슬롯을 두었으며, 수평하중의 작용으로 수평변형이 발생하더라도 보 부재에 손상이 발생하지 않게 하였다.

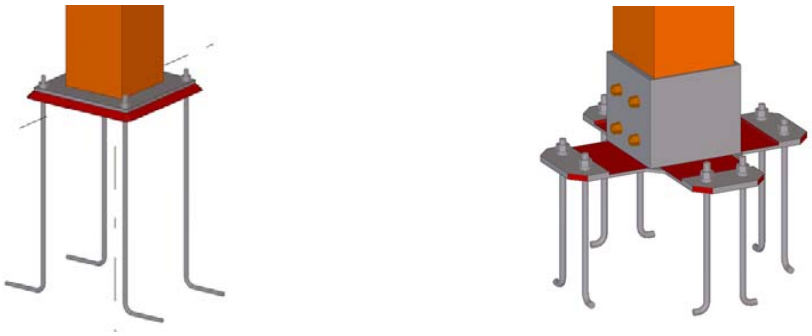
dCollection @ chosun



(a) 기존 접합부 (b) 손상집중 접합부

그림 2.11 Flexible한 보-기둥 접합부

기존의 주각부의 경우 기둥과 베이스 플레이트 용접을 통해 접합을 하고, 베이스 플레이트는 앵커볼트를 정착하여 콘크리트 기초와 일체화시켜 시공한다. 기존 주각부의 경우 지진하중 및 풍하중에 의해 주각부의 변형 발생 시, 베이스 플레이트 항복(베이스 플레이트 항복선행형)이 진행 된다. 이는, 베이스 플레이트와 기둥 사이가 용접으로 접합되어 있기 때문이다. 베이스 플레이트 손상만으로도 기둥의 재사용이 불가능한 문제점을 개선하기 위해 기둥과 베이스 플레이트를 보-기둥 접합부와 같이 Angle로 인한 손상집중 구간을 구성하여, 기둥의 손상을 방지하고 구조물의 해체 후 기둥의 재사용을 가능하게 하였다.



(a) 기존 주각부 (a) 손상집중 주각부

그림 2.12 조립・해체가 자유로운 주각부

dCollection @ chosun

그림 2.11, 2.12와 같이 Flexible한 보-기둥 접합부 및 주각부만으로 구조체를 구성할 시 수평하중에 대해 매우 취약하게 된다. 따라서 수평저항능력을 확보하기 위해서 수평저항장치 및 에너지 흡수장치가 필요하다. 그림 2.13, 2.14와 같이 원형 브레이스 및 댐퍼를 사용하여 수평내력을 확보할 수 있다.

그림 2.13 원형브레이스

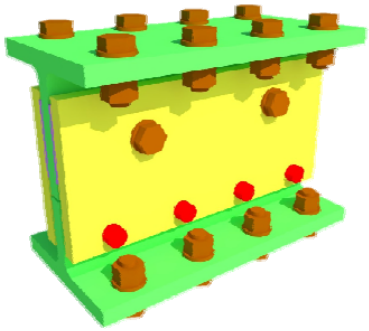


그림 2.14 전단형 마찰댐퍼

2.3 OpenAPI

API(Application Programming Interface: 응용 프로그래밍 인터페이스)는 응용 프로그램에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻한다. 주로 파일 제어, 창 제어, 화상 처리, 문자 제어 등을 위한 인터페이스를 제공한다.

OpenAPI란, 사용자가 일방적으로 사용자 인터페이스(UI: User Interface)를 제공받는데 그치지 않고 응용 프로그램과 서비스를 개발할 수 있도록 공개된 인터페이스이다. 어플리케이션을 만들 수 있는 플랫폼을 무상으로 자유롭게 사용할 수 있는 서비스이므로 개발자는 제공된 OpenAPI를 사용하여 직접 어플리케이션을 작성할 수 있다.

즉, OpenAPI는 누구나 접근하여 사용할 수 있는 공개 정보로써, 간단하고 직관적인 인터페이스의 제공을 통해 사용자의 참여를 유도하는 사용자 중심의 비즈니스 모델로 데이터 제어가 가능하다.

dCollection @ chosun

2.3.1 .NET FRAMEWORK

.NET FRAMEWORK 는 빠르게 변화하는 개발환경과 인터넷으로 집중되는 시대에 부응하기 위하여 마이크로소프트사가 개발한 프로그래밍 환경이다. 이것은 다양한 컴포넌트로 구성되어 있으며 MSIL이라는 중간언어를 사용하여 통합된 프로그래밍 환경을 제공한다. .NET FRAMEWORK의 특징은 플랫폼에 독립적이면서 프로그램을 개발하기가 쉬어졌다는 것이다.[6] 그림 2.15은 .NET FRAMEWORK의 구조도이다. 프로그램 사용 전 System Collections의 using 선언이 필요하며 프로그램 개발환경을 선행하여야 한다. 클래스(Class)는 필드(Field), 프로퍼티(Property), 메소드(Method)등의 데이터 연산자로 객체 생성이 가능하다. Collection은 string, int등의 변수집합으로 배열의 발전된 기능의 객체집합이다. 개발된 프로그래밍의 코딩이 완성되면 CSC(C Sharp Compiler)를 통해 빌드(compile) 해야 한다. 프로그램에서의 데이터 타입별로 데이터들을 정수, 실수, 문자, 문자열들을 구분하여 처리한다. .NET은 플랫폼 자체가 객체 지향적이며 .NET 언어들도 모두 완전한 객체 지향 언어이다. 이러한 .NET 개발 환경은 OpenAPI를 참조하여 모델 DB와 기능에 접근하여 제어하기 용이하다.[6-10]

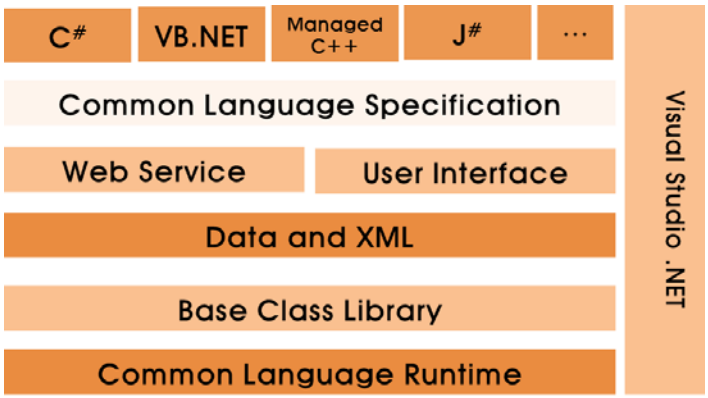


그림 2.15 .NET 프레임워크 구조도

2.3.2 TEKLA OpenAPI

.NET API로 알려진 Tekla OpenAPI는 Tekla Structures 프로그램에서 모델과 직접적으로 상호작용을 하고, third party applications를 위해 인터페이스를 제공한다.[40]

dCollection @ chosun

Tekla.Structures.dll과 Tekla.Structures.Model.dll은 참조가능 파일로 Tekla Structures 모델 DB를 링크 시킬 수 있으며, 오브젝트를 생성할 때 사용되는 클래스(Class)와 메소드(Method)들로 구성 되어 있다. Tekla.Structures.Model.dll은 Tekla. Structures.Model 네임스페이스와 Tekla.Structures.Model.UI 네임스페이스로 구분된다. Tekla.Structures.Model 네임스페이스는 Beam(Beam, Column, Panel), ContourPlate, BoltArray, Load, Component등 모델을 생성할 때 Model이라는 객체정보를 얻어올 수 있는 클래스를 포함하고 있으며, Tekla.Structures.Model.UI 네임스페이스는 Tekla Structures의 모델 DB로부터 객체정보를 불러오거나 불러온 객체들을 저장할 때 사용되는 클래스를 포함하고 있다. Tekla OpenAPI는 Model, Drawing, Scripting 등의 DLL(Dynamic Linked Library)로 구성되어 있다. DLL은 .NET Assembly라고도 하며, 연관된 기능 및 클래스를 구분하는 네임스페이스의 테두리 안에서 클래스(Class)와 메소드(Method)로 구성되어 있고, 클래스는 다시 멤버 변수와 멤버 함수로 구성된다. Tekla OpenAPI에서 구성되는 어셈블리는 다음과 같다.[25,40~42]

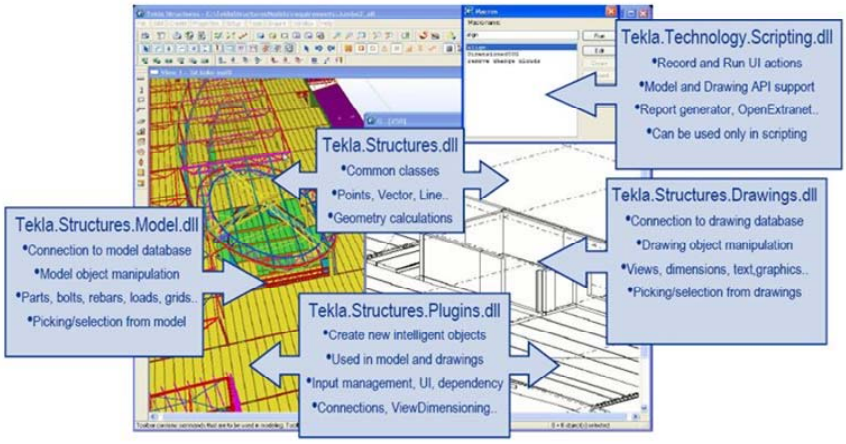


그림 2.17 Tekla OpenAPI

1. 공통 메소드

Tekla OpenAPI의 모든 객체 클래스는 객체를 조작할 수 있는 공용 메소드를 가지며 그 내용은 다음과 같다.

dCollection @ chosun

- Select() : 인터페이스 또는 모델 DB로부터 객체를 선택한다.
- Insert() : 새로운 모델의 인스턴스(클래스 객체)를 생성한다.
- Modify() : 인터페이스 또는 모델 DB에서 객체를 수정한다.
- Delete() : 인터페이스 또는 모델 DB로부터 객체를 삭제한다.

2. Tekla.Structures.dll

Tekla.Structures.dll은 Tekla Structures의 모델과 드로잉에서 사용되는 공용 클래스(common class)와 메소드(method)로 구성되어 있으며, Tekla Structures와 Tekla.Structures.Geometry3D 네임스페이스로 구분된다. Point, Vector 등의 기하학적 데이터를 처리할 수 있는 클래스(class)를 포함하고 있으며, 거리 및 교차점 계산, 평면의 이동 등과 같은 Geometry Library를 포함하고 있다.

3. Tekla.Structures.Model.dll

Tekla.Structures.Model.dll은 Tekla Structures의 모델 DB에서 객체 정보를 연결하거나 조작할 때 사용되는 클래스(class)와 메소드(method)로 구성되어 있다. 모델을 생성할 때 사용되는 객체들의 클래스(class)는 Beam, ContourPlate, BoltArray, Load, Component 등으로 기술한다. 한편, copy, move, combine and split 등은 기본 객체 운영을 포함하고 있으며, 모델의 객체를 조작하기 위한 메소드(method)이다. Tekla.Structures.Model 네임스페이스 어셈블리는 ClashCecker, Collaboration, History, Operations and UI를 조작하기 위한 하위 네임스페이스를 포함하고 있다.

4. Tekla.Structures.Datatype.dll

Tekla.Structures.Datatype.dll은 Tekla Structures 와 기타 Tekla Structures 어셈블리 사이에 데이터를 전달할 때 사용되는 unit type이며, 그 유형은 다음과 같다.

dCollection @ chosun

- Distance
- DistanceList
- Boolean
- Double
- Integer
- String

2.4 소결

본 장에서는 BIM의 전반적인 개요와 스마트그린 구조시스템의 개요에 대해 이론적 고찰을 함으로써 다음과 같은 결론을 얻었다.

1. 전세계적으로 2D 기반 정보교환 방식의 디자인 한계, 시공성 검토 미흡, 물량산출 오류 및 방대한 건축정보 통합관리의 어려움에 따라, 2D 기반의 한계를 극복하기 위해 BIM이 적용되며 성공적으로 프로젝트가 수행되어지고 있다. 국내에서도 BIM이 정책적으로 적용되고 있어 건설시장에서 BIM을 적용한 프로세스의 도입 필요성이 강조되고 있다.

2. 건설시장의 필요에 의해 도입된 BIM 프로그램에서 구조물을 모델링 하였을 때 표준 공법을 사용한 모델 디테일에 대하여 현재 제공하고 있는 기존 라이브러리의 조합으로 쉽게 표현이 가능하다. 하지만, 기존 라이브러리에서 제공되지 않는 특수구조형상 디테일의 경우 모델링 제한이 있고, 제한된 라이브러리의 한계를 극복하기 위해, .NET 환경에 최적화되어 있는 C#을 이용함으로써, OpenAPI를 통한 설계 모듈을 개발할 수 있다.

3. OpenAPI를 통한 설계 자동화 모듈은 스마트그린 구조시스템을 이루고 있는 핵심기술로써 Flexible한 접합부, 수평저항장치 및 에너지 흡수장치의 특수형상 디테일을 대상으로 한다.

dCollection @ chosun

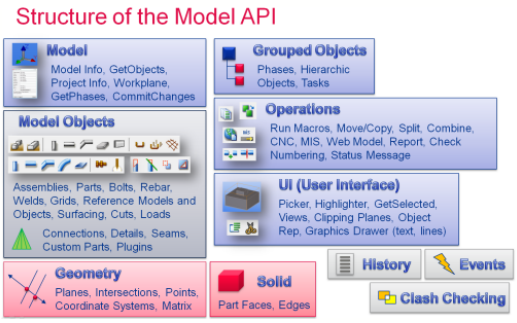
제3장 OpenAPI를 통한 스마트그린 구조시스템 설계자동화 모듈 개발

3.1 Tekla OpenAPI를 통한 설계자동화 모듈 개발 환경

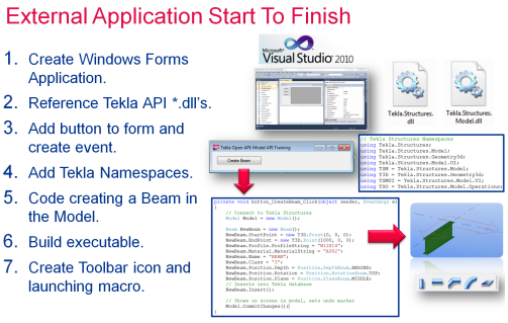
3.1.1 Tekla OpenAPI의 클래스와 객체

클래스(class)는 객체의 구조(structure)와 행위(behavior)를 정의하는 방법으로 C# 프로그래밍의 기본 단위이다. 즉, 하나의 C# 프로그램은 하나 이상의 클래스(class)로 구성된다. 즉, 클래스(class)는 객체지향 언어의 특성 중에 하나이며 이것은 프로그램의 유연성(flexibility), 이식성(portability) 및 재사용성(reusability)을 높인다. 클래스의 인스턴스(Instance)로 변수와 같은 역할을 하는 객체 선언을 위해서는 해당하는 클래스를 정의해야 한다. 정의된 클래스로부터 객체를 선언하는 방법은 클래스 이름을 쓰고 그에 해당하는 객체들을 나열하는 방식이다. 이 객체의 생성은 new 연산자와 함께 생성자(constructor)를 기술한다.[6]

그림 3.1(a)에서와 같이 Tekla OpenAPI의 구조에서 모델 DB를 활용하여 클래스(class)를 선언하고, 3.1(b)의 External Application 프로세스에 의거하여 사용자가 원하는 모델 디테일을 개발한다.[6~10,40]



(a) Model API 구조도



(b) External Application 프로세스

3.1 Tekla OpenAPI를 이용한 객체 생성 구조도

dCollection @ chosun

3.1.2 개발언어인 C# 프로그램의 체계

Visual C#은 System component에서 제공하지 않는 신규 Component를 사용자가 원하는 형태로 개발할 수 있다. 객체 지향적 프로그래밍 언어를 사용하므로 각 구성요소별 모델링이 자유롭고 개발에 제한이 없다. 각 요소별로 체계적으로 분류된 특수구조형상 디테일을 Visual C# 으로 프로그래밍하여 형성한다. 그림 3.2는 C# 소스파일을 단계별로 실행시킨 모델의 운영 시스템이다. C#이 하나의 응용 프로그램이 되기 위해서는 반드시 Main() 메소드가 존재해야 한다. 응용 프로그램에서 Main()은 실행이 처음 시작되는 부분이다. 그림 3.2에서와 같이 C# 언어로 작성된 응용 프로그램 중 winform application은 일반적인 윈도우 응용 프로그램과 같이 그래픽 사용자 인터페이스(GUI: Graphical User Interface)를 통해서 입출력을 수행하며 이벤트 처리방식을 통해 실행되는 윈도우 폼 응용 프로그램이다.[6,40]

본 연구의 특수구조형상의 디테일은 그림 3.2의 C# 프로그램 실행 모델을 근거로 단계별로 개발을 진행하였다.

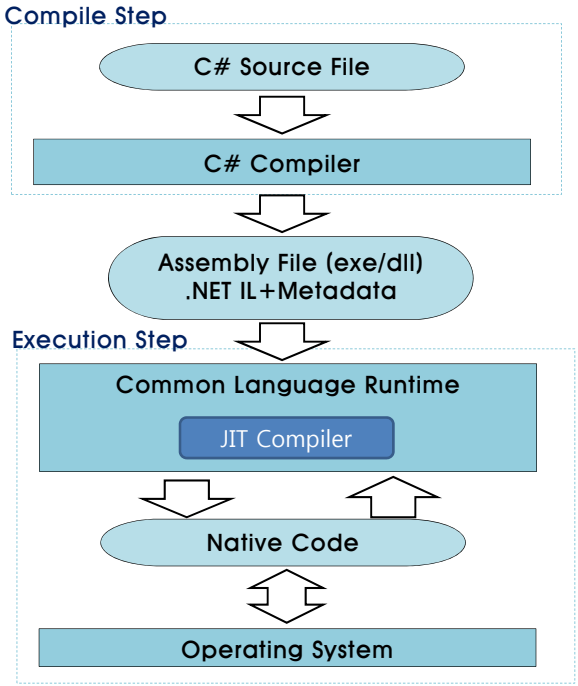


그림 3.2 C# 프로그램의 실행 모델

dCollection @ chosun

그림 3.3은 본 논문의 특수구조형상의 신규 라이브러리를 개발하기 위한 C# 프로그래밍의 개발 흐름도이다. 기존의 제한적이었던 라이브러리에 대한 극복이 가능하며 특수구조형상과 같은 신규 라이브러리의 디테일 모델에 대하여 제약 없이 개발이 가능하다.

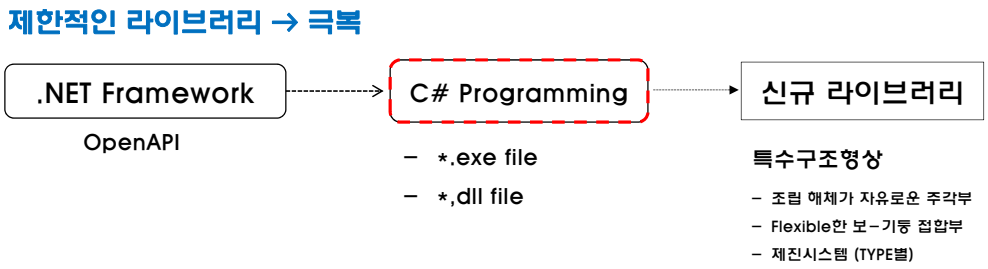


그림 3.3 C# 프로그래밍의 개발 흐름도

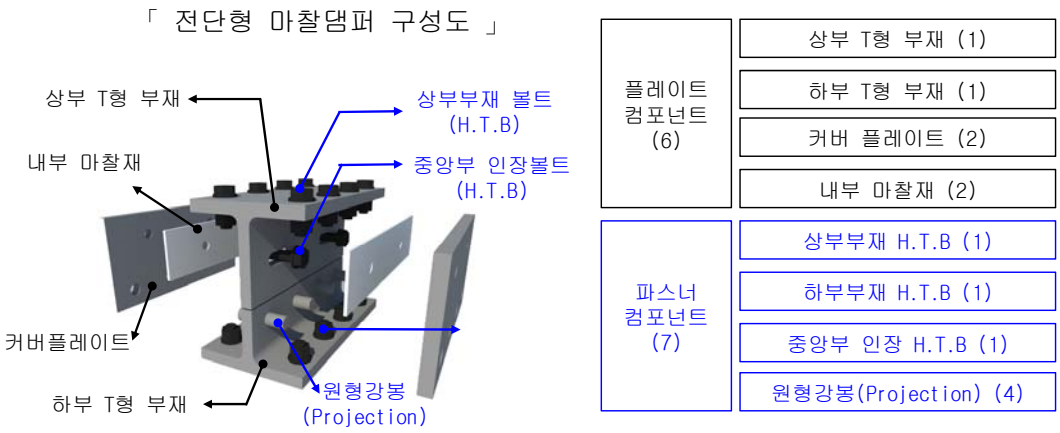
3.2 에너지 흡수장치 설계자동화 모듈 개발

3.2.1 전단형 마찰댐퍼 설계자동화 모듈 개발

3.2.1.1 전단형 마찰댐퍼의 정의

스마트그린 구조시스템은 조립·해체가 자유로운 구조로 건축물의 경제성과 안전성을 좌우하는 핵심 기술의 집합체이다. 스마트그린 구조시스템의 핵심 기술 중 전단형 마찰댐퍼는 접촉하는 2종 재료 사이의 접촉부가 상대 변위에 의해 미끄러짐으로써, 미끄러지는 방향과 역방향으로 마찰력이 발생하게 되고, 이 마찰력이 구조물의 진동에 의한 소성변형을 열에너지 형태로 발산시키는 제진장치이다. 전단형 마찰댐퍼는 지진이나 풍력과 같은 외력에 대한 흔들림을 상쇄시켜주는 기술로 신규 특수한 모델 종류에 속하며 BIM 모델 운용 측면에서 필수 구축요소라 할 수 있다. 그림 3.4(a)는 전단형 마찰댐퍼의 구성도를 나타내며 C#으로 전단형 마찰댐퍼를 개발하기 위한 구성요소 분류체계를 그림 3.4(b)에 나타내었다. 전단형 마찰댐퍼는 상부 T형 부재, 하부 T형 부재, 내부 마찰재, 커버플레이트, 고장력 볼트로 구성이 이루어지며 상부 T형 부재와 내부 마찰재 사이에서 마찰력이 발생하게 된다.

dCollection @ chosun



(a) 구성요소의 분류 (b) 구성요소 분류체계

그림 3.4 전단형 마찰댐퍼의 구성요소

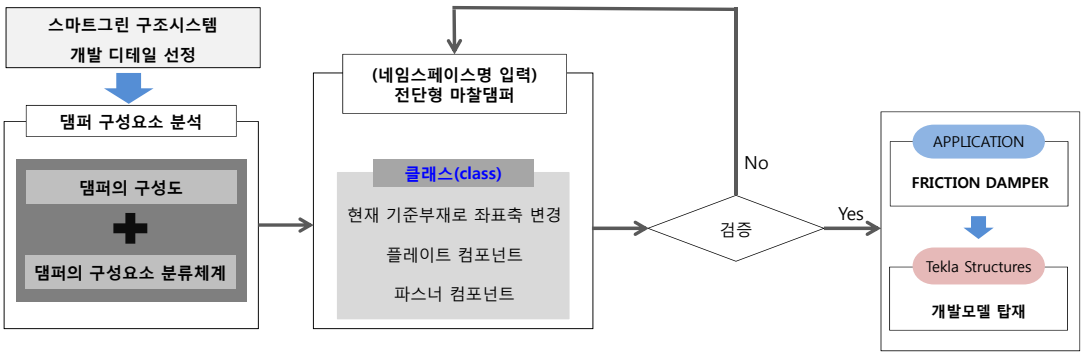


그림 3.5 전단형 마찰댐퍼의 개발 프로세스

본 연구에서는 댐퍼의 구성요소에 의거하여 그림 3.5과 같은 댐퍼 개발 프로세스를 진행하였다. 전단형 마찰댐퍼는 플레이트 컴포넌트 및 파스너 컴포넌트로 구분하여 각 세부 오브젝트를 구성하였다. 첫 번째로 플레이트 컴포넌트 파트는 상·하부 T형 부재, 커버 플레이트 및 내부 마찰재로 구성하였고, 파스너 컴포넌트는 상·하부 부재 볼트, 중앙부 인장 볼트 및 원형강봉으로 구성하였다. 각각의 컴포넌트 구성요소별로 Beam class, ContourPlate class 및 BoltArray class를 통한 오브젝트를 선언하여 컴포넌트 부위와 위치정보를 조합하면 전단형 마찰댐퍼가 생성된다. 클래스(class)는 객체를 만들기 위한 청사진으로 전단형 마찰댐퍼의 오브젝트를 생성하기 위한 틀의 역할

dCollection @ chosun

을 한다. 댐퍼의 오브젝트 는 각 부재의 속성(프로파일, 재질등)과 기능(수정, 삭제 등) 두 가지로 분류하여 제어한다.

