

Global Report

자동적인 건물 정보 모델 쿼리 생성

1. 서론
2. 관련 연구
3. 설계
4. 실행
5. 사례 연구
6. 한계
7. 결론 및 향후 연구

1. 서론

건물의 설계와 시공에 있어서 내제된 문제점은 서로 다른 분야들 간의 복잡한 협력체계이다(Lee et al. 2012). 에너지 효율적 건물(Energy Efficient Building, EEB)의 설계와 시공은 기존 건물의 설계와 시공에 비해 더 많은 협력자들이 연관되게 된다. 건축물의 에너지 효율성을 구축하기 위해서는 서로 다른 분야의 팀들이 일련의 에너지 시뮬레이션과 분석을 수행할 필요가 있다. 대부분의 EEB 설계자들과 분석가들은 분석 모델을 개발하는데 많은 시간을 소비한다. Bazjanac(2001)에 따르면 그들은 입력 데이터 구축에 50% 이상의 시간을 소비하는데, 그 시간 중에서도 다시 80% 이상을 기하 모델(geometric model) 생성에 소비한다. 건물정보모델링(Building Information Modelling, BIM)은 EEB 설계와 시공 분야에서 정보 공유 문제에 대한 해결능력(potential solution)을 가지고 있다. 많은 경우에서, EEB 설계자들과 분석가들은 여러 다양한 팀들로부터의 단편적인 데이터가 필요할 수 있다. 이러한 세밀한 객체 지향적(Object-Oriented) 표현방식은 건설 모델들이 사용자들의 요구에 따라 통합되기도 하고 분리될 수도 있다는 것을 의미한다. 이러한 특징으로 인해 EEB 설계자들과 분석가들은 모든 BIM 데이터를 통합하는 마스터 BIM 허브로부터 부분모델 정보요청(query : 정보 수집에 대한 요청 작업을 일컫는 전산 용어, 역주) 작업을 수행하게 된다.

이러한 BIM 부분모델 정보요청작업 수행에는 2가지 방식이 있다. 첫 번째 방식은 기존의 일반적 프로그래밍언어 또는, JAVA나 SQL과 같은 정보요청(query)언어들을 활용하는 것이다. 하지만 건축, 공학 및 건설분야(AEC Community)대부분의 사용자로 하여금 이러한 프로그래밍 언어를 완벽히 습득하도록 하는 것은 BIM에서 부분모델 정보요청 과업수행에 대한 비용을 증가시키는 한편, 컴퓨터 프로그래머들은 이 분야의 전반적인 작업절차에 대한 지식이 없고, 구조물 설계 요건들에 대한 포괄적인 이해가 부족하다. 더욱이, 사용자가 일련의 정보요청 체계를 작성하기 위해서는 반드시 BIM의 내부 데이터 조직을 이해하여야만 하는데, 이는 느슨한 결합원칙(loose-coupling principle, IT분야에서 향후 확장성을 고려하여 각 정보전달 체계를 너무 엄격하게 정하지 않는 원칙, 역주)에 위배되는 문제점이 발생한다.

두 번째 방식은 BIM 데이터를 호출하는 GUI(그래픽 사용자 인터페이스, 역주)기반의 필터를 사용한다. 이러한 GUI 기반의 적용방식은 쉽게 사용할 수 있으나, 제한된 수의 기본적인 수행용법에 한에서만 지원할 수 있다. EEB 시뮬레이션과 분석은 대단히 특성화(Specific)경향이 있어서, 인수(arguments)의 개수와 유형들이 경우에 따라 조정되어야 한다. 그래서 이러한 일련의 작업 절차는 EEB 작업에서 모든 부분모델 정보요청(query)작업 요건들을 지원하기에는 불충분하다.

이 논문에서는 기존의 방식들의 한계들을 극복하기 위한 새로운 방법을 제안한다. 이 방식은 부분모델 정보요청 코드를 반자동(semi-automatically)으로 생성하는 “모델 뷰 개념”(Model View Definition, MVD)을 사용한다. 이 새로운 기법에서는 사용자가 세부적인 BIM 내부 데이터 구조와 프로그래밍 언어를 파악할 필요가 없다. 보다 구체적으로 말하면, 최초로 MVD에서 데이터 스키마(데이터 구조를 정의하는 개념, Schema)와 제약사항(Constraints)들이 전단부(front-end)에서 분석된다. 그 다음으로 중간 표현구조(Intermediate Representation, IR)가 템플릿(미리 준비된 여러 형태의 표본들, 역주)을 기반으로 형태를 정하는 방법에 기초하여 생성된다. 마지막으로 이 중간 표현구조(IR)는 BIM 데이터 허브의 실행에 기반을 둔 특정 프로그램 언어의 부분모델 정보요청 코드로 변환된다.

본 연구진은 공개 프로그램(Open Source)인 BIMserver(Beetz et al. 2010)에 기초한 QueryGenerator라

불리는 시제품을 개발하였으며, 이를 실제 다중 영역 공기흐름 분석에 적용하여 효율적이고 정확하게 과업을 완료하는데 도움을 받았다.

2. 관련 연구

2.1. BIM: 데이터 스키마와 어플리케이션들

BIM 데이터는 기존의 몇 가지 개방형 표준이나 모델링 언어들에 의해 묘사되고, 모델링되며 변환될 수 있다. EXPRESS 데이터 모델링 언어는 ISO 10303에 규정되어 있다(Pratt, 2005). EXPRESS 언어는 정의(define)가 가능한 제품 데이터의 측면(aspect)을 명시한다. 산업 기반 클래스(Industry Foundation Classes, IFC)는 국제적인 개방형 표준이자 중립적인 플랫폼 데이터 모델이다(buildingSMART International Ltd. 2014a). 모델 뷰 개념(Model View Definition, MVD)은 AEC 산업에서 몇 가지 특수한 시나리오에 대한 교환 요건들(exchange requirements)을 구체화하기 위한 IFC의 부차적 체계(subset)를 규정한다(buildingSMART International Ltd. 2014b). 모델 뷰 개념(Model View Definition, MVD)들은 MVDXML 형식으로 인코딩될 수 있는데, 이 형식은 모델 뷰 개념에서 “특정 데이터 형식의 특정한 성질에 대한 허용 값”을 규정한다.(buildingSMART International Ltd. 2014b). Green Building XML (gbXML) 개방형 스키마는 BIM에서 EEB 분석도구들로 데이터 전송을 원활하게 하기 위해 설계되었다.(gbXML.org, 2014). gbXML은 다양한 공개 프로그램(Open Source)들, 상용 BIM 플랫폼 제품들 및 상용 분석도구들에 통합되어 왔다.