3.2.1.2 전단형 마찰댐퍼의 Visual C# 입력 데이터

댐퍼 컴포넌트별 개발에 앞서 그림 3.6과 같이 부재의 글로벌 좌표계(X.Y.Z)를 현재 선택할 Main Part(H-Beam)의 로컬 좌표계(X.Y.Z)로 댐퍼 오브젝트의 축을 변경하였다. 현재 베이스 부재가 가지고 있는 영역으로 좌표계를 설정함으로써, 댐퍼의 좌표 설정이 용이한 이점이 있다.

<부재의 좌표축 변경>

WorkPlaneHandler class

```
WorkPlaneHandler _WP = _model.GetWorkPlaneHandler();


TransformationPlane OriginPlane = _WP.GetCurrentTransformationPlane();

Picker pick = new TSMUI.Picker();

Beam MainPart = pick.PickObject(TSMUI.Picker.PickObjectEnum.PICK_ONE_PART, "Select Main Part.") as Beam;
ContourPlate SecondaryPart = pick.PickObject(TSMUI.Picker.PickObjectEnum.PICK_ONE_PART, "Select Secondar


TSM.UI.ModelObjectSelector mos = new TSM.UI.ModelObjectSelector();
```

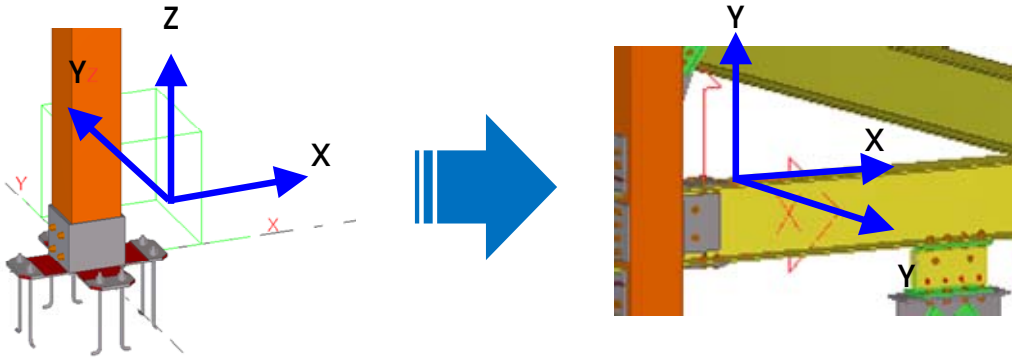
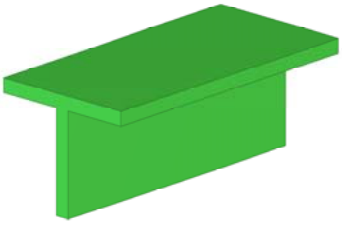


그림 3.6 좌표계 설정 변경 입력 데이터

dCollection @ chosun

Beam class **<상부 T형 부재>**

```
TSM.Beam B1 = new TSM.Beam();

double B1_Length = 400, B1_Heignt = 165, B1_Width = 200, B1_WebThickness = 20, B1_FlangeThickness = 20;

B1.GetReportProperty("LENGTH", ref B1_Length);
B1.GetReportProperty("HEIGHT", ref B1_Heignt);
B1.GetReportProperty("WIDTH", ref B1_Width);
B1.GetReportProperty("PROFILE.WEB_THICKNESS", ref B1_WebThickness);
B1.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref B1_FlangeThickness);

CoordinateSystem B1Coordi = B1.GetCoordinateSystem();


B1.Name = "DAMPER";
B1.Profile.ProfileString = "T165-20-20-200";
B1.Material.MaterialString = "SS400";
B1.Class = "3";
B1.Finish = "B1(UM)";
B1.StartPoint = new T3D.Point(1435, -200, 0);
B1.EndPoint = new T3D.Point(1435 + 400, -200, 0);
B1.Position.Depth = Position.DepthEnum.MIDDLE;
B1.Position.DepthOffset = -0.00;
B1.Position.Plane = Position.PlaneEnum.RIGHT;
B1.Position.PlaneOffset = 0.00;
B1.Position.Rotation = Position.RotationEnum.FRONT;
B1.Position.RotationOffset = -0.00000;
if (!B1.Insert())
    Console.WriteLine("B1 Insert failed!");
```

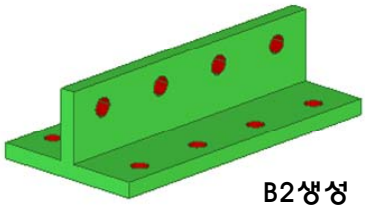


B1생성

그림 3.7 상부 T형 부재 입력데이터

Beam class **<하부 T형 부재>**

```
TSM.Beam B2 = new TSM.Beam(new Point(1435, -200 - B1_Heignt - 3, 0), new Point(1435 + 400, -200 - B1_He
B2.Name = "DAMPER";
B2.Profile.ProfileString = "T112-20-20-200";
B2.Material.MaterialString = "SS400";
B2.Class = "3";
B2.Finish = "B2(LM)";

B2.Position.Depth = Position.DepthEnum.MIDDLE;
B2.Position.DepthOffset = -0.00;
B2.Position.Plane = Position.PlaneEnum.LEFT;
B2.Position.PlaneOffset = 0.00;
B2.Position.Rotation = Position.RotationEnum.FRONT;
B2.Position.RotationOffset = 180;

if (!B2.Insert())
    Console.WriteLine("B2 Insert failed!");
```



B2생성

그림 3.8 하부 T형 부재 입력데이터

그림 3.7은 댐퍼의 상부 T형 강재의 입력데이터이고, 그림3.8은 하부 T형 강재의 입력데이터를 나타낸 것이다. 속성과 기능을 부여한 Beam 클래스 선언문에서, new 연산자의 Beam()은 생성자(constructor)라고 하는 특별한 메소드이다. Beam 클래스의 선언

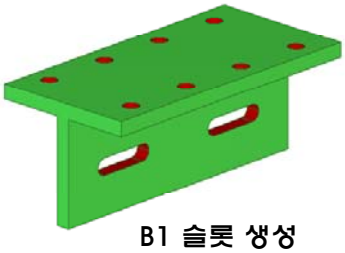
dCollection @ chosun

문에서 정의된 B1 및 B2는 객체를 뜻하며, 실체(Instance)라고도 기술한다. new 연산자는 생성자를 호출해서 B1과 B2 객체를 생성하는 데 중요한 역할을 한다. 사용자가 개발의 편의성을 위해 간단하게 정의한 객체 B1과 B2에서는 T형 강재에 대한 형상 프로파일, 재질 및 위치의 속성값을 입력한다. 이 모든 입력 데이터가 Tekla Structures 프로그램내에 3D 건축정보를 가지는 부재로 성공적으로 생성시키기 위해 Insert() 명령으로 완료한다.

ContourPlate class

<상부 T형 부재>

```
TSM.ContourPlate slotted_P1 = new TSM.ContourPlate();
TSM.ContourPoint point9 = new TSM.ContourPoint(new Point(1435 + 100 - 45, -200 - 100-11, 0), new TSM.Chamfe
TSM.ContourPoint point10 = new TSM.ContourPoint(new Point(1435 + 145, -200 - 100 - 11, 0), new TSM.Chamfer(
TSM.ContourPoint point11 = new TSM.ContourPoint(new Point(1435 + 145, -200 - 100 + 11, 0), new TSM.Chamfer(
TSM.ContourPoint point12 = new TSM.ContourPoint(new Point(1435 + 100 - 45, -200 - 100 + 11, 0), new TSM.Cha

slotted_P1.AddContourPoint(point9);
slotted_P1.AddContourPoint(point10);
slotted_P1.AddContourPoint(point11);
slotted_P1.AddContourPoint(point12);

slotted_P1.Profile.ProfileString = "PL20";
slotted_P1.Class = TSM.BooleanPart.BooleanOperativeClassName;
if (!slotted_P1.Insert())
    Console.WriteLine("slotted_P1 inser failed");

TSM.BooleanPart boolpart1 = new TSM.BooleanPart();
boolpart1.Father = B1;
boolpart1.SetOperativePart(slotted_P1);
if (!boolpart1.Insert())
    Console.WriteLine("Boolpart1 insert failed!");

Vector Vector1 = new Vector(200, 0, 0);
Vector Vector2 = new Vector(0, 0, 0);

if (slotted_P1.Delete())
{

    BooleanPart boolpart2 = Operation.CopyObject(boolpart1, Vector1) as BooleanPart;
    if (boolpart2 != null)
        Console.WriteLine("Boolpart3 insert failed!");

}
```



B1 슬롯 생성

그림 3.9 상부 T형 부재 슬롯 입력데이터

그림 3.9는 상부 T형 부재의 슬롯에 대한 입력데이터를 기술한 내용이다.

ContourPlate 클래스를 선언하여 slotted_P1이라는 객체를 지정하고 4개의 절점을 입력하여 슬롯 위치를 지정하였다. 컷팅할 부재에 대하여 생성 완료했던 상부 T형 부재의 객체 B1을 명시하고, 명시가 완료된 선행코드를 기반으로 슬롯은 Delete()를 호출하여 완성한다.

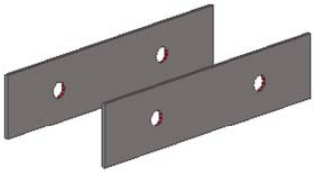
dCollection @ chosun

그림 3.10과 같이 내부 마찰재 입력데이터 또한 상기 T형 부재와 동일한 방식으로 Beam 클래스를 선언하고 new 연산자를 통해 Beam() 생성자를 호출하여 객체를 생성시킨다. 사용자가 직접 개발하기 쉽고 보기 편리하도록 B3 및 B4로 객체를 정의한 후 내부 마찰재를 생성한다. 그림 3.11의 커버 플레이트도 내부 마찰재와 동일한 코드 입력 방식을 취하였으며, 객체 B5 및 B6로 정의된 속성값(Name, Profile, Material 등)을 기준한 모델이 생성된다.

Beam class

<내부플레이트(마찰재)>

```
TSM.Beam B3 = new TSM.Beam();

double B3_Length = 400, B3_Heignt = 90, B3_Width = 5;

B3.GetReportProperty("LENGTH", ref B3_Length);
B3.GetReportProperty("HEIGHT", ref B3_Heignt);
B3.GetReportProperty("WIDTH", ref B3_Width);

CoordinateSystem B3Coordi = B3.GetCoordinateSystem();

B3.StartPoint = new T3D.Point(1435, -B1_FlangeThickness - 235, -B1_WebThickness / 2 - B3_Width / 2);
B3.EndPoint = new T3D.Point(1435 + 400, -B1_FlangeThickness - 235, -B1_WebThickness / 2 - B3_Width / 2

B3.Name = "DAMPER";
B3.Profile.ProfileString = "PL5*90";
B3.Material.MaterialString = "SS400";
B3.Class = "1";
B3.Finish = "DP_friction(CP1)";
B3.Position.Depth = Position.DepthEnum.MIDDLE;
B3.Position.DepthOffset = -0.00;
B3.Position.Plane = Position.PlaneEnum.RIGHT;
B3.Position.PlaneOffset = 0.00;
B3.Position.Rotation = Position.RotationEnum.FRONT;
B3.Position.RotationOffset = -0.00000;

TSM.Beam B4 = new TSM.Beam
    (new Point(1435, -B1_FlangeThickness - 235, B1_WebThickness / 2 + B3_Width / 2), new Point(1835, -B
B4.Name = "DAMPER";
B4.Profile.ProfileString = "PL5*90";
B4.Material.MaterialString = "SS400";
B4.Class = "1";
B4.Finish = "DP_friction(CP2)";
B4.Position.Depth = Position.DepthEnum.MIDDLE;
B4.Position.DepthOffset = -0.00;
B4.Position.Plane = Position.PlaneEnum.RIGHT;
B4.Position.PlaneOffset = 0.00;
B4.Position.Rotation = Position.RotationEnum.FRONT;
B4.Position.RotationOffset = -0.00000;

if (!B3.Insert())
    Console.WriteLine("B3 Insert failed!");
if (!B4.Insert())
    Console.WriteLine("B4 Insert failed!");
```



B3, B4 생성

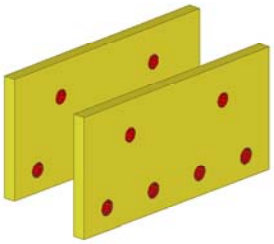
그림 3.10 내부 마찰재 입력데이터

dCollection @ chosun

<커버 플레이트>

Beam class

```
TSM.Beam B5 = new TSM.Beam();

double B5_Length = 400, B5_Heignt = 190, B5_Width = 20;

B5.GetReportProperty("LENGTH", ref B5_Length);
B5.GetReportProperty("HEIGHT", ref B5_Heignt);
B5.GetReportProperty("WIDTH", ref B5_Width);

CoordinateSystem B5Coordi = B5.GetCoordinateSystem();

B5.StartPoint = new T3D.Point(1435, -B1_FlangeThickness - 225, -B1_WebThickness / 2 - B3_Width - B5_Wi
B5.EndPoint = new T3D.Point(1435 + 400, -B1_FlangeThickness - 225, -B1_WebThickness / 2 - B3_Width - B5

B5.Profile.ProfileString = "PL20*190";
B5.Material.MaterialString = "SS400";
B5.Class = "6";
B5.Name = "DAMPER";
B5.Finish = "DP_CP3";
B5.Position.Depth = Position.DepthEnum.MIDDLE;
B5.Position.DepthOffset = -0.00;
B5.Position.Plane = Position.PlaneEnum.RIGHT;
B5.Position.PlaneOffset = 0.00;
B5.Position.Rotation = Position.RotationEnum.FRONT;
B5.Position.RotationOffset = -0.00000;

if (!B5.Insert())
    Console.WriteLine("B5 Insert failed!");

TSM.Beam B6 = new TSM.Beam();
B6.StartPoint = new T3D.Point(1435, -B1_FlangeThickness - 225, B1_WebThickness / 2 + B3_Width + B5_Wid
B6.EndPoint = new T3D.Point(1435 + 400, -B1_FlangeThickness - 225, B1_WebThickness / 2 + B3_Width + B5.

B6.Profile.ProfileString = "PL20*190";
B6.Material.MaterialString = "SS400";
B6.Class = "6";
B5.Name = "DAMPER";
B6.Finish = "DP_CP4";
B6.Position.Depth = Position.DepthEnum.MIDDLE;
B6.Position.DepthOffset = -0.00;
B6.Position.Plane = Position.PlaneEnum.RIGHT;
B6.Position.PlaneOffset = 0.00;
B6.Position.Rotation = Position.RotationEnum.FRONT;
B6.Position.RotationOffset = -0.00000;


if (!B5.Insert())
    Console.WriteLine("B5 Insert failed!");
if (!B6.Insert())
    Console.WriteLine("B6 Insert failed!");
```



B5, B6 생성

그림 3.11 커버 플레이트 입력데이터

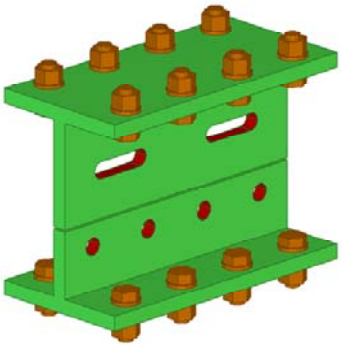
dCollection @ chosun

그림 3.12와 같이 파스너 컴포넌트로 분류되어 구성되어진 상·하부 부재 볼트에 관한 입력 데이터는 다음과 같다. 클래스(class)는 BoltArray로 선언하고, 모델의 객체는 bolt로 정의하여 상부와 하부의 볼트 생성을 위해 Insert()를 호출한다.

<상하부 볼트>

BoltArray class

```
BoltArray bolt = new BoltArray();

bolt.PartToBoltTo = B1;
bolt.PartToBeBolted = B1 as Beam;

bolt.FirstPosition = new Point(1435+50, -B1_FlangeThickness - 200, 0);
bolt.SecondPosition = new Point(1835, -B1_FlangeThickness - 200, 0);

bolt.BoltSize = 20;
bolt.BoltStandard = "H.T.B";
bolt.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt.Tolerance = 2;
bolt.CutLength = 100;
bolt.ExtraLength = 0;

bolt.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt.Position.Depth = Position.DepthEnum.MIDDLE;
bolt.Position.Plane = Position.PlaneEnum.MIDDLE;
bolt.Position.Rotation = Position.RotationEnum.TOP;

bolt.Bolt = true;
bolt.Washer1 = true;
bolt.Nut1 = true;

bolt.Hole1 = false;
bolt.Hole2 = false;
bolt.Hole3 = false;
bolt.Hole4 = false;

bolt.AddBoltDistX(100);
bolt.AddBoltDistX(100);
bolt.AddBoltDistX(100);
bolt.AddBoltDistY(140);

bolt.Insert();

bolt.PartToBoltTo = B2;
bolt.PartToBeBolted = B2 as Beam;
bolt.FirstPosition = new Point(1435+50, - 200-280, 0);
bolt.SecondPosition = new Point(1835, - 200-280, 0);

bolt.BoltSize = 20;
bolt.BoltStandard = "H.T.B";
bolt.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt.Tolerance = 2;
bolt.CutLength = 100;
bolt.ExtraLength = 0;

bolt.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt.Position.Depth = Position.DepthEnum.MIDDLE;
bolt.Position.Plane = Position.PlaneEnum.MIDDLE;
bolt.Position.Rotation = Position.RotationEnum.BELOW;
```



bolt 생성

그림 3.12 상·하부 볼트 입력데이터

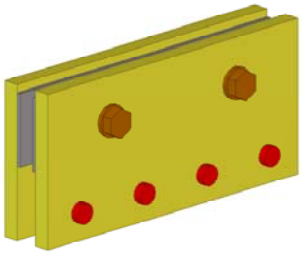
dCollection @ chosun

그림 3.13과 같이 볼트 파트로 상기 그림 3.12와 동일한 방법으로 클래스(class)는 BoltArray로 선언하고, 모델의 객체는 bolt2로 정의하여 중앙부의 인장 볼트도 Insert()로 호출하여 생성한다.

BoltArray class

<중앙부 인장 볼트>

```
BoltArray bolt2 = new BoltArray();

bolt2.PartToBoltTo = B6;
bolt2.PartToBeBolted = B5 as Beam;
bolt2.FirstPosition = new Point(1435, -B1_FlangeThickness - 200 - 80, 0);
bolt2.SecondPosition = new Point(1745, -B1_FlangeThickness - 200 - 80, 0);

bolt2.BoltSize = 20;
bolt2.BoltStandard = "H.T.B";
bolt2.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt2.Tolerance = 3;
bolt2.Length = 100;
bolt2.CutLength = 100;
bolt2.ExtraLength = 0;

bolt2.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt2.Position.Depth = Position.DepthEnum.MIDDLE;
bolt2.Position.Plane = Position.PlaneEnum.MIDDLE;
bolt2.Position.Rotation = Position.RotationEnum.FRONT;

bolt2.Bolt = true;
bolt2.Washer1 = true;
bolt2.Nut1 = true;

bolt2.Hole1 = false;
bolt2.Hole2 = false;
bolt2.Hole3 = false;
bolt2.Hole4 = false;

bolt2.AddBoltDistX(200);
bolt2.AddBoltDistY(0);
bolt2.StartPointOffset.Dx = 100.0;

bolt2.Insert();
```



bolt2 생성

그림 3.13 중앙부 인장 볼트 입력데이터

그림 3.14는 형식 변수 double 에 관한 내용이다. 부동 소수점 형식의 double 이 float에 비해 메모리를 두 배로 사용하지만, 그만큼 데이터의 손실이 적기 때문에 본 연구에서의 스마트그린 구조시스템 디테일 개발에서는 double 형식 변수를 사용하였다.

한 가지 예를 들어 내부 마찰재인 B3 객체를 정의할 때, Y축에 B1_FlangeThickness를 호출하고, Z축에 B1_WebThickness를 호출하여 위치 값을 계산한다.

dCollection @ chosun

float/double 형식 변수

Beam class

```
TSM.Beam B1 = new TSM.Beam();

double B1_Length = 400, B1_Heignt = 165, B1_Width = 200, B1_WebThickness = 20, B1_FlangeThickness = 20;

B1.GetReportProperty("LENGTH", ref B1_Length);
B1.GetReportProperty("HEIGHT", ref B1_Heignt);
B1.GetReportProperty("WIDTH", ref B1_Width);
B1.GetReportProperty("PROFILE.WEB_THICKNESS", ref B1_WebThickness);
B1.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref B1_FlangeThickness

CoordinateSystem B1Coordi = B1.GetCoordinateSystem();
```

double 형식변수를 사용하여 부재의 길이 높이 두께를 입력

```
TSM.Beam B3 = new TSM.Beam();

double B3_Length = 400, B3_Heignt = 90, B3_Width = 5;

B3.GetReportProperty("LENGTH", ref B3_Length);
B3.GetReportProperty("HEIGHT", ref B3_Heignt);
B3.GetReportProperty("WIDTH", ref B3_Width);

CoordinateSystem B3Coordi = B3.GetCoordinateSystem();
```

```
B3.StartPoint = new T3D.Point(1435, -B1_FlangeThickness - 235, -B1_WebThickness / 2 - B3_Width / 2);
B3.EndPoint = new T3D.Point(1435 + 400, -B1_FlangeThickness - 235, -B1_WebThickness / 2 - B3_Width / 2);

B3.Name = "PLATE";
B3.Profile.ProfileString = "PL5*90";
B3.Material.MaterialString = "SS400";
B3.Class = "40";
B3.Finish = "CP1";
B3.Position.Depth = Position.DepthEnum.MIDDLE;
B3.Position.DepthOffset = -0.00;
B3.Position.Plane = Position.PlaneEnum.RIGHT;
B3.Position.PlaneOffset = 0.00;
B3.Position.Rotation = Position.RotationEnum.FRONT;
B3.Position.RotationOffset = -0.00000;
```

그림 3.14 부동 소수점 형식변수 double의 데이터

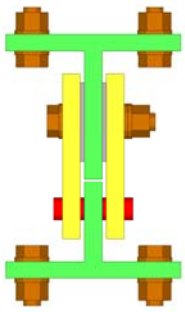
그림 3.15의 원형강봉은 분류체계에 의해 파스너 컴포넌트 파트로 분류되어져 있으며, Beam class를 선언하여 객체 B7~B10(4EA)을 지정하고, 객체의 생성을 위해 Insert()를 호출한다.

dCollection @ chosun

<원형강봉>

Beam class

```
Beam B7 = new Beam(new Point(1435 +50, -200 - 205, -50), new Point(1435 +50, -200 - 205, 50));
Beam B8 = new Beam(new Point(1435 + 150, -200 - 205, -50), new Point(1435 + 150, -200 - 205, 50));
Beam B9 = new Beam(new Point(1435 + 250, -200 - 205, -50), new Point(1435 + 250, -200 - 205, 50));
Beam B10 = new Beam(new Point(1435 + 350, -200 - 205, -50), new Point(1435 + 350, -200 - 205, 50));

B7.Name = "DAMPER";
B7.Profile.ProfileString = "D25";
B7.Material.MaterialString = "SS400";
B7.Class = "2";
B7.Finish = "DP_SteelBar 1";
B7.Position.Plane = Position.PlaneEnum.MIDDLE;
B7.Position.PlaneOffset = 0.00;
B7.Position.Rotation = Position.RotationEnum.TOP;
B7.Position.RotationOffset = 0.00;
B7.Position.Depth = Position.DepthEnum.MIDDLE;
B7.Position.DepthOffset = -0.00;

B8.Name = "DAMPER";
B8.Profile.ProfileString = "D25";
B8.Material.MaterialString = "SS400";
B8.Class = "2";
B8.Finish = "DP_SteelBar 2";
B8.Position.Plane = Position.PlaneEnum.MIDDLE;
B8.Position.PlaneOffset = 0.00;
B8.Position.Rotation = Position.RotationEnum.TOP;
B8.Position.RotationOffset = 0.00;
B8.Position.Depth = Position.DepthEnum.MIDDLE;
B8.Position.DepthOffset = -0.00;

B9.Name = "DAMPER";
B9.Profile.ProfileString = "D25";
B9.Material.MaterialString = "SS400";
B9.Class = "2";
B9.Finish = "DP_SteelBar 3";
B9.Position.Plane = Position.PlaneEnum.MIDDLE;
B9.Position.PlaneOffset = 0.00;
B9.Position.Rotation = Position.RotationEnum.TOP;
B9.Position.RotationOffset = 0.00;
B9.Position.Depth = Position.DepthEnum.MIDDLE;
B9.Position.DepthOffset = -0.00;

B10.Name = "DAMPER";
B10.Profile.ProfileString = "D25";
B10.Material.MaterialString = "SS400";
B10.Class = "2";
B10.Finish = "DP_SteelBar 4";
B10.Position.Plane = Position.PlaneEnum.MIDDLE;
B10.Position.PlaneOffset = 0.00;
B10.Position.Rotation = Position.RotationEnum.TOP;
B10.Position.RotationOffset = 0.00;
B10.Position.Depth = Position.DepthEnum.MIDDLE;
B10.Position.DepthOffset = -0.00;

B7.Insert();
B8.Insert();
B9.Insert();
B10.Insert();
```



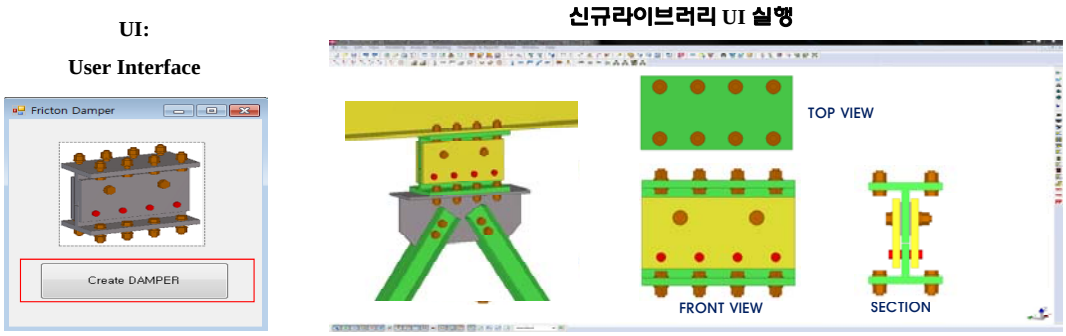
B7-B10 생성

그림 3.15 원형강봉의 입력데이터

dCollection @ chosun

3.2.1.3 전단형 마찰댐퍼의 APPLICATION

그림 3.16 과 같이 플레이트 컴포넌트와 파스너 컴포넌트가 분류 기준에 맞춰 개발되었으며 컴파일러(Compiler)를 통해 빌드 시켰다. Application을 통해 Tekla Structures 프로그램에 각각의 세부 디테일인 컴포넌트 입력 데이터의 조합으로 전체 마찰댐퍼가 완성이 되었으며 정확하게 탑재가 되고 있음을 확인하였다.



<Application> <Tekla structures 프로그램>

그림 3.16 전단형 마찰댐퍼의 APPLICATION

3.2.2 강재댐퍼 설계자동화 모듈 개발

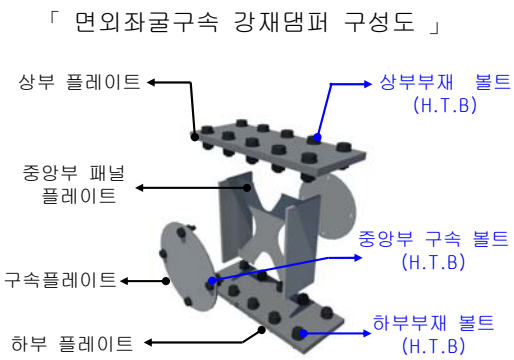
3.2.2.1 강재댐퍼의 정의

1. 면외좌굴구속 강재댐퍼

기존 강재댐퍼는 패널의 면외좌굴 후 소성화에 따른 소성변형 능력 저하로 인해 댐퍼로서의 기능을 효과적으로 발휘하지 못한다. 이를 보완하기 위하여 개발된 면외좌굴구속 강재댐퍼는 중앙부 패널플레이트의 양면에 구속플레이트로 패널을 구속함으로써 강재의 소성화에 따른 면외좌굴을 억제하여 수평으로 작용하는 외력을 흡수함으로써 스마트그린 구조시스템 구조물의 안정성을 향상시킨 댐퍼이다. 강재 플레이트가 소성

dCollection @ chosun

변형 후 다시 원래 형태로 돌아오는 과정에서 이력감쇠 저하현상이 발생하더라도, 강재 플레이트에 결합된 구속판 자체의 텐션에 의해 외력을 흡수하는 효과를 유지할 수 있다. 그림 3.17(a)는 면외좌굴구속 강재댐퍼의 구성도를 나타낸다. 면외좌굴구속 강재댐퍼는 그림 3.17(b)와 같이 구조물의 보에 결합되는 상부 플레이트와 하부 플레이트 사이에 수직으로 중앙부 패널 플레이트가 마련되고, 패널부 양면에서 중앙부 패널을 원형 플레이트가 구속시키는 구조로 구성된다.



(a) 구성요소의 분류

플레이트 컴포넌트 (5)	상부 플레이트 (1)
	중앙부 패널 플레이트 (1)
	하부 플레이트 (1)
	중앙부 구속 플레이트 (2)
파스너 컴포넌트 (4)	상하부 부재 H.T.B (2)
	중앙부 구속 H.T.B (2)

(b) 구성요소 분류체계

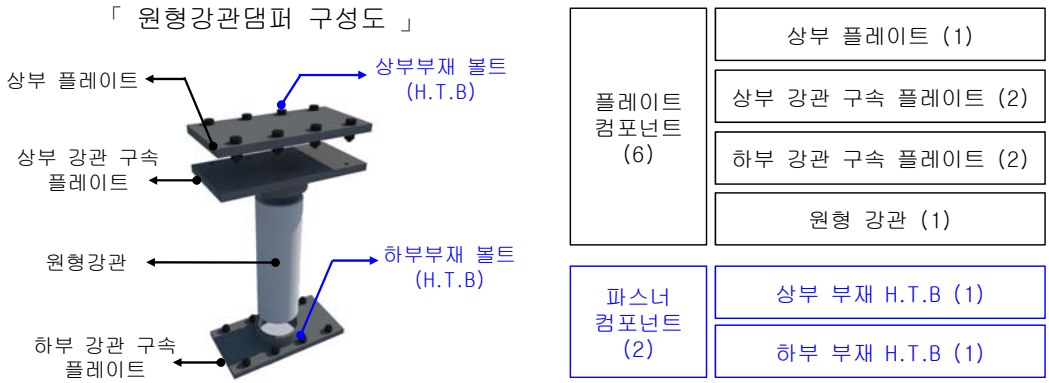
그림 3.17 면외좌굴구속 강재댐퍼의 구성요소

2. 원형강관댐퍼

원형강관댐퍼는 원형강관의 형상이 기하학적인 대칭으로 인하여 다방향 댐퍼로 사용이 가능한 특징이 있다. 또한 원형강관댐퍼는 설치와 교체가 용이하며, 간단한 구조로서 제작비용이 절감되어 저비용 고효율의 에너지 흡수장치로서의 이용이 가능하다는 이점이 있다. 원형강관댐퍼는 외경과 상하부 소켓 사이에 노출되는 원형강관 부재의 길이비가 1: $\sqrt{3}$일 때 가장 효율적으로 에너지를 흡수한다.[15]

그림 3.18(a)는 원형강관댐퍼의 구성도를 나타낸다. 원형강관댐퍼는 그림 3.18(b)와 같이 상부플레이트, 상부 강관구속 플레이트, 하부 강관 구속플레이트, 원형강관 및 상・하부 볼트로 구성되어 있다.

dCollection @ chosun



(a) 구성요소의 분류 (b) 구성요소 분류체계

그림 3.18 원형강관댐퍼의 구성요소

강재댐퍼의 구조상세(detail) 모델을 설계 자동화하기 위한 기반연구 프로세스는 그림 3.19와 같다. 강재댐퍼 설계자동화 모듈을 효율적으로 구축하기 위해 댐퍼의 구성도와 구성요소별 분류 기준 적립을 선행하였다. Visual C#의 개발을 위해 네임스페이스명을 전단형 마찰댐퍼로 정의하고, 클래스 선언을 통하여 각 컴포넌트 부재의 속성 및 기능을 Tekla Structures 시스템 DB와 매핑(mapping)하였다. C#에서 개발한 3D 정보화 건축 모델이 Tekla Structures에 정확하게 탑재가 되는지 검증을 한 후 최종 사용자 인터페이스(UI) 어플리케이션을 완성하였다.

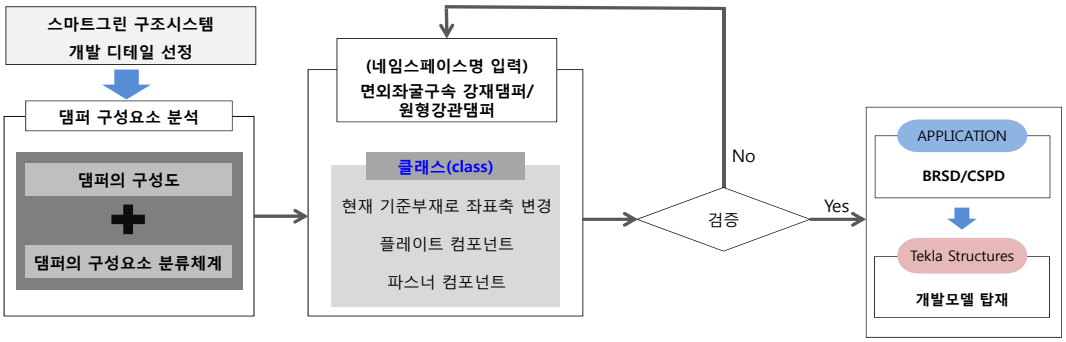


그림 3.19 강재댐퍼의 개발 프로세스

dCollection @ chosun

3.2.2.2 강재댐퍼의 Visual C# 입력데이터

강재댐퍼의 입력데이터에서는 속성과 기능을 부여한 Beam class, ContourPlate class 및 BoltArray class를 선언하고, 사용자가 각 오브젝트에 해당하는 객체명을 지정하여 디테일을 생성하였으며 부록 1과 부록 2에 그 내용을 나타내었다.

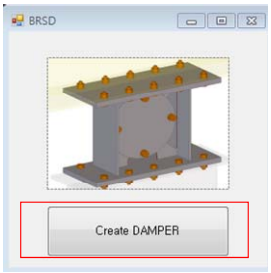
3.2.2.3 강재댐퍼의 APPLICATION

그림 3.20, 3.21과 같이 플레이트 컴포넌트와 파스너 컴포넌트가 분류 기준에 맞춰 개발 되었으며 컴파일러(Compiler)를 통해 빌드 시켰다. 면외좌굴구속 강재댐퍼와 원형강관댐퍼의 Application을 통해 Tekla Structures 프로그램에 각각의 세부 디테일인 컴포넌트의 입력 데이터의 조합으로 전체 강재댐퍼가 완성이 되었으며 정확하게 탑재가 될 수 있음을 확인하였다.

1. 면외좌굴구속 강재댐퍼

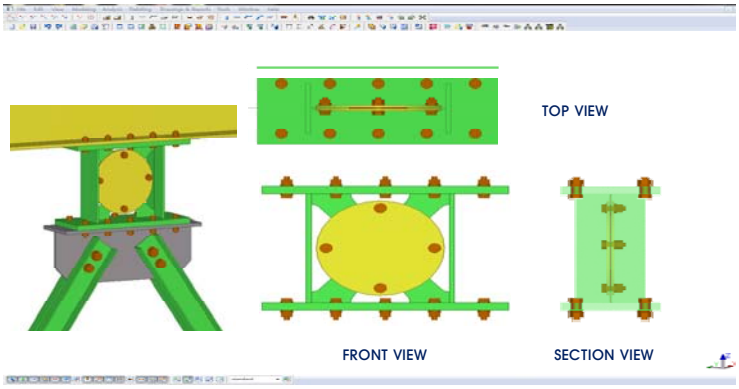
UI:

User Interface



<Application>

신규라이브러리 UI 실행



<Tekla structures 프로그램>

그림 3.20 면외좌굴구속 강재댐퍼의 APPLICATION

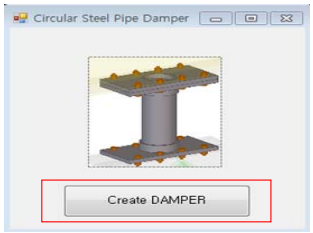
dCollection @ chosun

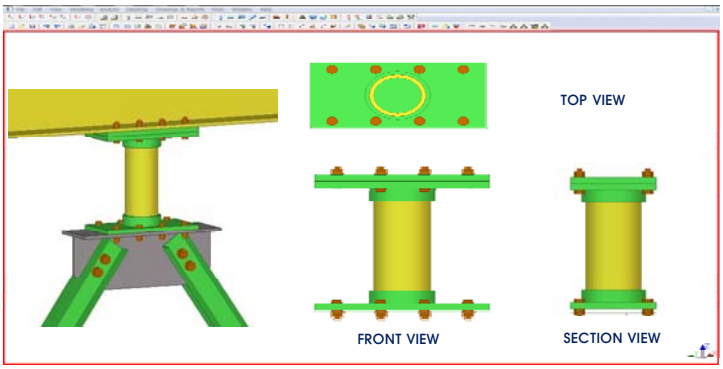
2. 원형강관댐퍼

신규라이브러리 UI 실행

UI:

User Interface





<Application> **<Tekla structures 프로그램>**

그림 3.21 원형강관댐퍼의 APPLICATION

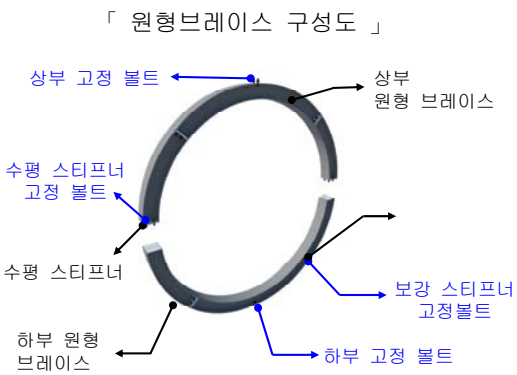
3.3 수평저항장치 설계자동화 모듈 개발

3.3.1 원형 브레이스 설계자동화 모듈 개발

3.3.1.1 원형 브레이스의 정의

스마트그린 구조시스템의 접합부는 Flexible한 접합부를 구성하기 때문에, 외부에서 수평 하중이 구조물에 작용 하였을 때 수평내력이 부족한 단점이 있다. 이러한 구조물의 횡방향 강성을 확보하기 위해 브레이스를 설치한다. 하지만 지진과 같은 외력에 의한 하중은 반복하중이 작용하게 되는데 이 때, K형 및 X형 브레이스의 경우 압축좌굴을 일으키게 되며, 브레이스가 좌굴하고 난 후에는 기둥과 보에도 변형이 발생하게 되는 결함을 가지고 있다. 하지만, 원형 브레이스는 종래의 K형 및 X형 브레이스 설치 구조에 비해 구조물에 작용하는 외력에 대한 보강강도를 향상시켜 구조물의 안전성을 높일 수 있는 특징이 있다. 그림 3.22(a)는 원형 브레이스의 구성도를 나타내고, 구성요소 분류체계를 그림 3.22(b)에 나타내었다. 원형 브레이스는 시공 편의성을 위해 2개의 Beam과 보강 스티프너 볼트로 구성된다.

dCollection @ chosun



(a) 구성요소의 분류

플레이트 컴포넌트 (26)	상부 원형 브레이스 (1)
	하부 원형 브레이스 (1)
	보강 스티프너 (16)
	수평 스티프너 (8)
파스너 컴포넌트 (8)	보강 스티프너 고정 볼트 (4)
	수평 스티프너 고정 볼트 (2)
	상, 하부 고정 볼트 (2)

(b) 구성요소 분류체계

그림 3.22 원형브레이스의 구성요소

원형 브레이스의 구조상세 모델에 대하여 설계 자동화 모듈을 구현하기 위한 기반연구 프로세스는 그림 3.23과 같다. 원형 브레이스 설계 자동화 모듈을 효율적으로 구축하기 위한 원형 브레이스의 구성도와 구성요소 분류체계를 선행하였다. Visual C#의 개발을 위해 네임스페이스명을 원형 브레이스로 정의하고, 클래스(class)를 선언하였다. 개발된 댐퍼의 검증 및 Tekla Structures에 개발 모델 탑재를 위해 최종 사용자 인터페이스(UI) 어플리케이션 프로세스에 기반하여 진행하였다.

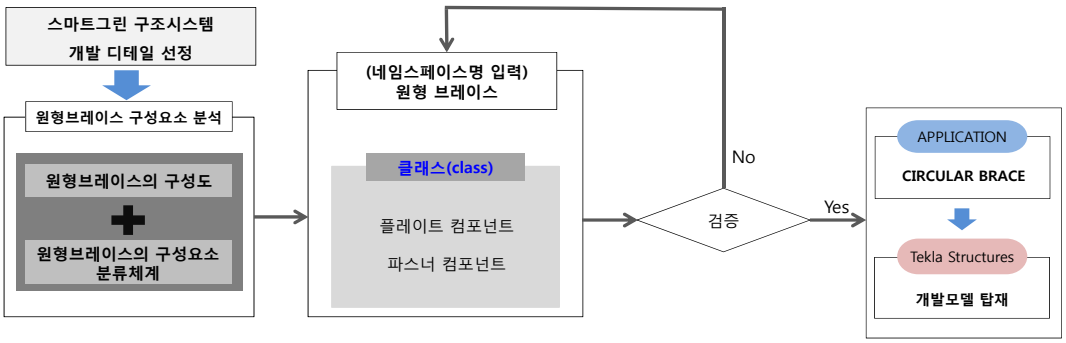


그림 3.23 원형 브레이스의 개발 프로세스

dCollection @ chosun

3.3.1.2 원형 브레이스의 Visual C# 입력데이터

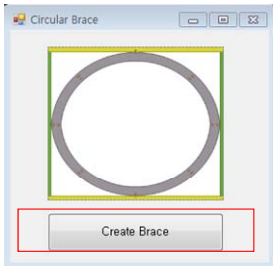
그림 3.22(b)의 컴포넌트 구성요소별로 Beam class, ContourPlate class 및 BoltArray class의 오브젝트를 선언하였다. 입력데이터의 세부 내역은 부록 3에 나타내었다.

3.3.1.3 원형 브레이스의 APPLICATION

그림 3.24와 같이 원형브레이스는 플레이트 컴포넌트와 파스너 컴포넌트가 분류 기준에 맞춰 개발 되었으며 컴파일러(Compiler)를 통해 빌드 시켰다. 원형 브레이스의 Application을 통해 Tekla Structures 프로그램에 각각의 세부 디테일인 컴포넌트의 입력 데이터의 조합으로 탑재가 될 수 있음을 확인하였다.

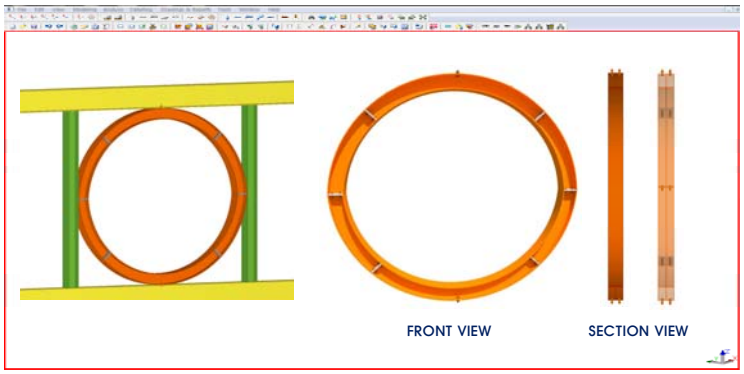
UI:

User Interface



<Application>

신규라이브러리 UI 실행



<Tekla structures 프로그램>

그림 3.24 원형 브레이스의 APPLICATION

dCollection @ chosun

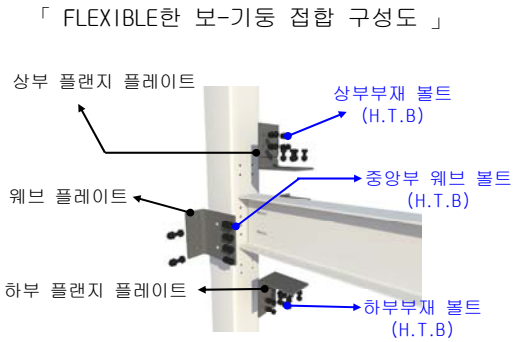
3.4 신개념 구조 접합 설계자동화 모듈 개발

3.4.1 Flexible한 보-기둥 접합부 설계자동화 모듈 개발

3.4.1.1 Flexible한 보-기둥 접합부의 정의

보-기둥 접합부를 강접합으로 구성하고 접합부 강성을 크게 설계하면, 지진에 의한 진동이나 강풍 등에 의한 외력이 발생했을 때, 주 구조체인 기둥 및 보가 외력에 의해 손상된다. 스마트그린 구조시스템의 보-기둥 접합부의 경우 Flexible한 접합부는 기본 반강접 접합부 중 Top and Seat Angle 접합부를 기초로 하였다. 기둥 및 보에 손상을 방지를 위해 Angle에 노치를 만들고, 구조물에 수평하중이 작용하였을 때 Angle에 응력을 집중시켜 기둥과 보에는 손상이 가해지지 않도록 하였다.

그림 3.25(a)에는 Flexible한 보-기둥 접합부의 구성도를 나타내고, 구성요소 분류체계를 그림 3.25(b)에 나타내었다. Flexible한 보-기둥 접합부는 상부 플랜지 플레이트, 하부 플랜지 플레이트, 웨브 플레이트, 볼트로 구성이 이루어지며, 상·하부 플랜지 플레이트에 노치를 만들어 응력을 집중 시킨다.



(a) 구성요소의 분류

플레이트 컴포넌트 (4)	상부 플랜지 플레이트 (1)
	하부 플랜지 플레이트 (1)
	웨브 플레이트 (2)
파스너 컴포넌트 (8)	상부부재 H.T.B (2)
	하부부재 H.T.B (2)
	중앙부 웨브 H.T.B (4)

(b) 구성요소 분류체계

그림 3.25 Flexible한 보-기둥 접합부의 구성요소

dCollection @ chosun

신개념 접합부는 구성요소에 의거하여 그림 3.26과 같은 Flexible한 보-기둥 접합부 개발 프로세스에 의해 개발을 진행하였다. 접합부 설계 자동화 모듈을 효율적으로 구축하기 위한 구성요소 분석의 구성도와 구성요소 분류체계를 선행하였다. Visual C#의 개발을 위해 네임스페이스명을 Flexible한 보-기둥 접합부로 정의하고, 클래스를 선언하였다. 개발된 댐퍼의 검증 및 Tekla Structures에 개발 모델 탑재를 위해 최종 UI어플리케이션 프로세스에 기반하여 진행하였다.

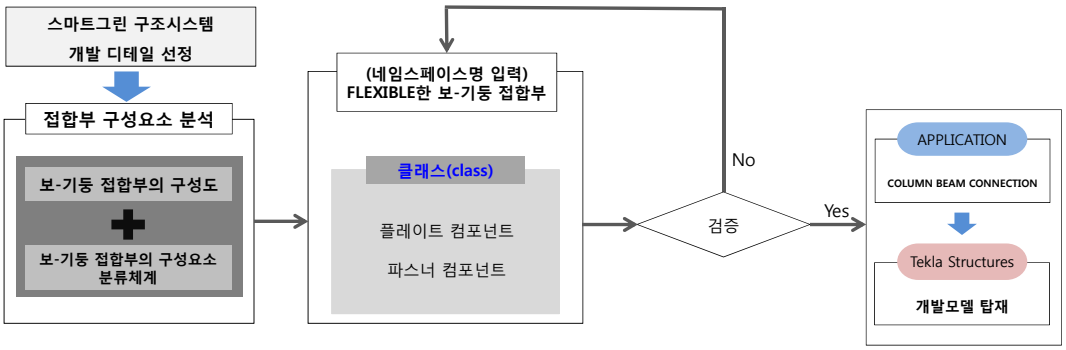


그림 3.26 Flexible한 보-기둥 접합부의 개발 프로세스

3.4.1.2 Flexible한 보-기둥 접합부의 Visual C# 입력데이터

그림 3.25(b)의 컴포넌트 구성요소별로 Beam class, ContourPlate class 및 BoltArray class로 오브젝트를 선언하였다. 입력데이터의 세부 내역은 부록 5에 나타내었다.

3.4.1.3 Flexible한 보-기둥 접합부의 APPLICATION

그림 3.27과 같이 Flexible한 보-기둥 접합부는 플레이트 컴포넌트와 파스너 컴포넌트가 분류 기준에 맞춰 개발 되었으며 컴파일러(Compiler)를 통해 빌드 시켰다.

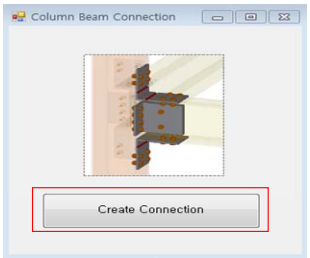
Flexible한 보-기둥 접합부의 Application을 통해 Tekla Structures 프로그램에 각각의 세부 디테일인 컴포넌트의 입력 데이터의 조합으로 탑재가 될 수 있음을 확인하였다.

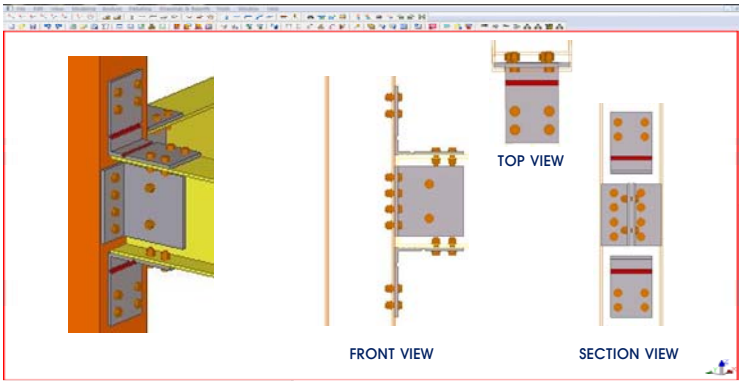
dCollection @ chosun

신규라이브러리 UI 실행

UI:

User Interface





<Application>

<Tekla structures 프로그램>

그림 3.27 Flexible한 보-기둥 접합부의 APPLICAION

3.4.2 조립・해체가 자유로운 주각부 설계자동화 모듈 개발

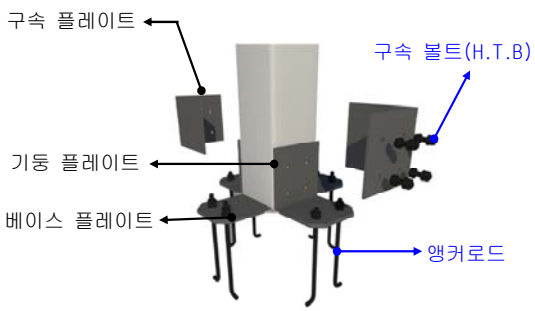
3.4.2.1 조립・해체가 자유로운 주각부의 정의

일반적으로 주각부는 기둥으로 전달되는 하중을 기초에 전달하는 역할을 하는 것으로 하중을 기초로 균등하게 분산시키는 역할을 한다. 주각부는 기둥과 베이스 플레이트 용접을 통해 접합을 하고, 베이스 플레이트는 앵커볼트를 정착하여 콘크리트 기초와 일체화시켜 시공한다. 조립・해체가 자유로운 주각부의 경우 기둥과 베이스플레이트는 볼트접합을 통해 해체가 자유로운 형태로 구성하였고, 수평하중의 외력이 발생하였을 때 베이스 플레이트에 손상을 집중시켜 기둥의 재사용을 가능하게 하였다.

그림 3.28(a)에서는 Flexible한 주각부 구성도를 나타내고, 구성요소 분류체계를 그림 3.28(b)에 나타내었다. 조립・해체가 자유로운 주각부는 기둥 구속 플레이트, 기둥플레이트, 베이스플레이트 및 앵커로드로 구성이 이루어진다.

dCollection @ chosun

「 조립해체가 자유로운 주각부 구성도 」



(a) 구성요소의 분류

플레이트 컴포넌트 (12)	기둥 구속 플레이트 (2)
	기둥 플레이트 (2)
	베이스 플레이트 (8)
파스너 컴포넌트 (10)	구속 H.T.B (2)
	앵커로드 (8)

(b) 구성요소 분류체계

그림 3.28 조립・해체가 자유로운 주각부의 구성요소

주각 부는 구성요소에 의거하여 그림 3.29와 같은 주각부 개발 프로세스에 의해 개발을 진행하였다. 주각부 설계 자동화 모듈을 효율적으로 구축하기 위한 구성도와 구성요소 분류체계를 선행하였다. Visual C#의 개발을 위해 네임스페이스명을 조립・해체가 자유로운 주각부로 정의하고, 클래스(class)를 선언하였다. 개발된 댐퍼의 검증 및 Tekla Structures에 개발 모델 탑재를 위해 최종 사용자 인터페이스(UI) 어플리케이션 프로세스에 기반 하여 진행하였다.

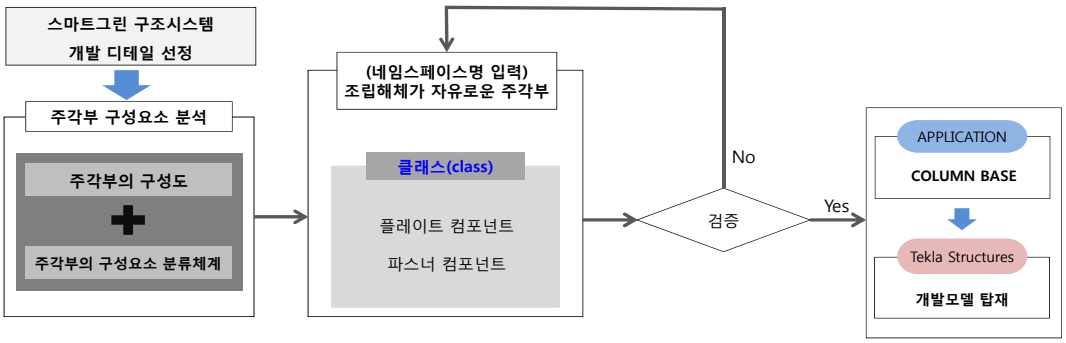


그림 3.29 조립・해체가 자유로운 주각부의 개발 프로세스

dCollection @ chosun

3.4.2.2 조립・해체가 자유로운 주각부의 Visual C# 입력데이터

그림 3.28(b)의 컴포넌트 구성요소별로 Beam class, ContourPlate class 및 BoltArray class의 오브젝트를 선언하였다. 입력데이터의 세부 내역은 부록 4에 나타내었다.

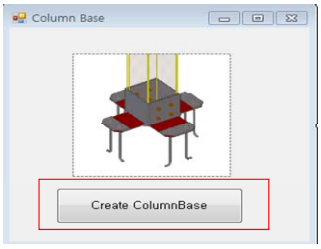
3.4.2.3 조립・해체가 자유로운 주각부의 APPLICATION

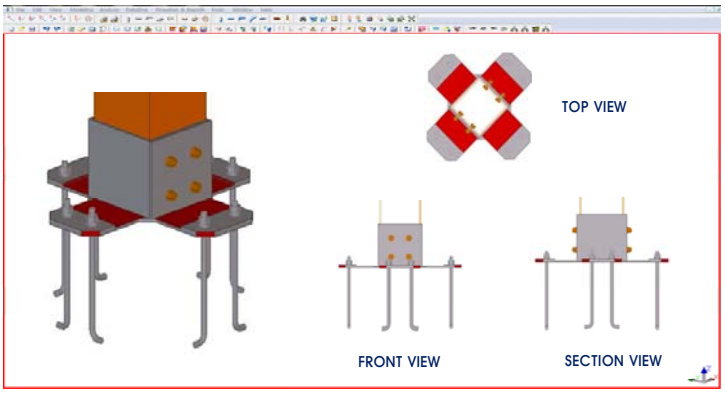
그림 3.30과 같이 조립・해체가 자유로운 주각부는 플레이트 컴포넌트와 파스너 컴포넌트가 분류 기준에 맞춰 개발 되었으며 컴파일러(Compiler)를 통해 빌드 시켰다. 조립・해체가 자유로운 주각부의 Application을 통해 Tekla Structures 프로그램에 각각의 세부 디테일인 컴포넌트의 입력 데이터의 조합으로 탑재가 될 수 있음을 확인하였다.

신규라이브러리 UI 실행

UI:

User Interface





<Application>

<Tekla structures 프로그램>

그림 3.30 조립・해체가 자유로운 주각부의 APPLICATION

dCollection @ chosun

3.5 소결

본 장에서는 기존 라이브러리에서 제공하지 않는 특수구조형상을 구축하기 위해 Tekla OpenAPI를 통한 설계 모듈 개발을 수행 하였으며, 다음과 같은 결과를 얻었다.

1. 스마트그린 구조시스템의 특수구조형상과 같은 디테일은 기존 라이브러리에서 제공하지 않아 모델링이 제한적이다. 하지만, C#을 이용한 OpenAPI를 통해 신규 라이브러리를 개발함으로써 특수구조형상 모델링의 한계를 극복하였다.

2. 개발언어인 C# 프로그램 체계 분석을 선행하고, 특수구조형상 설계자동화 모듈 개발을 통해 어플리케이션을 완성하였다. 개발된 어플리케이션을 이용하여 Tekla Structures 프로그램내에 원하는 특수구조형상의 모델 탑재가 완료됨을 확인하였다.

3. 특수구조형상에 대한 설계 자동화 모듈을 개발하기 위한 모델링 프로세스 개발과 각 요소별 C#코딩 방법을 제시함으로써, 다양하고 복합화 되어 가는 현재 건축시스템에 BIM이 효율적으로 표현할 수 있는 방안을 제시 하였다.

dCollection @ chosun

제4장 BIM 객체정보 모델 데이터 활용

4.1 BIM 데이터 활용을 위한 스마트그린 구조시스템 3차원 건축 정보 모델링

BIM은 가상공간에서 기획, 설계, 엔지니어링, 시공, 유지관리, 폐기단계의 건축물 Life Cycle을 모델링하여 다차원 부재정보를 관리하는 과정을 나타내며 크게 모델링 기술과 데이터 활용기술로 나눌 수 있다. BIM을 적용하면 3차원 모델로부터 물량을 자동으로 추출할 수 있으며, 건물의 디자인 형태, 건축 공법, 자재 등이 변경되었을 때 변경된 물량을 바로 알아낼 수 있고 신속하게 대응할 수 있다. 하지만 BIM 데이터를 활용하기 위해서는 건물의 부재들이 올바른 물량산출이 가능하도록 BIM 데이터가 정확히 모델링이 되어야 한다.

기존 라이브러리에서 제공하지 않는 특수구조형상을 3장에서 개발하였고, 사용자가 직접 개발한 설계 자동화 모듈을 이용하여 그림 4.1과 같이 4층 규모의 강구조 Frame에 장착함으로써 3차원 모델 객체에 건축정보를 포함한 BIM기반 스마트그린 구조시스템 구조물을 완성하였다.

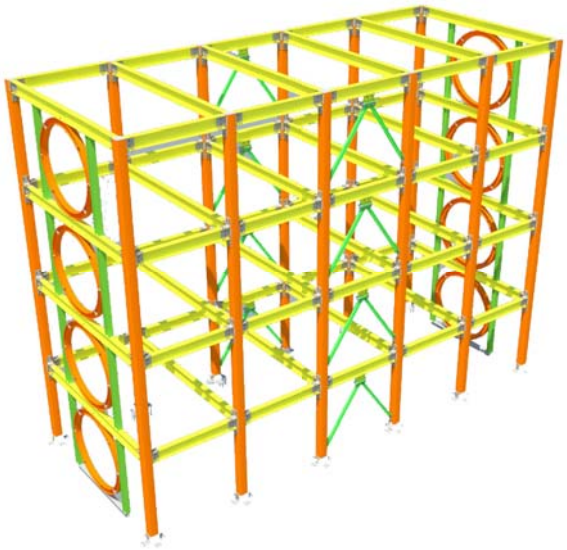


그림 4.1 스마트그린 구조시스템 3차원 건축 정보 모델링

dCollection @ chosun

4.2 BIM기반 스마트그린 구조시스템의 부재 정보 관리

4.2.1 BIM기반 스마트그린 구조시스템의 물량 산출

일반적으로 기존의 물량산출 프로세스는 2D 기반의 구조물 설계를 완료한 후, 2D 설계도서를 기반으로 해당 부재의 물량을 산출하여 표준품셈을 적용시킨다. 2D 기반의 물량산출 프로세스는 많은 인적, 물적 자원을 낭비하게 되며, 작업자의 업무 능력에 따른 휴먼에러가 발생할 수 있다. 또한 설계도면을 작성중이거나 현장시공 중에 설계변경을 요구하는 불가피한 경우, 다시 물량산출 작업을 실시해야 하며 설계도서도 개별적으로 모두 수정해야 하는 번거로움이 발생한다.

2D 기반의 물량산출과 대비하여 BIM 데이터를 활용하면, 구체적이고 체계적인 물량데이터의 생성과 집계가 가능하다. BIM을 활용한 설계진행과 동시에 3D 정보화 건축모델링이 완성되고, 구조물의 계획단계에서부터 구조체 물량데이터가 체계적으로 축적되므로 부분 및 전체 물량 정보를 정량적으로 파악하기 쉽다. 또한 설계 변경이 적용될 경우, 3D 모델에 객체 변경 정보가 즉시 반영되므로 물량정보 데이터도 동시에 변경되어 물량 자동산출이 가능하다.

본 장에서는 4.1절에 완성한 스마트그린 구조시스템의 3D 모델링을 기반으로 기둥, 보, 주각부, 보-기둥 접합부, 마찰댐퍼, K형 브레이스, 원형 브레이스, 볼트 등의 각 요소별 물량과 구조물 전체의 물량을 산출하였다.

스마트그린 구조시스템의 전체 물량을 그림 4.2에 나타내었다. 구조물의 전체 중량은 50.2456 ton으로 산출되었으며, 각 부재별 길이, 재질, 부재 치수 등의 정보 확인도 가능하였다. 더욱이 볼트의 사이즈 및 길이별 소요 개수 등도 확인이 가능하였다.

그림 4.3은 스마트그린 구조시스템의 전체 물량 중 보의 물량만 개별적으로 산출하여 나타낸 것이다. 사용자가 원하는 부재 부분의 물량 확인이 가능하며, 보의 전체 중량, 재질뿐만 아니라 보의 길이 및 모델링 되어진 보의 전체 개수 등 원하는 부재 정보를 쉽게 자동 산출할 수 있다. 기둥, 접합부 등에 대한 물량 산출 결과는 부록 6에 첨부하였다.

한편 3차원 건축 정보를 포함하고 있는 객체들의 조합으로 전체 모델이 완성되었으며, BIM을 활용한 부재의 요소별 정보(부재 명칭, 프로파일, 재질, 길이, 수량, 중량 등)를 토대로 Template을 작성하고, 물량 정보 데이터를 수정할 수 있다.

dCollection @ chosun

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)	
AN1	D22	519.8	96	SS400	0.0	0.0	1.5	144.0	
				Sub Total:	0.0			144.0	
DP5	D25	100.0	32	SS400	0.0	0.0	0.4	12.8	
				Sub Total:	0.0			12.8	
BR_pol	H200*200*8*12	4540.1	7	SS400	37.1	49.9	226.5	1585.5	
BR_pol	H200*200*8*12	4540.1	7	SS400	37.1	49.9	226.5	1585.5	
MB13	H200*200*8*12	3800.0	2	SS400	8.8	49.9	189.6	379.2	
MC5	H200*200*8*12	3200.0	16	SS400	59.2	49.9	159.7	2555.2	
P24	H200*200*8*12	4540.1	2	SS400	10.6	49.9	226.5	453.0	
				Sub Total:	152.8			6558.4	
MB2	H400*200*8*13	3270.0	6	SS400	30.6	66.0	215.8	1294.8	보
MB4	H400*200*8*13	6870.0	2	SS400	21.4	66.0	453.4	906.8	
MB5	H400*200*8*13	3270.0	24	SS400	122.4	66.0	215.8	5179.2	
MB6	H400*200*8*13	6870.0	12	SS400	128.4	66.0	453.4	5440.8	
MB7	H400*200*8*13	3270.0	2	SS400	10.2	66.0	215.8	431.6	
MB8	H400*200*8*13	3270.0	8	SS400	40.8	66.0	215.8	1726.4	
MB9	H400*200*8*13	6870.0	4	SS400	42.8	66.0	453.4	1813.6	
MB11	H400*200*8*13	6870.0	6	SS400	64.2	66.0	453.4	2720.4	
				Sub Total:	460.8			19513.6	
NUT1	HXGON44-44	22.0	96	SS400	0.0	0.0	0.2	19.2	
				Sub Total:	0.0			19.2	
PL5	L140*300*15	300.0	80	SS400	24.0	0.0	15.0	1200.0	
PL6	L140*300*15	300.0	48	SS400	14.4	0.0	15.0	720.0	
PL7	L140*300*15	300.0	48	SS400	14.4	0.0	15.0	720.0	
PL8	L140*300*15	300.0	80	SS400	24.0	0.0	15.0	1200.0	
				Sub Total:	76.8			3840.0	
PL1	L300*300*15	200.0	60	SS400	18.0	0.0	13.8	828.0	
PL2	L300*300*15	200.0	36	SS400	10.8	0.0	13.8	496.8	
PL3	L300*300*15	200.0	80	SS400	24.0	0.0	13.8	1104.0	
PL4	L300*300*15	200.0	48	SS400	14.4	0.0	13.8	662.4	
				Sub Total:	67.2			3091.2	
PL18	L330*310*10	300.0	12	SS400	4.8	0.0	14.8	177.6	
				Sub Total:	4.8			177.6	
PL19	L340*320*10	300.0	12	SS400	4.8	0.0	15.3	183.6	
				Sub Total:	4.8			183.6	
WS1	PD44*11	11.0	96	SS400	0.0	0.0	0.1	9.6	
				Sub Total:	0.0			9.6	
DP2	T112-20-20-200	400.0	8	SS400	2.4	0.0	18.3	146.4	
				Sub Total:	2.4			146.4	
DP4	T165-20-20-200	400.0	8	SS400	2.4	0.0	21.7	173.6	
				Sub Total:	2.4			173.6	
BR1	U100*50*5	2686.5	32	SS400	32.0	0.0	20.0	640.0	
				Sub Total:	32.0			640.0	
MC2	[]300*300*10	14400.0	8	SS400	138.4	0.0	1311.3	10490.4	기둥
MC3	[]300*300*10	14400.0	2	SS400	34.6	0.0	1311.3	2622.6	
MC4	[]300*300*10	14400.0	2	SS400	34.6	0.0	1311.3	2622.6	
				Sub Total:	207.6			15735.6	
				Total :	1011.6			50245.6	

그림 4.2 스마트그린 구조시스템 전체 물량 산출표

dCollection @ chosun

TeklaStructures PART SUMMARY LIST Page: 1
Project name :Smart Green Structur Phase:
Date 19.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
MB2	H400*200*8*13	3270.0	6	SS400	30.6	66.0	215.8	1294.8
MB4	H400*200*8*13	6870.0	2	SS400	21.4	66.0	453.4	906.8
MB5	H400*200*8*13	3270.0	24	SS400	122.4	66.0	215.8	5179.2
MB6	H400*200*8*13	6870.0	12	SS400	128.4	66.0	453.4	5440.8
MB7	H400*200*8*13	3270.0	2	SS400	10.2	66.0	215.8	431.6
MB8	H400*200*8*13	3270.0	8	SS400	40.8	66.0	215.8	1726.4
MB9	H400*200*8*13	6870.0	4	SS400	42.8	66.0	453.4	1813.6
MB11	H400*200*8*13	6870.0	6	SS400	64.2	66.0	453.4	2720.4
				Sub Total:	460.8			19513.6
				Total :	460.8			19513.6

그림 4.3 스마트그린 구조시스템 보 물량 산출표

4.2.2 기존 구조시스템과 스마트그린 구조시스템의 물량 비교

일반적으로 모든 건축물은 생산된 뒤 일정 기간이 지나면 구조물 전체가 해체가 되거나 폐기가 된다. 강재는 재순환성이 매우 높은 건축재료 임에도 불구하고 생산, 유지관리, 해체단계 후에 보수/재생 공장을 거쳐 다시 건축물의 원재료로 사용되는 비율은 현재 20% 미만이다. 스마트그린 구조시스템은 이러한 과정 중에 보수/재생 공장을 거치지 않고 해체 현장에서 바로 새로운 현장의 재료로 사용되는 경우로써, 처음 설계단계에서부터 해체를 고려한 신개념 구조시스템이라 할 수 있다.

2-3개월마다 조립·해체가 빈번하게 일어나는 모델하우스를 대상으로 본 4.2.2절의 연구를 진행하였다. 기존 공법으로 시공되어진 구조시스템의 경우 강접합으로 구성하였다. 스마트그린 구조시스템의 경우 Flexible한 접합부를 구성하고, 접합부에 손상을 집중시켰다. Flexible한 접합부로 구성되어 수평하중에 취약한 단점을 보완하기 위해, 수평저항장치 및 에너지 흡수장치를 장착하여 수평저항성능을 확보하고, 구조물의 안전성을 향상시켰다. 스마트그린 구조시스템의 경우 기존 구조시스템에서 사용되지 않는 브레이스와 댐퍼의 물량이 추가되었음을 확인할 수 있다. 조립·해체가 자유로운 스마트그린 구조시스템의 경우 시공 초기 투입 물량이 높게 나타났으나. 해체 후 재사용성은 70%정도로 나타났다.

주요 강재의 재사용이 가능하도록 구성된 스마트그린 구조시스템의 가치를 확인하기

dCollection @ chosun

위해 4.2.1절에서 산출된 스마트그린 구조시스템 물량과 기존 구조시스템의 물량을 비교·평가 하였다. 기존 구조시스템 모델링의 경우 스마트그린 구조시스템과 동일한 규모로 모델하우스를 선정하고, 기둥과 보 사이즈를 같게 설계하였다.

스마트그린 구조시스템과 기존 구조시스템의 부재별 물량을 표 4.1에 나타내었다. 초기 물량의 경우 기존 구조시스템 42,729kg에 대비하여 스마트그린 구조시스템이 50,245kg으로 스마트그린 구조시스템이 더 높게 나타났다. 스마트그린 구조시스템은 주 구조체인 기둥과 보만 재사용하고, 접합 및 추가 디테일 부재는 폐기하는 것으로 가정하였다. 이에 추가로, 그림 4.4와 같이 5개 현장의 모델하우스가 1세대로 지정이 되고, 매 3개월마다 조립과 해체가 4세대에 걸쳐 이루어진다고 가정하였다. 모델하우스는 전체 1년의 기간동안 5개 현장이 한 그룹으로 4번의 조립 단계와 3번의 해체 단계로 이어지며, 이 과정 중 부재가 다음 세대의 현장으로 자유롭게 이동하는 재사용 부재 순환 실적에 따른 결과를 그림 4.5에 나타내었다. 스마트그린 구조시스템을 적용한 모델하우스의 경우 초기 물량이 기존 구조시스템을 적용한 모델하우스보다 높게 나타났지만 2세대 현장부터 이전세대의 주 구조체(기둥, 보)가 재사용되어진 결과, 강재의 누적 소요 물량이 기존 구조시스템을 적용한 경우보다 낮게 나타났다.

스마트그린 구조시스템을 적용한 경우의 효율성을 확인하기 위해서는 정밀한 시나리오를 적용하여 검증하여야 할 것으로 판단되지만, 본 논문에서는 BIM 데이터를 활용하는 작업방법을 제안하고자 간단한 시나리오를 통해 비교·분석하였다.

표 4.1 기존 구조시스템 및 스마트그린 구조시스템 물량 비교

재사용 여부	기존 구조	MEMBER	스마트그린 구조	재사용 여부
X	15,718kg	기둥	15,736kg	0
X	19,513kg	보	19,514kg	0
X	482kg	주각부	534kg	X
X	7,016kg	보-기둥 접합부	6,931kg	X
.	.	마찰댐퍼+K형 브레이스	972kg	X
.	.	원형 브레이스	6558kg	X
	42,729kg	총물량	50,245kg	

dCollection @ chosun

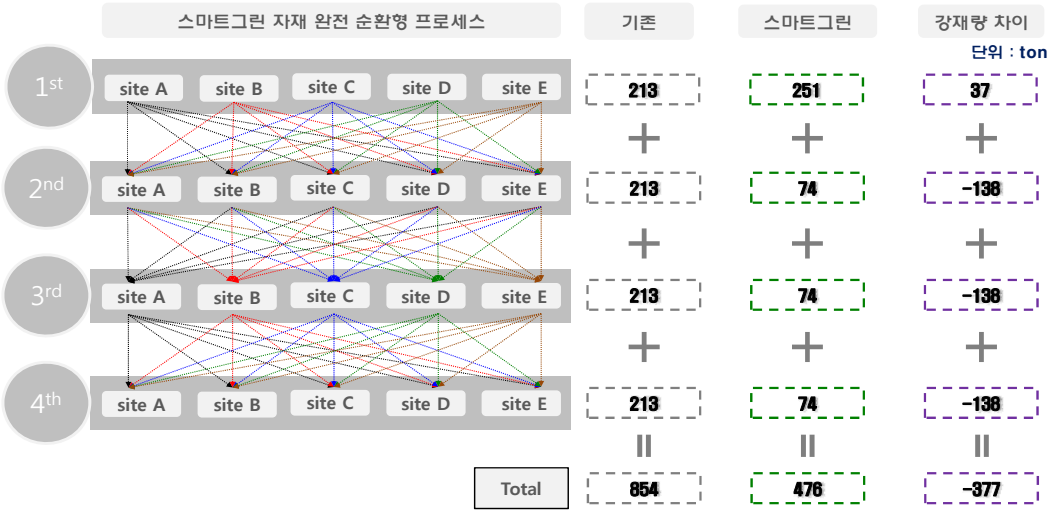


그림 4.4 세대 단위별 철골 자재 물량 비교

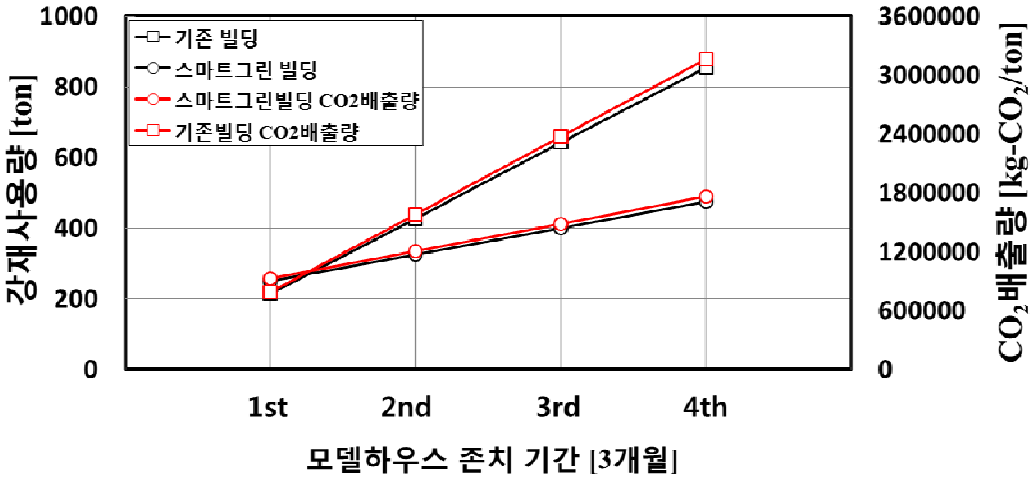


그림 4.5 모델하우스 강재의 누적 사용량 비교

dCollection @ chosun

4.3 BIM기반 스마트그린 구조시스템의 도면 출력

4.3.1 시스템상의 도면 출력 구성

스마트그린 구조시스템의 3D 객체 정보를 담고 있는 도면은 물량산출 정보와 함께 중요한 의사소통의 수단이 된다. 이와 함께 BIM을 이용하여 광범위한 정보관리가 용이해지고 정보의 통합으로 의사소통이 수월해 지고 있다. BIM의 도입으로 건축 설계도서 작성에서는 기존 2D 도면 작업 문제점으로 지적되었던 분절된 업무 구조를 개선시킬 수 있다. 특히, 건설 현장에서 수시로 발생하는 설계변경에 대하여 물량 변경과 동시에 도면의 수정 또한 자동으로 변경 가능하다.

본 연구에서의 스마트그린 구조시스템은 강구조체로써 접합 상세(detail) 도면을 생성해야 한다. 한편, 앞의 3장에서 이미 개발이 완료된 스마트그린 구조시스템의 상세 도면을 생성하기 위해서는 템플릿(Template)을 작성해야 한다. 도면에 추가적으로 사용자에게 필요한 정보를 표현하기 위해서는 템플릿을 헤더(Header), 페이지헤더(Page Header), 로우(Row)등으로 구성해야 한다. 헤더에는 도면 타이틀, 도면번호 및 로우의 필드항목 등을 명시한다. 로우에는 각 필드의 변수 속성값을 구성하고, 필드의 변수는 각 부재의 속성값인 부재 프로파일, 수량, 길이, 중량 등을 명시한다. 명시된 속성변수는 로우의 열마다 정리가 되며 표현하고자 하는 부재에 적용이 되어 도면에 나타나는 디테일에 따른 물량이 자동으로 작성된다. 기존 2D 작업에서는 사용자가 입력하지 않고서는 부재 정보를 표현할 수 없었지만, BIM 기반 설계도서에서는 템플릿의 속성변수값을 추가하는 작업만으로 도면에 디테일 부재정보가 일목요연하게 정리된다.

본 장에서 표현할 디테일 도면에서는 템플릿의 작업으로 Drawing Title, 물량산출(BOM), 축척, 문자크기 등으로 구성된 설계도서가 작성이 된다. 템플릿을 통해 구성되어진 도면의 양식은 전체 도면에서의 위치를 배치하고, 본 연구의 스마트그린 구조시스템 접합 부위의 상세도면 작성기준을 표준화 하였다. 뿐만 아니라, 그림 4.6에서와 같이 도면의 Layout 구성도 통일시켜 일관성 있게 도면을 생성하였다.

dCollection @ chosun

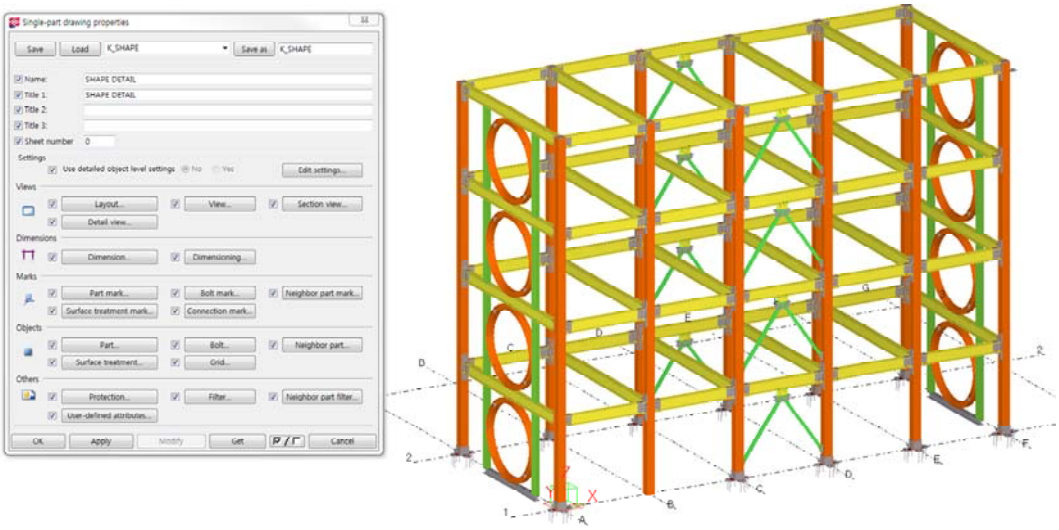


그림 4.6 도면 Layout 의 구성

4.3.2 스마트그린 구조시스템 핵심 기술요소 디테일 도면 출력

4.3.2.1 전체 구조물의 도면 출력

그림 4.7은 스마트그린 구조시스템의 전체 구조물의 도면을 나타낸 것이다. X축과 Y축을 기준하여 섹션을 나누어 단면도를 도면 한 장에 나타내었다. 스마트그린 구조시스템의 구조적 접합 및 수평저항장치 등이 3D 건축 정보 모델링에서 가감 없이 도면에 완전하게 표현되었으며, 도면의 레이아웃, 타이틀, 축척 등을 수정하여 도면생성을 완료하였다.

dCollection @ chosun

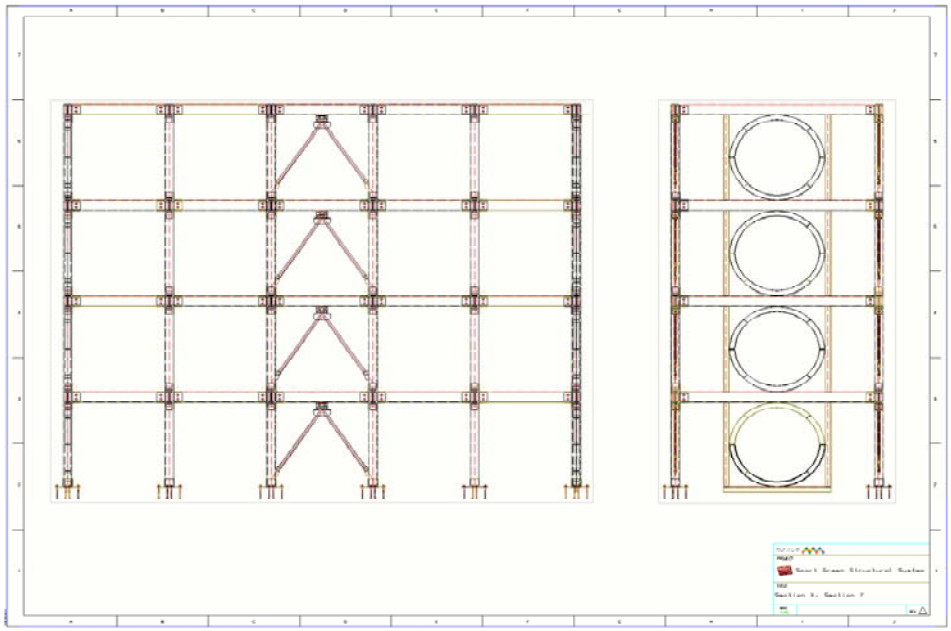


그림 4.7 전체 구조물의 도면

4.3.2.2 에너지 흡수장치의 도면 출력

그림 4.8은 대표적인 예로 전단형 마찰댐퍼의 도면 생성을 나타낸 것이다. 도면 레이아웃을 작성할 때 Drawing Title과 물량산출(BOM) 등 여러 가지 정보를 도면에 포함시킨다. 그림 4.8과 같이 도면에서 물량산출(BOM)은 우측 상단에 배치하고, Drawing Title은 우측 하단에 배치하였다. 또한 전단형 마찰댐퍼 디테일 도면에 대하여 정면도 및 단면도를 배치하였으며, 레이아웃을 사용자가 요구하는 물량만 뽑아낼 수 있도록 정리함으로써 도면에 나타나는 특수구조 형상 부재의 물량 및 볼트 등의 수량도 확인할 수 있도록 하였다. 이와 같이 디테일 도면 출력만으로도 해당하는 부재의 정량적인 물량을 확인할 수 있는 장점이 있다. 스마트그린 구조시스템의 다른 구조 상세인 에너지 흡수장치(강재댐퍼), 수평저항장치(브레이스), Flexible한 접합부(보-기둥 접합부 및 주각부) 디테일 도면은 부록 7에 나타내었다.

dCollection @ chosun

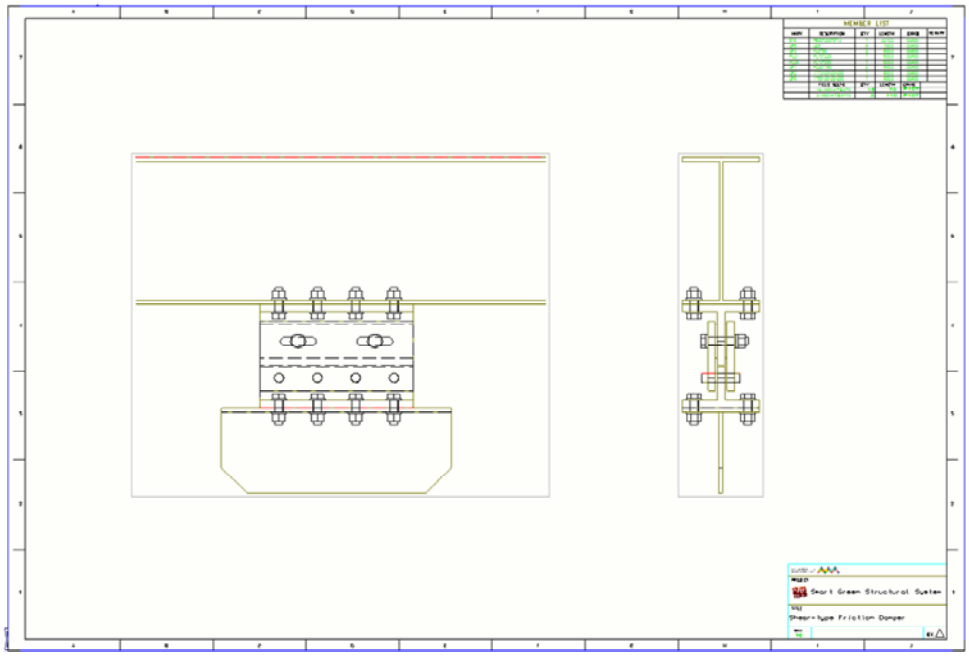


그림 4.8 전단형 마찰댐퍼 도면

4.4 소결

본 장에서는, 스마트그린 구조시스템을 대상으로 BIM을 기반한 3차원 건축정보 모델링을 완성하였고, 완성된 객체정보 모델 데이터를 활용하여 다음과 같은 결론을 얻었다.

1. BIM 데이터를 활용하여 스마트그린 구조시스템 구조물의 전체 물량 및 부재별 물량이 정량적이고 구체적으로 자동 산출되었다. 자동 물량산출을 활용하여 기존 구조시스템과 스마트그린 구조시스템 구조물의 재사용 순환 실적에 따른 물량 절감 효과를 확인하였다.

2. 스마트그린 구조시스템의 자동 도면 생성 결과 사용자가 요구하는 부분의 도면이 원활하게 생성되었다. 건축정보 모델로 도면이 자동 생성됨으로써 설계변경이 있을 경우, 3D 정보 모델 수정만으로도 도면 생성에 즉시 반영이 되고 시간과 비용이 절감될 수 있을 것으로 사료된다.

dCollection @ chosun

제5장 결 론

본 연구에서는 스마트그린 구조시스템의 S-BIM 기반 다차원 설계 모듈을 구현하여 다음과 같은 결론을 얻었다.

1. 구조 BIM 분야에서는 신기술/신공법 등의 새로운 구조형상에 대해 라이브러리를 제공하지 않아 모델링이 불가능하였다. 이에 효율적으로 대응하기 위하여 .NET FRAMEW-ORK 환경에 최적화되어 있는 C#을 이용함으로써, Tekla Structures 소프트웨어의 DB에 직접 접근하여 신규 라이브러리를 개발할 수 있었다.

2. 특수구조형상 라이브러리를 생성할 수 있는 프로세스 개발 및 설계자동화 모듈을 개발함으로써, Tekla Structures 프로그램내에 특수구조형상 모델링이 가능하였고, 이를 통해 FEEDBACK화된 데이터를 향후에도 활용할 수 있다.

3. BIM 데이터를 활용하여 스마트그린 구조시스템 구조물의 전체 물량 및 부재별 물량이 정량적이고 구체적으로 자동 산출되며, 자동 도면 출력 결과 사용자가 요구하는 부분의 도면이 원활하게 생성되었다.

4. BIM을 활용할시 3D 정보 모델을 적용함으로써 설계 변경 등 수정 사항이 있을 때 즉시 데이터가 반영이 되고, 이를 통해 시간과 비용이 절감될 수 있을 것으로 사료된다.

dCollection @ chosun

참고문헌

1. 국토교통부(2010), ‘건축 분야 BIM 적용 가이드(National Architectural BIM Guide)’
2. 조달청(2010), ‘시설사업 BIM적용 기본지침서 v1.0’
3. Stephen Kieran, James Tmberlake저, 윤용집, 임진택 공역(2006), “건축의 재구성(refabricating ARCHITECTURE)”, Spacetime・시공문화사
4. 야마나시 토모히코 저, 김명근 역(2011), “BIM 건축혁명”, 기문당
5. 김인한, 박정대, 박철수, 정종현, 추승연(2013), “BIM 총서-BIM: 건축설계 및 엔지니어링(BIM in Architecture: Design and Engineering)”, 기문당
6. 오세만, 이양선, 김정숙, 이창환(2005), “C#입문 이론과 실습(Introduction to C#: Theory and Practice)”, 생능출판사
7. 김상형(2008), “.NET 프로그래밍 정복(C#・윈폼・ADO)”, 가메출판사
8. 원철연(2010), “초보자를 위한 C# and Database 완벽가이드”, 영진닷컴
9. 박상현(2011), “뇌를 자극하는 C# 4.0 프로그래밍”, 한빛미디어
10. ANK Co., Ltd., 조정현(2009), “C#이 보이는 그림책”, 성안당
11. Ali M. Malkawi, Godfred Augenroe(2004), “Advanced Building Simulation”, Taylor & Francis e-Library, LONDON
12. Sohail Razvi(2008), “BIM and the Process Improvement Movement : Building a Case for a Combined BIM-CMMI Framework”, AECbytes, pp.1-8
13. Robert Leicht & John Messner(2008), “MOVING TOWARD AN ‘INTELLIGENT’ SHOP MODELING PROCESS, Journal of ITcon, v.13, pp.286-302
14. Kihong Ku, Spiro N. pollalis, Martin A. Fischer and Dennis R. Shelden(2008), “3D MODEL-BASED COLLABORATION IN DESIGN DEVELOPMENT AND CONSTRUCTION OF COMPLEX SHAPED BUILDINGS, Journal of ITcon, v.13, pp.458-485
15. D.Y.Abebe, J.W.Kim and J.H.Choi(2013), “HYSTERESIS CHARACTERISTICS OF CIRCULAR PIPE STEEL DAMPER USING LYP225”, Proceeding of Steel Innovations

dCollection @ chosun

Conference, 21-22 February, Christchurch, New Zealand.
16. 목학수, 조종래, 문광섭(2003), “재활용을 고려한 분리를 위한 설계(DfD) 지원 시스템 개발”, 한국경영과학회/대한산업공학회 2003 춘계동동학술회, pp.814-820
17. 조종래, 권재수, 홍병권, 홍존희, 권문식(2007), “설계단계 적용을 위한 차량의 해체용이설계(DfD: Design for Disaasembly) 통합시스템 개발”, 한국정밀공학회지, 제24권 제8호
18. 최철호, 윤상희, 황경미, 진상윤, 윤수원(2008), “성균관대학교 학술정보관 사례에서 본 비정형 건축물의 BIM 적용 현황과 발전 방향”, 대한건축학회 학술저널, 제52권 제4호, pp.66
19. 신태송, 조영상, 천진호, 엄진업, 양종민(2008), “Building Information Model(BIM) 기반의 건축 구조 설계 Pilot Study”, 한국전산구조공학회 2008정기학술회, pp.265-268
20. 서종철, 김인한(2009), “국내 건설 공공발주에서 BIM의 도입 및 적용을 위한 기본 방향에 관한 연구”, 대한건축학회논문집, 제25권 제9호, pp.21-30
21. 김유리, 조영상(2009), “구조설계 단계에서 한국형 BIM프로세스 적용 방안에 대한 사례 연구”, 대한건축학회논문집 구조계, 제25권 제10호, pp.19-26
22. 이병해(2010), “BIM의 구조공학분야 국내 소개 및 도입 현황”, 대한건축학회 학회지, 제54권 제1호, pp.52-57
23. 엄진업, 신태송(2010), “BIM 기반의 철골접합부 모델링 자동화에 관한 연구”, 한국강구조학회 논문집, 제22권 제1호, pp.99-108
24. 신태송, 양종민(2010), “BIM을 이용한 철골접합부 매크로 D/B작성 및 T-Split 접합부의 사용자 매크로 구축”, 대한건축학회지회연합회 논문집, 제12권 제1호, pp.175-182
25. 엄진업(2010), “BIM 기반의 구조 상세설계 자동화 시스템 구축에 관한 연구”, 동명대학교 석사학위 논문
26. 엄진업, 신태송(2011), “BIM기반의 구조설계와 상세설계의 인터페이스 모듈개발”, 한국강구조학회논문집, 제23권 제1호, pp.113-124

dCollection @ chosun

27. 엄진업, 신태송(2011), “BIM 기반의 고력볼트 마찰접합부 설계자동화 시스템의 프로토타입 구축”, 한국강구조학회논문집, 제23권 제5호, pp.637-646
28. 이승일, 권남하, 조영상(2011), “BIM기반 Set-based Design 적용 방안에 관한 연구”, 대한건축학회논문집 구조계, 제27권 제7호, pp.11-19
29. 최중식(2011), “개방형 BIM환경에서의 건축설계 품질향상을 위한 품질관리 기준 개발 및 적용에 관한 연구”, 경희대학교 석사학위 논문
30. 장자왕, 이제혁, 장현석, 홍성욱, 조영상(2012), “구조BIM 환경에서의 상호운영 체계 개선을 통한 플랫 플레이트 철근 상세배근 방법론에 관한 연구”, 대한건축학회논문집 구조계, 제28권 제8호, pp.3-10
31. 이재철, 정종현, 김치경, 김지현, 김민수(2012), “IFC를 이용한 건축강구조 표준 접합상세의 표현”, 대한건축학회논문집 구조계, 제28권 제7호, pp.65-72
32. 고아라, 이승일, 조영상(2013), “구조BIM 기반 철골접합부 자동화 설계시스템 개발”, 대한건축학회 논문집 구조계, 제29권 제2호(통권 292호), pp.21-28, 2013
33. 김태형, 신태송, “BIM기반의 브레이스 접합부의 구조설계 인터페이스 모듈 개발”, 한국강구조학회 논문집, 제25권 제2호, pp.197-207
35. 엄진업, 신태송(2013), “지속 가능한 구조설계 정보의 인터페이스 모듈 개발”, 한국강구조학회 논문집, 제25권 제5호, pp.509-517
36. 국가법령정보센터 website (2014): 폐기물관리법, 환경부(자원순환정책과), http://www.law.go.kr/LSW/lsStmdInfoP.do?lsiSeq=150419
37. 조달청 website (2014): http://www.pps.go.kr/kor/index.do
38. 국토교통부 website (2014): http://www.molit.go.kr/portal.do
39. 한국빌딩스마트협회 website (2014): http://www.buildingsmart.or.kr/
40. Tekla Structures(Trimble社) website (2014): http://www.tekla.com
41. TeklaOpenAPI Reference for Tekla Structures 20.0
42. Tekla OpenAPI Developer’s Guide (Tekla Structures 20.0), 2014

dCollection @ chosun

부록

1. 면외좌굴구속 강재댐퍼의 Visual C# 입력데이터

<상·하부 플레이트 부재>

ContourPlate class

```
TSM.ContourPlate Plate1 = new TSM.ContourPlate();

double Plate1_Width = 20;

Plate1.GetReportProperty("WIDTH", ref Plate1_Width);

CoordinateSystem Plate1Coordi = Plate1.GetCoordinateSystem();

TSM.ContourPoint point1 = new TSM.ContourPoint(new Point(1635 - 250, -200, -100), null);
TSM.ContourPoint point2 = new TSM.ContourPoint(new Point(1635 + 250, -200, -100), null);
TSM.ContourPoint point3 = new TSM.ContourPoint(new Point(1635 + 250, -200, 100), null);
TSM.ContourPoint point4 = new TSM.ContourPoint(new Point(1635 - 250, -200, 100), null);

Plate1.AddContourPoint(point1);
Plate1.AddContourPoint(point2);
Plate1.AddContourPoint(point3);
Plate1.AddContourPoint(point4);

Plate1.Name = "DAMPER";
Plate1.Profile.ProfileString = "PL20";
Plate1.Material.MaterialString = "SS400";
Plate1.Class = "3";
Plate1.Finish = "DP_Plate1";


Plate1.Position.Depth = Position.DepthEnum.FRONT;

if (!Plate1.Insert())
    Console.WriteLine("Plate1 inser failed");
```

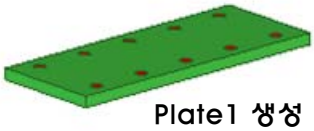


Plate1 생성

```
TSM.ContourPlate Plate2 = new TSM.ContourPlate();
TSM.ContourPoint point5 = new TSM.ContourPoint(new Point(1635 - 250, -510 - Plate1_Width, -100), null);
TSM.ContourPoint point6 = new TSM.ContourPoint(new Point(1635 + 250, -510 - Plate1_Width, -100), null);
TSM.ContourPoint point7 = new TSM.ContourPoint(new Point(1635 + 250, -510 - Plate1_Width, 100), null);
TSM.ContourPoint point8 = new TSM.ContourPoint(new Point(1635 - 250, -510 - Plate1_Width, 100), null);

Plate2.AddContourPoint(point5);
Plate2.AddContourPoint(point6);
Plate2.AddContourPoint(point7);
Plate2.AddContourPoint(point8);

Plate2.Name = "DAMPER";
Plate2.Profile.ProfileString = "PL20";
Plate2.Material.MaterialString = "SS400";
Plate2.Class = "3";
Plate2.Finish = "DP_Plate2";

if (!Plate2.Insert())
    Console.WriteLine("Plate2 inser failed");
```

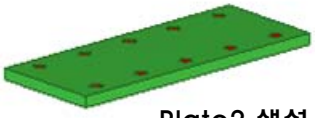


Plate2 생성

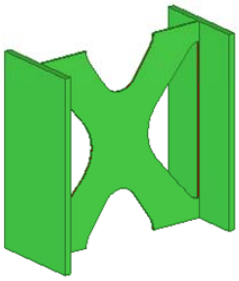
그림 1 상・하부 플레이트 부재 입력 데이터

dCollection @ chosun

<중앙부 패널 플레이트 부재>

Beam class

```
TSM.Beam B1 = new TSM.Beam();

double B1_Length = 300, B1_Heignt = 165, B1_Width = 140, B1_WebThickness = 3, B1_FlangeThickness = 10;

B1.GetReportProperty("LENGTH", ref B1_Length);
B1.GetReportProperty("HEIGHT", ref B1_Heignt);
B1.GetReportProperty("WIDTH", ref B1_Width);
B1.GetReportProperty("PROFILE.WEB_THICKNESS", ref B1_WebThickness);
B1.GetReportProperty("PROFILE.FLANGE_THICKNESS", ref B1_FlangeThickness);

CoordinateSystem B1Coordi = B1.GetCoordinateSystem();

B1.Name = "DAMPER";
B1.Profile.ProfileString = "HI300-3-10*140";
B1.Material.MaterialString = "SS400";
B1.Class = "3";
B1.Finish = "DP_B1";

B1.StartPoint = new T3D.Point(1635, -220, 0);
B1.EndPoint = new T3D.Point(1635, -520, 0);
B1.Position.Depth = Position.DepthEnum.MIDDLE;
B1.Position.DepthOffset = -0.00;
B1.Position.Plane = Position.PlaneEnum.MIDDLE;
B1.Position.PlaneOffset = 0.00;
B1.Position.Rotation = Position.RotationEnum.FRONT;
B1.Position.RotationOffset = -0.00000;

if (B1.Insert())
    Console.WriteLine("B1 insert failed!");
```



B1 생성

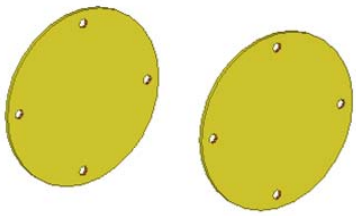
그림 2. 중앙부 패널 플레이트 부재 입력 데이터

dCollection @ chosun

<중앙부 구속 플레이트 부재>

Beam class

```
TSM.Beam Circle1 = new TSM.Beam( );
double Circle1_Diameter = 260;

Circle1.GetReportProperty("DIAMETER", ref Circle1_Diameter);

Circle1.Profile.ProfileString = "D260";
Circle1.Material.MaterialString = "SS400";
Circle1.Class = "6";
Circle1.Finish = "Circle1";


Circle1.StartPoint = new T3D.Point(1635, -240, -1.5);
Circle1.EndPoint = new T3D.Point(1635, -240, -6.5);
Circle1.Position.Depth = Position.DepthEnum.MIDDLE;
Circle1.Position.DepthOffset = -0.00;
Circle1.Position.Plane = Position.PlaneEnum.RIGHT;
Circle1.Position.PlaneOffset = 0.00;
Circle1.Position.Rotation = Position.RotationEnum.FRONT;
Circle1.Position.RotationOffset = -0.00000;


TSM.Beam Circle2 = new TSM.
Beam(new Point(1635, -240, 6.5), new Point(1635, -240, 1.5));
Circle2.Profile.ProfileString = "D260";
Circle2.Material.MaterialString = "SS400";
Circle2.Class = "6";
Circle2.Finish = "Circle2";

Circle2.Position.Depth = Position.DepthEnum.MIDDLE;
Circle2.Position.DepthOffset = -0.00;
Circle2.Position.Plane = Position.PlaneEnum.RIGHT;
Circle2.Position.PlaneOffset = 0.00;
Circle2.Position.Rotation = Position.RotationEnum.FRONT;
Circle2.Position.RotationOffset = -0.00000;


if (!Circle1.Insert())
    Console.WriteLine("Circle1 Insert failed!");
if (!Circle2.Insert())
    Console.WriteLine("Circle2 Insert failed!");
```



Circle1, Circle2 생성

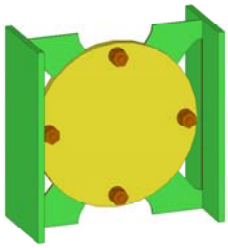
그림 3 구속 플레이트 부재 입력 데이터

dCollection @ chosun

<중앙부 구속 볼트>

BoltArray class

```
TSM.BoltArray bolt1 = new TSM.BoltArray();

bolt1.PartToBoltTo = Circle1;
bolt1.PartToBeBolted = Circle2 as TSM.Beam;
bolt1.FirstPosition = new Point(1635, -260, 0);
bolt1.SecondPosition = new Point(1635, -500, 0);

bolt1.BoltSize = 12;
bolt1.BoltStandard = "H.T.B";
bolt1.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt1.Tolerance = 2;
bolt1.CutLength = 100;
bolt1.Length = 60;
bolt1.ExtraLength = 0;

bolt1.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt1.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt1.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt1.Position.Rotation = Position.RotationEnum.TOP;
bolt1.Position.RotationOffset = 90;

bolt1.Bolt = true;
bolt1.Washer1 = true;
bolt1.Washer2 = true;
bolt1.Nut1 = true;
bolt1.Nut2 = false;

bolt1.Hole1 = false;
bolt1.Hole2 = false;

bolt1.AddBoltDistX(0);
bolt1.AddBoltDistX(220);
bolt1.AddBoltDistY(0);

if (bolt1.Insert())
    Console.WriteLine("bolt1 insert failed!");

TSM.BoltArray bolt2 = new TSM.BoltArray();

bolt2.PartToBoltTo = Circle1;
bolt2.PartToBeBolted = Circle2 as TSM.Beam;
bolt2.FirstPosition = new Point(1525, -360, 0);
bolt2.SecondPosition = new Point(1765, -360, 0);

bolt2.BoltSize = 12;
bolt2.BoltStandard = "H.T.B";
bolt2.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt2.Tolerance = 2;
bolt2.CutLength = 100;
bolt2.Length = 60;
bolt2.ExtraLength = 0;

bolt2.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt2.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt2.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt2.Position.Rotation = Position.RotationEnum.TOP;
bolt2.Position.RotationOffset = 90;
```



Bolt1, bolt2 생성

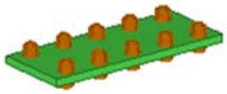
그림 4 중앙부 구속 볼트 입력 데이터

dCollection @ chosun

<상·하부 플레이트 볼트>

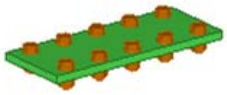
BoltArray class

```
TSM.BoltArray bolt3 = new TSM.BoltArray( );

bolt3.PartToBoltTo = Plate1;
bolt3.PartToBeBolted = Plate1 as TSM.ContourPlate;
bolt3.FirstPosition = new Point(1635 - 200, -240, 0);
bolt3.SecondPosition = new Point(1635 + 250, -240, 0);

bolt3.BoltSize = 12;
bolt3.BoltStandard = "H.T.B";
bolt3.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt3.Tolerance = 2;
bolt3.CutLength = 100;
bolt3.Length = 60;
bolt3.ExtraLength = 0;

bolt3.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;
bolt3.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt3.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt3.Position.Rotation = Position.RotationEnum.TOP;
bolt3.Position.RotationOffset = 0;

bolt3.Bolt = true;
bolt3.Washer1 = true;
bolt3.Washer2 = true;
bolt3.Nut1 = true;
bolt3.Nut2 = false;

bolt3.Hole1 = false;
bolt3.Hole2 = false;

bolt3.AddBoltDistX(100);
bolt3.AddBoltDistX(100);
bolt3.AddBoltDistX(100);
bolt3.AddBoltDistX(100);
bolt3.AddBoltDistY(140);

if (bolt3.Insert())
    Console.WriteLine("bolt3 insert failed!");
```



Bolt 3 생성

```
TSM.BoltArray bolt4 = new TSM.BoltArray( );

bolt4.PartToBoltTo = Plate2;
bolt4.PartToBeBolted = Plate2 as TSM.ContourPlate;
bolt4.FirstPosition = new Point(1635 - 200, -500, 0);
bolt4.SecondPosition = new Point(1635 + 250, -500, 0);

bolt4.BoltSize = 12;
bolt4.BoltStandard = "H.T.B";
bolt4.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt4.Tolerance = 2;
bolt4.CutLength = 100;
bolt4.Length = 60;
bolt4.ExtraLength = 0;

bolt4.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;
bolt4.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt4.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt4.Position.Rotation = Position.RotationEnum.BELOW;
bolt4.Position.RotationOffset = 0;
```



Bolt 4 생성

그림 5 상・하부 플레이트 볼트 입력 데이터

dCollection @ chosun

2. 원형강관댐퍼의 Visual C# 입력데이터

<상부 플레이트 부재>

ContourPlate class

```
TSM.ContourPlate Plate1 = new TSM.ContourPlate();

double Plate1_Width = 20;
Plate1.GetReportProperty("WIDTH", ref Plate1_Width);
CoordinateSystem Plate1Coordi = Plate1.GetCoordinateSystem();

TSM.ContourPoint point1 = new TSM.ContourPoint(new Point(1635-200, -200, -100), null);
TSM.ContourPoint point2 = new TSM.ContourPoint(new Point(1635+200 , -200, -100), null);
TSM.ContourPoint point3 = new TSM.ContourPoint(new Point(1635+200, -200, 100), null);
TSM.ContourPoint point4 = new TSM.ContourPoint(new Point(1635-200, -200, 100), null);

Plate1.AddContourPoint(point1);
Plate1.AddContourPoint(point2);
Plate1.AddContourPoint(point3);
Plate1.AddContourPoint(point4);

Plate1.Name = "DAMPER";
Plate1.Profile.ProfileString = "PL20";
Plate1.Material.MaterialString = "SS400";
Plate1.Class = "3";
Plate1.Finish = "DP_Plate1";
Plate1.Position.Depth = Position.DepthEnum.FRONT;

if (!Plate1.Insert())
    Console.WriteLine("DP_Plate1 insert failed");
```

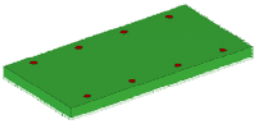


Plate1 생성

```
TSM.ContourPlate CP1 = new TSM.ContourPlate();
TSM.ContourPoint point13 = new TSM.ContourPoint(new Point(1570, -220, -65), new TSM.Chamfer(65, 0,
TSM.ContourPoint point14 = new TSM.ContourPoint(new Point(1570,-220, 65), new TSM.Chamfer(65, 0, TS
TSM.ContourPoint point15 = new TSM.ContourPoint(new Point(1570+130, -220, 65), new TSM.Chamfer(65,
TSM.ContourPoint point16 = new TSM.ContourPoint(new Point(1570+130, -220, -65), new TSM.Chamfer(65,

CP1.AddContourPoint(point13);
CP1.AddContourPoint(point14);
CP1.AddContourPoint(point15);
CP1.AddContourPoint(point16);

CP1.Profile.ProfileString = "PL40";
CP1.Class = TSM.BooleanPart.BooleanOperativeClassName;
if (!CP1.Insert())
    Console.WriteLine("CP1 inser failed");

TSM.BooleanPart boolpart1 = new TSM.BooleanPart();
boolpart1.Father = Plate1;
boolpart1.SetOperativePart(CP1);
if (!boolpart1.Insert())
    Console.WriteLine("Boolpart1 insert failed!");

CP1.Delete();
```

그림 6 상부 플레이트 입력 데이터

dCollection @ chosun

<상부 강관 구속 플레이트 부재>

ContourPlate class

```
TSM.ContourPlate Plate2 = new TSM.ContourPlate();

TSM.ContourPoint point5 = new TSM.ContourPoint(new Point(1635 - 200, - 200-Plate1_Width , -100), nu
TSM.ContourPoint point6 = new TSM.ContourPoint(new Point(1635 + 200, -200 - Plate1_Width, -100), nu
TSM.ContourPoint point7 = new TSM.ContourPoint(new Point(1635 + 200, -200 - Plate1_Width, 100), nul
TSM.ContourPoint point8 = new TSM.ContourPoint(new Point(1635 - 200, -200 - Plate1_Width, 100), nul

Plate2.AddContourPoint(point5);
Plate2.AddContourPoint(point6);
Plate2.AddContourPoint(point7);
Plate2.AddContourPoint(point8);

Plate2.Name = "DAMPER";
Plate2.Profile.ProfileString = "PL20";
Plate2.Material.MaterialString = "SS400";
Plate2.Class = "3";
Plate2.Finish = "DP_Plate2";
Plate2.Position.Depth = Position.DepthEnum.FRONT;

if (!Plate2.Insert())
    Console.WriteLine("DP_Plate2 insert failed");
```

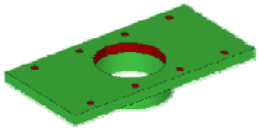


Plate2, B1 생성

Beam class

```
TSM.Beam B1 = new TSM.Beam();

double B1_Diameter=150,B1_Platethickness=10;

B1.GetReportProperty("DIAMETER", ref B1_Diameter);
B1.GetReportProperty("PLATETHICKNESS", ref B1_Platethickness);

CoordinateSystem B1Coordi = B1.GetCoordinateSystem();

B1.Name = "DAMPER";
B1.Profile.ProfileString = "PD150*10";
B1.Material.MaterialString = "SS400";
B1.Class = "3";
B1.Finish = "DP_SP(B1)";

B1.StartPoint = new T3D.Point(1635, -240, 0);
B1.EndPoint = new T3D.Point(1635, -280, 0);
B1.Position.Depth = Position.DepthEnum.MIDDLE;
B1.Position.DepthOffset = -0.00;
B1.Position.Plane = Position.PlaneEnum.MIDDLE;
B1.Position.PlaneOffset = 0.00;
B1.Position.Rotation = Position.RotationEnum.FRONT;
B1.Position.RotationOffset = -0.00000;

if (!B1.Insert())
    Console.WriteLine("B1 Insert failed!");

TSM.Weld W1 = new TSM.Weld();
W1.MainObject = Plate2 as TSM.ContourPlate;
W1.SecondaryObject = B1 as TSM.Beam;

W1.TypeAbove = TSM.Weld.WeldTypeEnum.WELD_TYPE_FILLET;
if (!W1.Insert())
    Console.WriteLine("W1 insert failed!");
W1.Modify();
```

그림 7 상부 플레이트 입력 데이터

dCollection @ chosun

<하부 강관 구속 플레이트 부재>

ContourPlate class

```
TSM.ContourPlate Plate3 = new TSM.ContourPlate();
TSM.ContourPoint point9 = new TSM.ContourPoint(new Point(1635 - 200, -590 - Plate1_Width, -100), null);
TSM.ContourPoint point10 = new TSM.ContourPoint(new Point(1635 + 200, -590 - Plate1_Width, -100), null);
TSM.ContourPoint point11 = new TSM.ContourPoint(new Point(1635 + 200, -590 - Plate1_Width, 100), null);
TSM.ContourPoint point12 = new TSM.ContourPoint(new Point(1635 - 200, -590 - Plate1_Width, 100), null);

Plate3.AddContourPoint(point9);
Plate3.AddContourPoint(point10);
Plate3.AddContourPoint(point11);
Plate3.AddContourPoint(point12);

Plate3.Name = "DAMPER";
Plate3.Profile.ProfileString = "PL20";
Plate3.Material.MaterialString = "SS400";
Plate3.Class = "3";
Plate3.Finish = "DP_Plate3";

if (!Plate3.Insert())
    Console.WriteLine("Plate3 inser failed");
```

Beam class

```
TSM.Beam B3 = new TSM.
Beam(new Point(8925, 0, 10320), new Point(8925, 0, 10360));

B3.Name = "DAMPER";
B3.Profile.ProfileString = "PD150*10";
B3.Material.MaterialString = "SS400";
B3.Class = "3";
B3.Finish = "DP_SP(B3)";

B3.StartPoint = new T3D.Point(1635, -600, 0);
B3.EndPoint = new T3D.Point(1635, -560, 0);
B3.Position.Depth = Position.DepthEnum.MIDDLE;
B3.Position.DepthOffset = -0.00;
B3.Position.Plane = Position.PlaneEnum.MIDDLE;
B3.Position.PlaneOffset = 0.00;
B3.Position.Rotation = Position.RotationEnum.FRONT;
B3.Position.RotationOffset = -0.00000;

if (B3.Insert())
    Console.WriteLine("B3 insert failed!");

TSM.Weld W2 = new TSM.Weld();
W2.MainObject = B3 as TSM.Beam;
W2.SecondaryObject = Plate3 as TSM.ContourPlate;

W2.TypeAbove = TSM.Weld.WeldTypeEnum.WELD_TYPE_FILLET;
if (!W2.Insert())
    Console.WriteLine("W2 insert failed!");
W2.Modify();
```

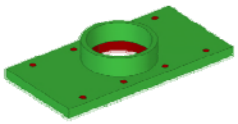


Plate3, B3 생성

그림 8 하부 강관구속 플레이트 입력 데이터

dCollection @ chosun

<중앙부 원형강관 >

Beam class

```
TSM.Beam B2 = new TSM.Beam();

double B2_Diameter = 130, B2_Platethickness = 10;

B2.GetReportProperty("DIAMETER", ref B2_Diameter);
B2.GetReportProperty("PLATETHICKNESS", ref B2_Platethickness);

CoordinateSystem B2Coordi = B2.GetCoordinateSystem();

B2.Name = "DAMPER";
B2.Profile.ProfileString = "PD130*10";
B2.Material.MaterialString = "SS400";
B2.Class = "20";
B2.Finish = "DP_MainSP(B2)";


B2.StartPoint = new T3D.Point(1635, -200, 0);
B2.EndPoint = new T3D.Point(1635, -600, 0);
B2.Position.Depth = Position.DepthEnum.MIDDLE;
B2.Position.DepthOffset = -0.00;
B2.Position.Plane = Position.PlaneEnum.MIDDLE;
B2.Position.PlaneOffset = 0.00;
B2.Position.Rotation = Position.RotationEnum.FRONT;
B2.Position.RotationOffset = -0.00000;

if (!B2.Insert())
    Console.WriteLine("B2 Insert failed!");
```

B2 생성

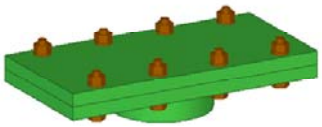
그림 9 중앙부 원형강관 입력 데이터

dCollection @ chosun

<상부 플레이트 볼트 >

BoltArray class

```
TSM.BoltArray bolt1 = new TSM.BoltArray();

bolt1.PartToBoltTo = Plate1;
bolt1.PartToBeBolted = MainPart;

bolt1.FirstPosition = new Point(1635 - 150, -200 - Plate1_Width, 0);
bolt1.SecondPosition = new Point(1635 + 150, -200 - Plate1_Width, 0);

bolt1.BoltSize = 12;
bolt1.BoltStandard = "H.T.B";
bolt1.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt1.Tolerance = 2;
bolt1.CutLength = 100;
bolt1.Length = 60;
bolt1.ExtraLength = 0;

bolt1.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt1.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt1.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt1.Position.Rotation = Position.RotationEnum.TOP;


bolt1.Bolt = true;
bolt1.Washer1 = true;
bolt1.Washer2 = false;
bolt1.Nut1 = true;
bolt1.Nut2 = false;

bolt1.Hole1 = false;
bolt1.Hole2 = false;
bolt1.Hole3 = false;
bolt1.Hole4 = false;
bolt1.Hole5 = false;

bolt1.AddBoltDistX(100);
bolt1.AddBoltDistX(100);
bolt1.AddBoltDistX(100);
bolt1.AddBoltDistY(160);

if (bolt1.Insert())
    Console.WriteLine("bolt1 insert failed!");
```



bolt1 생성

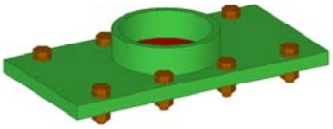
그림 10 상부 플레이트 볼트 입력 데이터

dCollection @ chosun

<하부 플레이트 볼트 >

BoltArray class

```
TSM.BoltArray bolt2 = new TSM.BoltArray();

bolt2.PartToBoltTo = Plate3;
bolt2.PartToBeBolted = SecondaryPart;
bolt2.FirstPosition = new Point(1635 - 150, -600 - Plate1_Width, 0);
bolt2.SecondPosition = new Point(1635 + 150, -600 - Plate1_Width, 0);

bolt2.BoltSize = 12;
bolt2.BoltStandard = "H.T.B";
bolt2.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt2.Tolerance = 2;
bolt2.CutLength = 100;
bolt2.Length = 60;
bolt2.ExtraLength = 0;

bolt2.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt2.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt2.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt2.Position.Rotation = Position.RotationEnum.TOP;
bolt2.Position.RotationOffset = 180;

bolt2.Bolt = true;
bolt2.Washer1 = true;
bolt2.Washer2 = false;
bolt2.Nut1 = true;
bolt2.Nut2 = false;

bolt2.Hole1 = false;
bolt2.Hole2 = false;
bolt2.Hole3 = false;
bolt2.Hole4 = false;
bolt2.Hole5 = false;

bolt2.AddBoltDistX(100);
bolt2.AddBoltDistX(100);
bolt2.AddBoltDistX(100);
bolt2.AddBoltDistY(160);

if (bolt2.Insert())
    Console.WriteLine("bolt2 insert failed!");
```



bolt2 생성

그림 11 하부 플레이트 볼트 입력 데이터

dCollection @ chosun

3. 원형 브레이스의 Visual C# 입력데이터

<하부 브레이스 부재>

PolyBeam class

```
PolyBeam CB1 = new PolyBeam();
CB1.Name = "BEAM";
CB1.Profile.ProfileString = "H200*200*8*12";
CB1.Material.MaterialString = "SS400";
CB1.Class = "13";
CB1.Finish = "Poly_Brace";


CB1.AddContourPoint(point11);
CB1.AddContourPoint(point12);
CB1.AddContourPoint(point13);
```

CB1 생성

```
ContourPlate cp01 = new ContourPlate();
cp01.AddContourPoint(new ContourPoint(new T3D.Point(-100, 1890, 1600), null));
cp01.AddContourPoint(new ContourPoint(new T3D.Point(-100, 2110.78, 1600), null));
cp01.AddContourPoint(new ContourPoint(new T3D.Point(100, 2110.78, 1600), null));
cp01.AddContourPoint(new ContourPoint(new T3D.Point(100, 1890, 1600), null));
cp01.Profile.ProfileString = "PL15";
cp01.Position.Depth = Position.DepthEnum.FRONT;
cp01.Class = BooleanPart.BooleanOperativeClassName;
if (!cp01.Insert())
    MessageBox.Show("cp01 insert failed!");

BooleanPart boolpart01 = new BooleanPart();
boolpart01.Father = CB1;
boolpart01.SetOperativePart(cp01);
if (!boolpart01.Insert())
    MessageBox.Show("Booleanpart01 insert failed");
cp01.Delete();

ContourPlate cp011 = new ContourPlate();
cp011.AddContourPoint(new ContourPoint(new T3D.Point(-100, 5090, 1600), null));
cp011.AddContourPoint(new ContourPoint(new T3D.Point(-100, 5310, 1600), null));
cp011.AddContourPoint(new ContourPoint(new T3D.Point(100, 5310, 1600), null));
cp011.AddContourPoint(new ContourPoint(new T3D.Point(100, 5090, 1600), null));
cp011.Profile.ProfileString = "PL15";
cp011.Position.Depth = Position.DepthEnum.FRONT;
cp011.Class = BooleanPart.BooleanOperativeClassName;
if (!cp011.Insert())
    MessageBox.Show("cp011 insert failed!");

BooleanPart boolpart011 = new BooleanPart();
boolpart011.Father = CB1;
boolpart011.SetOperativePart(cp011);
if (!boolpart011.Insert())
    MessageBox.Show("Booleanpart011 insert failed");
cp011.Delete();
```

그림 12. 하부 브레이스 입력 데이터

dCollection @ chosun

<상부 브레이스 부재>

PolyBeam class

```
PolyBeam CB2 = new PolyBeam();
CB2.Name = "BEAM";
CB2.Profile.ProfileString = "H200*200*8*12";
CB2.Material.MaterialString = "SS400";
CB2.Class = "13";
CB2.Finish = "Poly_Brace";
```

CB2 생성

```
CB2.AddContourPoint(point21);
CB2.AddContourPoint(point22);
CB2.AddContourPoint(point23);
ContourPlate cp02 = new ContourPlate();
cp02.AddContourPoint(new ContourPoint(new T3D.Point(-100, 1890, 1600), null));
cp02.AddContourPoint(new ContourPoint(new T3D.Point(-100, 2110.78, 1600), null));
cp02.AddContourPoint(new ContourPoint(new T3D.Point(100, 2110.78, 1600), null));
cp02.AddContourPoint(new ContourPoint(new T3D.Point(100, 1890, 1600), null));
cp02.Profile.ProfileString = "PL15";
cp02.Position.Depth = Position.DepthEnum.BEHIND;
cp02.Class = BooleanPart.BooleanOperativeClassName;
if (!cp02.Insert())
    MessageBox.Show("cp02 insert failed!");

BooleanPart boolpart02 = new BooleanPart();
boolpart02.Father = CB2;
boolpart02.SetOperativePart(cp02);
if (!boolpart02.Insert())
    MessageBox.Show("Booleanpart02 insert failed");
cp02.Delete();

ContourPlate cp021 = new ContourPlate();
cp021.AddContourPoint(new ContourPoint(new T3D.Point(-100, 5090, 1600), null));
cp021.AddContourPoint(new ContourPoint(new T3D.Point(-100, 5310, 1600), null));
cp021.AddContourPoint(new ContourPoint(new T3D.Point(100, 5310, 1600), null));
cp021.AddContourPoint(new ContourPoint(new T3D.Point(100, 5090, 1600), null));
cp021.Profile.ProfileString = "PL15";
cp021.Position.Depth = Position.DepthEnum.BEHIND;
cp021.Class = BooleanPart.BooleanOperativeClassName;
if (!cp021.Insert())
    MessageBox.Show("cp021 insert failed!");

BooleanPart boolpart021 = new BooleanPart();
boolpart021.Father = CB2;
boolpart021.SetOperativePart(cp021);
if (!boolpart021.Insert())
    MessageBox.Show("Booleanpart021 insert failed");
cp021.Delete();
```

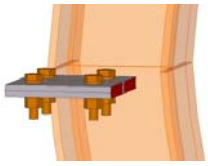
그림 13. 상부 브레이스 입력 데이터

dCollection @ chosun

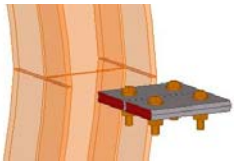
<수평 스티프너 & 볼트>

ContourPlate class

```
ContourPlate cp1 = new ContourPlate();
cp1.AddContourPoint(new ContourPoint(new T3D.Point(-4, 1905, 1600), null));
cp1.AddContourPoint(new ContourPoint(new T3D.Point(-4, 2101, 1600), null));
cp1.AddContourPoint(new ContourPoint(new T3D.Point(-100, 2101, 1600), null));
cp1.AddContourPoint(new ContourPoint(new T3D.Point(-100, 1905, 1600), null));
cp1.Name = "PLATE";
cp1.Profile.ProfileString = "PL10";
cp1.Material.MaterialString = "SS400";
cp1.Finish = "Stiffner1";
cp1.Class = "99";

cp1.Position.Depth = Position.DepthEnum.FRONT;
cp1.Position.DepthOffset = -0.00;
cp1.Insert();
```



cp1 생성



cp2 생성

```
ContourPlate cp2 = new ContourPlate();
cp2.AddContourPoint(new ContourPoint(new T3D.Point(4, 1905, 1600), null));
cp2.AddContourPoint(new ContourPoint(new T3D.Point(4, 2101, 1600), null));
cp2.AddContourPoint(new ContourPoint(new T3D.Point(100, 2101, 1600), null));
cp2.AddContourPoint(new ContourPoint(new T3D.Point(100, 1905, 1600), null));
cp2.Name = "PLATE";
cp2.Profile.ProfileString = "PL10";
cp2.Material.MaterialString = "SS400";
cp2.Finish = "Stiffner2";
cp2.Class = "99";
cp2.Position.Depth = Position.DepthEnum.FRONT;
cp2.Position.DepthOffset = -0.00;
cp2.Insert();
```

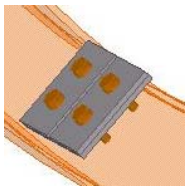
그림 14. 수평 스티프너 & 볼트 입력 데이터

dCollection @ chosun

ContourPlate class

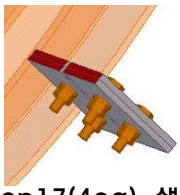
<보강 스티프너 & 볼트>

```
ContourPlate cp9 = new ContourPlate();
cp9.AddContourPoint(new ContourPoint(new T3D.Point(-4, 2550.47, 610.80), null));
cp9.AddContourPoint(new ContourPoint(new T3D.Point(-4, 2413.23, 481.63), null));
cp9.AddContourPoint(new ContourPoint(new T3D.Point(-100, 2413.23, 481.63), null));
cp9.AddContourPoint(new ContourPoint(new T3D.Point(-100, 2550.47, 610.80), null));
cp9.Name = "PLATE";
cp9.Profile.ProfileString = "PL10";
cp9.Material.MaterialString = "SS400";
cp9.Finish = "Stiffner9";
cp9.Class = "99";
cp9.Position.Depth = Position.DepthEnum.FRONT;
cp9.Position.DepthOffset = -0.00;
cp9.Insert();
```



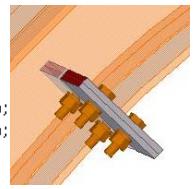
cp9(4ea) 생성

```
ContourPlate cp17 = new ContourPlate();
cp17.AddContourPoint(new ContourPoint(new T3D.Point(4, 4641.53, 619.73), null));
cp17.AddContourPoint(new ContourPoint(new T3D.Point(4, 4787.91, 482.33), null));
cp17.AddContourPoint(new ContourPoint(new T3D.Point(100, 4787.91, 482.33), null));
cp17.AddContourPoint(new ContourPoint(new T3D.Point(100, 4641.53, 619.73), null));
cp17.Name = "PLATE";
cp17.Profile.ProfileString = "PL10";
cp17.Material.MaterialString = "SS400";
cp17.Finish = "Stiffner16";
cp17.Class = "99";
cp17.Position.Depth = Position.DepthEnum.FRONT;
cp17.Position.DepthOffset = -0.00;
cp17.Insert();
```



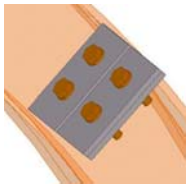
cp17(4ea) 생성

```
ContourPlate cp19 = new ContourPlate();
cp19.AddContourPoint(new ContourPoint(new T3D.Point(4, 2412.47, 2717.67), null));
cp19.AddContourPoint(new ContourPoint(new T3D.Point(4, 2558.47, 2580.27), null));
cp19.AddContourPoint(new ContourPoint(new T3D.Point(100, 2558.47, 2580.27), null));
cp19.AddContourPoint(new ContourPoint(new T3D.Point(100, 2412.47, 2717.67), null));
cp19.Name = "PLATE";
cp19.Profile.ProfileString = "PL10";
cp19.Material.MaterialString = "SS400";
cp19.Finish = "Stiffner19";
cp19.Class = "99";
cp19.Position.Depth = Position.DepthEnum.BEHIND;
cp19.Position.DepthOffset = -0.00;
cp19.Insert();
```



cp19(4ea) 생성

```
ContourPlate cp23 = new ContourPlate();
cp23.AddContourPoint(new ContourPoint(new T3D.Point(-4, 4788.26, 2716.96), null));
cp23.AddContourPoint(new ContourPoint(new T3D.Point(-4, 4642.29, 2579.58), null));
cp23.AddContourPoint(new ContourPoint(new T3D.Point(-100, 4642.29, 2579.58), null));
cp23.AddContourPoint(new ContourPoint(new T3D.Point(-100, 4788.26, 2716.96), null));
cp23.Name = "PLATE";
cp23.Profile.ProfileString = "PL10";
```



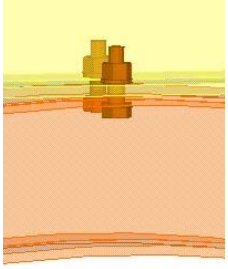
cp23(4ea) 생성

그림 15. 보강 스티프너 & 볼트 입력 데이터

<상, 하부 볼트>

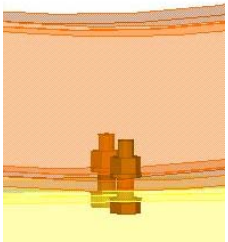
BoltArray class

```
BoltArray Bolt11 = new BoltArray();

Bolt11.PartToBeBolted = CB1;
Bolt11.PartToBoltTo = CB1 as PolyBeam;

Bolt11.FirstPosition = new Point(50, 3600, 0);
Bolt11.SecondPosition = new Point(-50, 3600, 0);

Bolt11.StartPointOffset.Dx = 0;

Bolt11.BoltSize = 24;
Bolt11.Tolerance = 2.00;
Bolt11.BoltStandard = "H.T.B";
Bolt11.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
Bolt11.CutLength = 105;
```



bolt11 생성

```
BoltArray Bolt12 = new BoltArray();
Bolt12.PartToBeBolted = CB2;
Bolt12.PartToBoltTo = CB2 as PolyBeam;

Bolt12.FirstPosition = new Point(50, 3600, 3200);
Bolt12.SecondPosition = new Point(-50, 3600, 3200);

Bolt12.StartPointOffset.Dx = 0;

Bolt12.BoltSize = 24;
Bolt12.Tolerance = 2.00;
Bolt12.BoltStandard = "H.T.B";
Bolt12.BoltType = BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
Bolt12.CutLength = 105;

Bolt12.Length = 60;
Bolt12.ExtraLength = 15;
Bolt12.ThreadInMaterial = BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

Bolt12.Position.Depth = Position.DepthEnum.MIDDLE;
Bolt12.Position.Plane = Position.PlaneEnum.MIDDLE;
Bolt12.Position.Rotation = Position.RotationEnum.BACK;


Bolt12.AddBoltDistX(100);
Bolt12.AddBoltDistY(0);
```



bolt12 생성

그림 16. 상・하부 볼트 입력 데이터

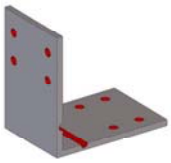
dCollection @ chosun

4. Flexible한 보-기둥 접합부의 Visual C# 입력데이터

<상부 플랜지 플레이트>

Beam class

```
TSM.Beam B3 = new TSM.Beam(new Point(300, -100, 3900), new Point(300, 100, 3900));
B3.Name = "PLATE";
B3.Profile.ProfileString = "L300*300*15";
B3.Material.MaterialString = "SS400";
B3.Class = "1";
B3.Finish = "PL_Top";
if (B3.Insert())
    Console.WriteLine("Connection1 insert failed!");
```



B3 생성

ContourPlate class

```
TSM.ContourPlate B3_Slotted1 = new TSM.ContourPlate();
TSM.ContourPoint point17 = new TSM.ContourPoint(new Point(215, 0, 3615), null);
TSM.ContourPoint point18 = new TSM.ContourPoint(new Point(217.93, 0, 3607.93), null);
TSM.ContourPoint point19 = new TSM.ContourPoint(new Point(225, 0, 3605), null);
TSM.ContourPoint point20 = new TSM.ContourPoint(new Point(232.07, 0, 3607.93), null);
TSM.ContourPoint point21 = new TSM.ContourPoint(new Point(235, 0, 3615), null);

B3_Slotted1.AddContourPoint(point17);
B3_Slotted1.AddContourPoint(point18);
B3_Slotted1.AddContourPoint(point19);
B3_Slotted1.AddContourPoint(point20);
B3_Slotted1.AddContourPoint(point21);


B3_Slotted1.Profile.ProfileString = "PL200";
B3_Slotted1.Class = TSM.BooleanPart.BooleanOperativeClassName;
if (!B3_Slotted1.Insert())
    Console.WriteLine("B3_Slotted1 inser failed");

TSM.BooleanPart boolpart7 = new TSM.BooleanPart();
boolpart7.Father = B3;
boolpart7.SetOperativePart(B3_Slotted1);
if (!boolpart7.Insert())
    Console.WriteLine("Boolpar7 insert failed!");

B3_Slotted1.Delete();
```

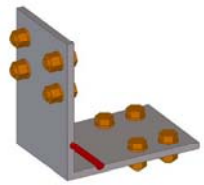
그림 21. 상부 플랜지 플레이트 입력 데이터

dCollection @ chosun

BoltArray class

<상부 플랜지 플레이트 볼트>

```
TSM.BoltArray bolt1 = new TSM.BoltArray();

bolt1.PartToBoltTo = B1;
bolt1.PartToBeBolted = B3 as TSM.Beam;
bolt1.FirstPosition = new Point(185, -65, 3805);
bolt1.SecondPosition = new Point(185, 75, 3805);

bolt1.BoltSize = 20;
bolt1.BoltStandard = "H.T.B";
bolt1.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt1.Tolerance = 2;
bolt1.CutLength = 150;
bolt1.Length = 60;
bolt1.ExtraLength = 0;

bolt1.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt1.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt1.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt1.Position.Rotation = TSM.Position.RotationEnum.TOP;
bolt1.Position.RotationOffset = 90;
bolt1.Position.Rotation = TSM.Position.RotationEnum.BACK;
bolt1.Position.RotationOffset = 90;


bolt1.AddBoltDistX(130);
bolt1.AddBoltDistY(70);

if (bolt1.Insert())
    Console.WriteLine("bolt1 insert failed!");
```



Bolt1, bolt2 생성

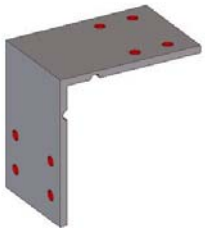
그림 22. 상부 플랜지 플레이트 볼트 입력 데이터

dCollection @ chosun

<하부 플랜지 플레이트>

Beam class

```
TSM.Beam B4 = new TSM.Beam(new Point(450, -100, 3050), new Point(450, 100, 3050));
B4.Position.Rotation = TSM.Position.RotationEnum.TOP;
B4.Position.RotationOffset = 90;
B4.Name = "PLATE";
B4.Profile.ProfileString = "L300*300*15";
B4.Material.MaterialString = "SS400";
B4.Class = "1";
B4.Finish = "PL_Bottom";
if (B4.Insert())
    Console.WriteLine("Connection2 insert failed!");
```



B4 생성

ContourPlate class

```
TSM.ContourPlate B4_Slotted2 = new TSM.ContourPlate();
TSM.ContourPoint point44 = new TSM.ContourPoint(new Point(165, 0, 3135), null);
TSM.ContourPoint point45 = new TSM.ContourPoint(new Point(157.93, 0, 3132.07), null);
TSM.ContourPoint point46 = new TSM.ContourPoint(new Point(155, 0, 3125), null);
TSM.ContourPoint point47 = new TSM.ContourPoint(new Point(157.93, 0, 3117.93), null);
TSM.ContourPoint point48 = new TSM.ContourPoint(new Point(165, 0, 3115), null);

B4_Slotted2.AddContourPoint(point44);
B4_Slotted2.AddContourPoint(point45);
B4_Slotted2.AddContourPoint(point46);
B4_Slotted2.AddContourPoint(point47);
B4_Slotted2.AddContourPoint(point48);


B4_Slotted2.Profile.ProfileString = "PL200";
B4_Slotted2.Class = TSM.BooleanPart.BooleanOperativeClassName;
if (!B4_Slotted2.Insert())
    Console.WriteLine("B4_Slotted2 inser failed");

TSM.BooleanPart boolpart10 = new TSM.BooleanPart();
boolpart10.Father = B4;
boolpart10.SetOperativePart(B4_Slotted2);
if (!boolpart10.Insert())
    Console.WriteLine("boolpart10 insert failed!");

B4_Slotted2.Delete();
```

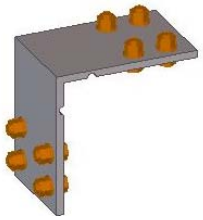
그림 23. 하부 플랜지 플레이트 입력 데이터

dCollection @ chosun

<하부 플랜지 플레이트 볼트>

BoltArray class

```
TSM.BoltArray bolt3 = new TSM.BoltArray();

bolt3.PartToBoltTo = B1;
bolt3.PartToBeBolted = B4 as TSM.Beam;
bolt3.FirstPosition = new Point(185, -65, 2985);
bolt3.SecondPosition = new Point(185, 75, 2985);

bolt3.BoltSize = 20;
bolt3.BoltStandard = "H.T.B";
bolt3.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt3.Tolerance = 2;
bolt3.CutLength = 150;
bolt3.Length = 60;
bolt3.ExtraLength = 0;

bolt3.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt3.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt3.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt3.Position.Rotation = TSM.Position.RotationEnum.TOP;
bolt3.Position.RotationOffset = 90;
bolt3.Position.Rotation = TSM.Position.RotationEnum.BACK;
bolt3.Position.RotationOffset = 90;


bolt3.AddBoltDistX(130);
bolt3.AddBoltDistY(70);

if (bolt3.Insert())
    Console.WriteLine("bolt3 insert failed!");
```



Bolt3, bolt4 생성

그림 24. 하부 플랜지 플레이트 볼트 입력 데이터

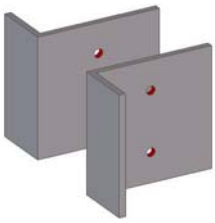
dCollection @ chosun

<웨브 플레이트>

Beam class

```
TSM.Beam B5 = new TSM.Beam(new Point(300, -150, 3250), new Point(300, -150, 3550));
B5.Position.Rotation = TSM.Position.RotationEnum.TOP;
B5.Position.RotationOffset = 90;
B5.Name = "PLATE";
B5.Profile.ProfileString = "L140*300*15";
B5.Material.MaterialString = "SS400";
B5.Class = "1";
B5.Finish = "PL_HOR";
if (B5.Insert())
    Console.WriteLine("Connection3 insert failed!");
```

```
TSM.Beam B6 = new TSM.Beam(new Point(300, 4, 3250), new Point(300, 4, 3550));
B6.Position.Rotation = TSM.Position.RotationEnum.TOP;
B6.Position.RotationOffset = 90;
B6.Position.Rotation = TSM.Position.RotationEnum.BACK;
B6.Position.RotationOffset = 90;
B6.Name = "PLATE";
B6.Profile.ProfileString = "L300*140*15";
B6.Material.MaterialString = "SS400";
B6.Class = "1";
B6.Finish = "PL_HOR";
if (B6.Insert())
    Console.WriteLine("Column2 insert failed!");
```



B5, B6 생성

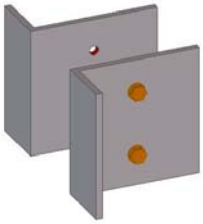
그림 25. 웨브 플레이트 입력 데이터

dCollection @ chosun

<웨브 플레이트 볼트>

BoltArray class

```
TSM.BoltArray bolt7 = new TSM.BoltArray();

bolt7.PartToBoltTo = B5;
bolt7.PartToBeBolted = B6 as TSM.Beam;
bolt7.FirstPosition = new Point(300, 0, 3475);
bolt7.SecondPosition = new Point(300, 0, 3325);

bolt7.BoltSize = 20;
bolt7.BoltStandard = "H.T.B";
bolt7.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt7.Tolerance = 2;
bolt7.CutLength = 150;
bolt7.Length = 60;
bolt7.ExtraLength = 0;

bolt7.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt7.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt7.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt7.Position.Rotation = TSM.Position.RotationEnum.BACK;
bolt7.Position.RotationOffset = 90;

bolt7.AddBoltDistX(150);
bolt7.AddBoltDistY(0);


if (bolt7.Insert())
    Console.WriteLine("bolt7 insert failed!");
```



bolt7 생성

그림 26. 웨브 플레이트 볼트 입력 데이터

dCollection @ chosun

5. 조립·해체가 자유로운 주각부의 Visual C# 입력데이터

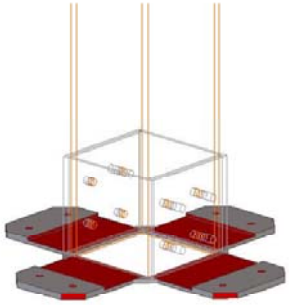
<베이스 플레이트>

Beam class

```
TSM.Beam B5 = new TSM.Beam(new Point(-135, -150, 10), new Point(-135, -450, 10));
B5.Position.Rotation = TSM.Position.RotationEnum.TOP;
B5.Position.RotationOffset = 90;
B5.Name = "PLATE";
B5.Profile.ProfileString = "PL20*270";
B5.Material.MaterialString = "SS400";
B5.Class = "1";
B5.Finish = "PL_basePL1";
if (B5.Insert())
    Console.WriteLine("B5 insert failed!");
```

ContourPlate class

```
TSM.ContourPlate B5_1 = new TSM.ContourPlate();
TSM.ContourPoint point23 = new TSM.ContourPoint(new Point(-135, -400, 10), null);
TSM.ContourPoint point24 = new TSM.ContourPoint(new Point(-135, -450, 10), null);
TSM.ContourPoint point25 = new TSM.ContourPoint(new Point(-85, -450, 10), null);

B5_1.AddContourPoint(point23);
B5_1.AddContourPoint(point24);
B5_1.AddContourPoint(point25);

B5_1.Profile.ProfileString = "PL20";
B5_1.Class = TSM.BooleanPart.BooleanOperativeClassName;
if (!B5_1.Insert())
    Console.WriteLine("B5_1 insert failed");

TSM.BooleanPart boolpart7 = new TSM.BooleanPart();
boolpart7.Father = B5;
boolpart7.SetOperativePart(B5_1);
if (!boolpart7.Insert())
    Console.WriteLine("Boolpart7 insert failed!");

B5_1.Delete();
```



B5(4ea) 생성

B5_1(4ea) 손상 유도부 컷팅

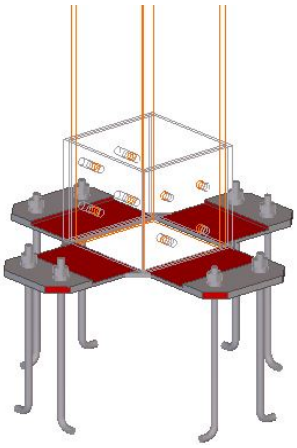
그림 17. 베이스 플레이트 입력 데이터

dCollection @ chosun

<앵커로드>

Beam class

```
TSM.Beam B40 = new TSM.Beam(new Point(-386, 82.5, 80), new Point(-386, 82.5, -400));
B40.Position.Rotation = TSM.Position.RotationEnum.TOP;
B40.Position.RotationOffset = 180;
B40.Name = "ANCHORROD";
B40.Profile.ProfileString = "D22";
B40.Material.MaterialString = "SS400";
B40.Class = "1";
B40.Finish = "ANCHORROD";
if (B40.Insert())
    Console.WriteLine("B40 insert failed!");

TSM.Beam B28 = new TSM.Beam(new Point(-353, 82.5, 20), new Point(-353, 82.5, 31));
B28.Position.Rotation = TSM.Position.RotationEnum.TOP;
B28.Position.RotationOffset = 180;
B28.Name = "WASHER";
B28.Profile.ProfileString = "PD44*11";
B28.Material.MaterialString = "SS400";
B28.Class = "1";
B28.Finish = "WASHER";
if (B28.Insert())
    Console.WriteLine("B28 insert failed!");


TSM.Beam B29 = new TSM.Beam(new Point(-375, 82.5, 31), new Point(-375, 82.5, 53));
B29.Position.Rotation = TSM.Position.RotationEnum.TOP;
B29.Position.RotationOffset = 180;
B29.Name = "NUT";
B29.Profile.ProfileString = "HXGON44-44";
B29.Material.MaterialString = "SS400";
B29.Class = "1";
B29.Finish = "NUT";
if (B29.Insert())
    Console.WriteLine("B29 insert failed!");
```



B40(8ea) 생성

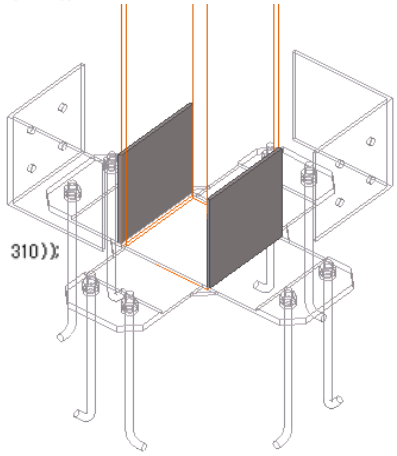
그림 18. 앵커로드 입력 데이터

dCollection @ chosun

<기둥 플레이트>

Beam class

```
TSM.Beam B15 = new TSM.Beam(new Point(-155, -150, 10), new Point(-155, -150, 310));
B15.Position.Rotation = TSM.Position.RotationEnum.TOP;
B15.Position.RotationOffset = 90;
B15.Name = "PLATE";
B15.Profile.ProfileString = "PL10*300";
B15.Material.MaterialString = "SS400";
B15.Class = "1";
B15.Finish = "PL_base1";
if (B15.Insert())
    Console.WriteLine("B15 insert failed!");


TSM.Beam B17 = new TSM.Beam(new Point(155, -150, 10), new Point(155, -150, 310));
B17.Position.Rotation = TSM.Position.RotationEnum.TOP;
B17.Position.RotationOffset = 90;
B17.Name = "PLATE";
B17.Profile.ProfileString = "PL10*300";
B17.Material.MaterialString = "SS400";
B17.Class = "1";
B17.Finish = "PL_base1";
if (B17.Insert())
    Console.WriteLine("B17 insert failed!");
```



B15, B17 생성

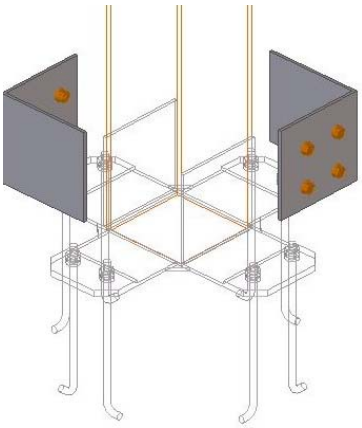
그림 19. 기둥 플레이트 입력 데이터

dCollection @ chosun

<기둥 구속 플레이트 & 볼트>

Beam class

```
TSM.Beam B18 = new TSM.Beam(new Point(-160, 5,10), new Point(-160, 5, 310));
B18.Profile.ProfileString = "L330*310*10";
B18.Name = "PLATE";
B18.Material.MaterialString = "SS400";
B18.Class = "1";
B18.Finish = "PL_base3";
if (B18.Insert())
    Console.WriteLine("B18 insert failed!");

TSM.Beam B19 = new TSM.Beam(new Point(170, 0, 10), new Point(170, 0, 310));
B19.Position.Rotation = TSM.Position.RotationEnum.TOP;
B19.Position.RotationOffset = 180;
B19.Name = "PLATE";
B19.Profile.ProfileString = "L340*320*10";
B19.Material.MaterialString = "SS400";
B19.Class = "1";
B19.Finish = "PL_base2";
if (B19.Insert())
    Console.WriteLine("B19 insert failed!");
```



B18, B19 생성

BoltArray class

```
TSM.BoltArray bolt2 = new TSM.BoltArray();

bolt2.PartToBoltTo = B19;
bolt2.PartToBeBolted = B15 as TSM.Beam;
bolt2.PartToBeBolted = B1 as TSM.Beam;
bolt2.FirstPosition = new Point(-100, 0, 80);
bolt2.SecondPosition = new Point(-100, 0, 230);

bolt2.BoltSize = 20;
bolt2.BoltStandard = "H.T.B";
bolt2.BoltType = TSM.BoltGroup.BoltTypeEnum.BOLT_TYPE_SITE;
bolt2.Tolerance = 2;
bolt2.CutLength = 150;
bolt2.Length = 60;
bolt2.ExtraLength = 0;

bolt2.ThreadInMaterial = TSM.BoltGroup.BoltThreadInMaterialEnum.THREAD_IN_MATERIAL_YES;

bolt2.Position.Depth = TSM.Position.DepthEnum.MIDDLE;
bolt2.Position.Plane = TSM.Position.PlaneEnum.MIDDLE;
bolt2.Position.Rotation = TSM.Position.RotationEnum.FRONT;
bolt2.Position.RotationOffset = 180;

if (bolt2.Insert())
    Console.WriteLine("bolt2 insert failed!");
```

Bolt2, bolt3 생성

그림 20. 기둥 구속 플레이트 & 볼트 입력 데이터

dCollection @ chosun

6. 부재 요소별 물량 데이터

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
MC2	[]300*300*10	14400.0	8	SS400	138.4	0.0	1311.3	10490.4
MC3	[]300*300*10	14400.0	1	SS400	17.3	0.0	1311.3	1311.3
MC4	[]300*300*10	14400.0	2	SS400	34.6	0.0	1311.3	2622.6
MC6	[]300*300*10	14400.0	1	SS400	17.3	0.0	1311.3	1311.3
				Sub Total:	207.6			15735.6
				Total :	207.6			15735.6

그림 27. 기둥 물량 산출

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
PL5	L140*300*15	300.0	80	SS400	24.0	0.0	15.0	1200.0
PL6	L140*300*15	300.0	48	SS400	14.4	0.0	15.0	720.0
PL7	L140*300*15	300.0	48	SS400	14.4	0.0	15.0	720.0
PL8	L140*300*15	300.0	80	SS400	24.0	0.0	15.0	1200.0
				Sub Total:	76.8			3840.0
PL1	L300*300*15	200.0	60	SS400	18.0	0.0	13.8	828.0
PL2	L300*300*15	200.0	36	SS400	10.8	0.0	13.8	496.8
PL3	L300*300*15	200.0	80	SS400	24.0	0.0	13.8	1104.0
PL4	L300*300*15	200.0	48	SS400	14.4	0.0	13.8	662.4
				Sub Total:	67.2			3091.2
				Total :	144.0			6931.2

그림 28. 접합부 물량 산출

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
AN1	D22	519.8	96	SS400	0.0	0.0	1.5	144.0
				Sub Total:	0.0			144.0
NUT1	HXGON44-44	22.0	96	SS400	0.0	0.0	0.2	19.2
				Sub Total:	0.0			19.2
PL18	L330*310*10	300.0	12	SS400	4.8	0.0	14.8	177.6
				Sub Total:	4.8			177.6
PL19	L340*320*10	300.0	12	SS400	4.8	0.0	15.3	183.6
				Sub Total:	4.8			183.6
WS1	PD44*11	11.0	96	SS400	0.0	0.0	0.1	9.6
				Sub Total:	0.0			9.6
				Total :	9.6			534.0

그림 29. 주각부 물량 산출

dCollection @ chosun

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
DP5	D25	100.0	32	SS400	0.0	0.0	0.4	12.8
				Sub Total:	0.0			12.8
DP2	T112-20-20-200	400.0	8	SS400	2.4	0.0	18.3	146.4
				Sub Total:	2.4			146.4
DP4	T165-20-20-200	400.0	8	SS400	2.4	0.0	21.7	173.6
				Sub Total:	2.4			173.6
BR1	U100*50*5	2686.5	32	SS400	32.0	0.0	20.0	640.0
				Sub Total:	32.0			640.0
				Total :	36.8			972.8

그림 30. 전단형 마찰댐퍼와 K형 브레이스 물량 산출

TeklaStructures PART SUMMARY LIST — Page: 1
Project name :Smart Green Structur — Phase:
Date 23.06.2014

mark	Profile	length(mm)	Q'ty	Grade	Area(m²)	Unit(kg/m)	Weight/EA(kg/m)	Weight(kg/m)
BR_pol	H200*200*8*12	4540.1	7	SS400	37.1	49.9	226.5	1585.5
BR_pol	H200*200*8*12	4540.1	7	SS400	37.1	49.9	226.5	1585.5
MB13	H200*200*8*12	3800.0	2	SS400	8.8	49.9	189.6	379.2
MC5	H200*200*8*12	3200.0	16	SS400	59.2	49.9	159.7	2555.2
P24	H200*200*8*12	4540.1	2	SS400	10.6	49.9	226.5	453.0
				Sub Total:	152.8			6558.4
				Total :	152.8			6558.4

그림 31. 원형 브레이스 물량 산출

TeklaStructures MAIN BOLT SUMMARY LIST — Page 1
Project name :Smart Green Structur — Phase
Date 23.06.2014

Size	Name	length	Q'ty
M 16	H.T.B	65	188
M 20	H.T.B	50	64
M 20	H.T.B	60	1920
M 20	H.T.B	65	992
M 20	H.T.B	70	128
M 20	H.T.B	75	288
M 20	H.T.B	110	16

그림 32. 볼트 수량 산출

dCollection @ chosun

7. 자동 도면 생성

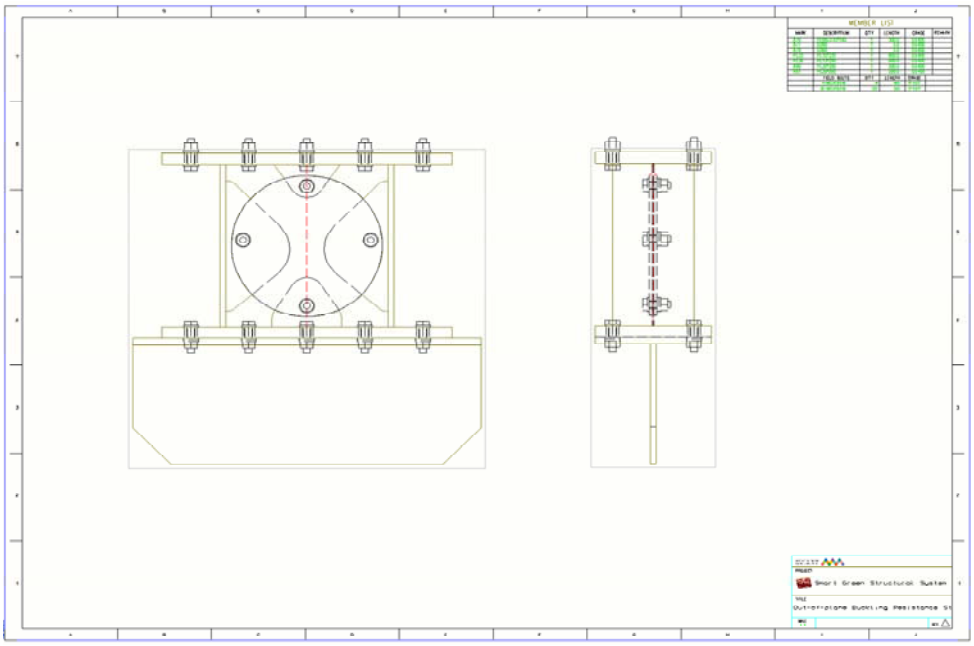


그림 33. 면외좌굴구속 강재댐퍼 도면

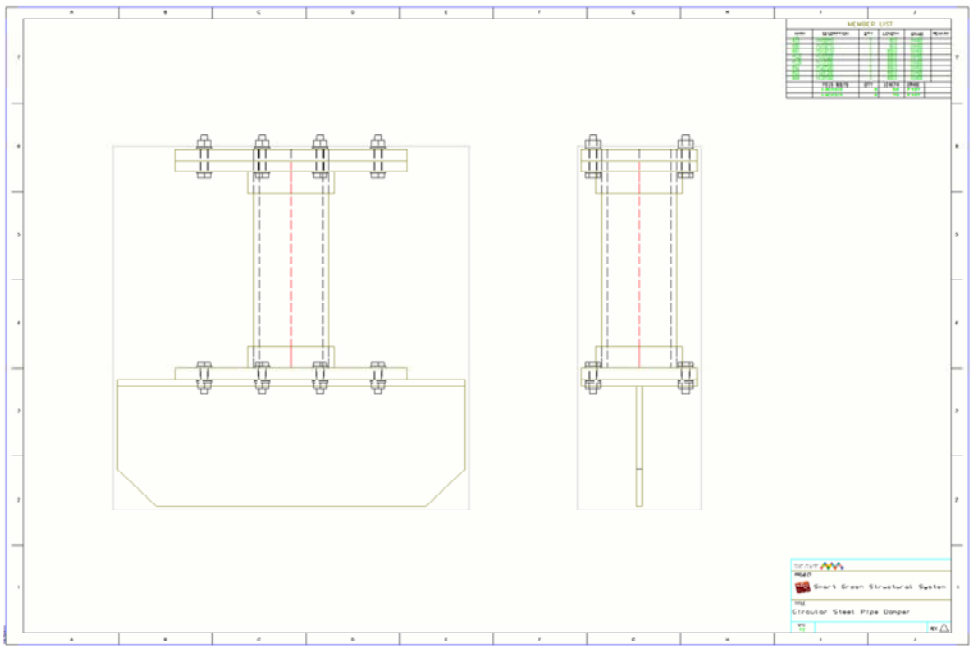


그림 34. 원형강관댐퍼 도면

dCollection @ chosun

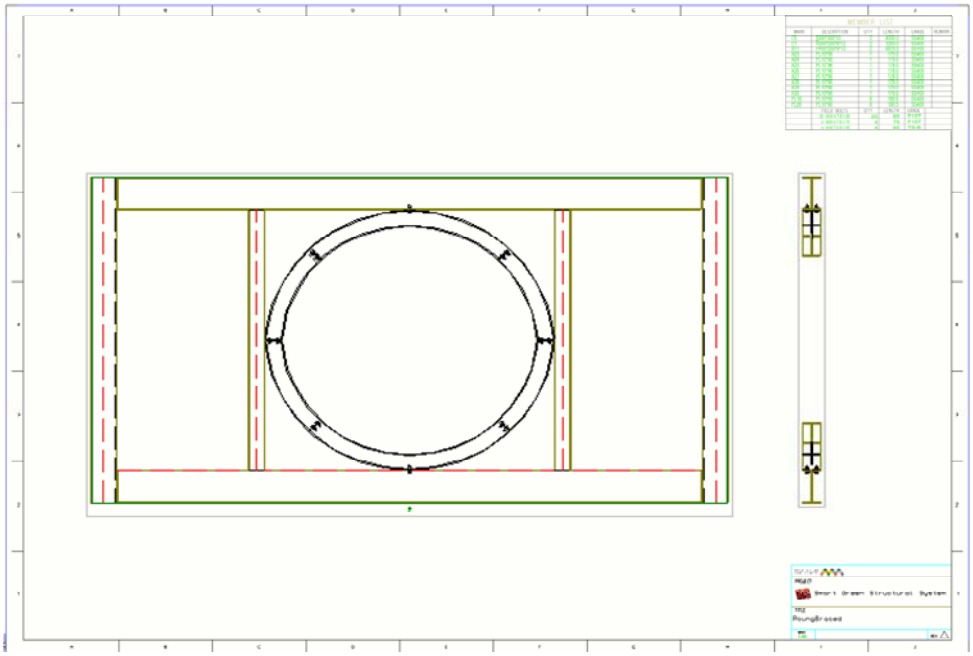


그림 35. 원형 브레이스 도면

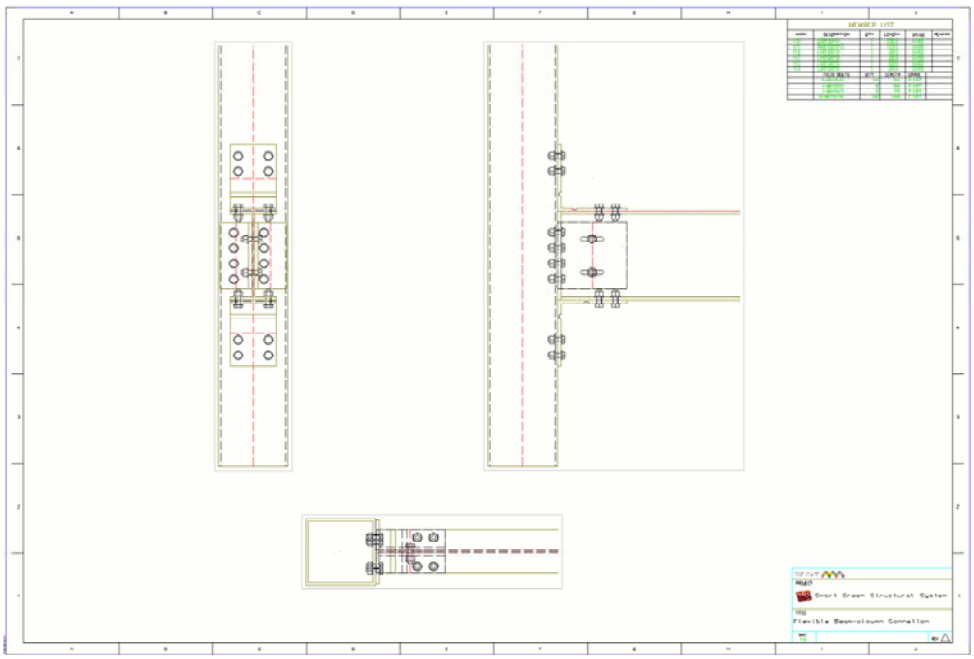


그림 36. Flexible한 보-기둥 접합부 도면

dCollection @ chosun

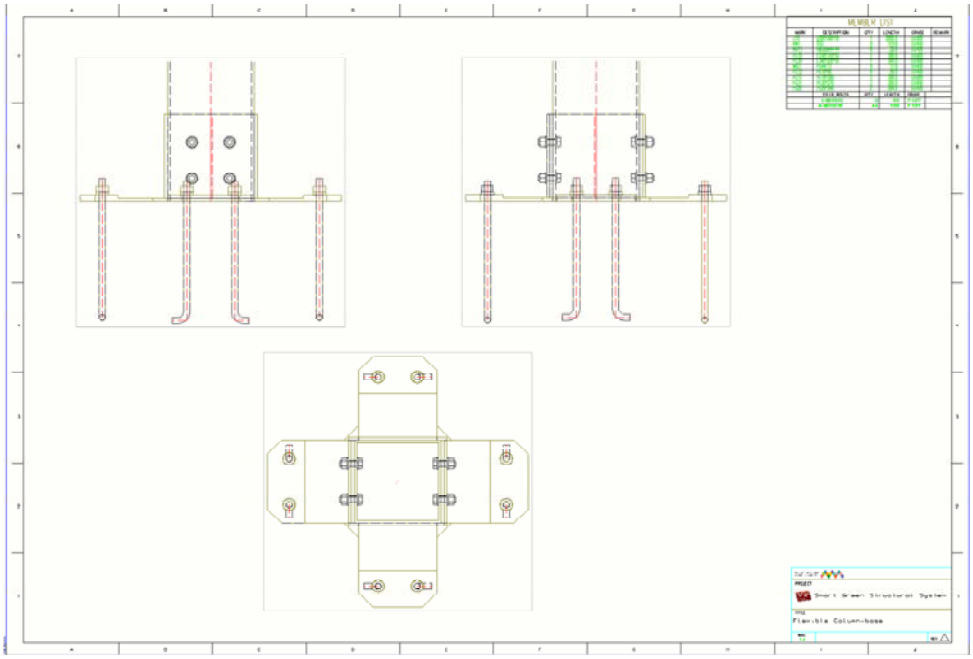


그림 37. 조립・해체가 자유로운 주각부 도면

dCollection @ chosun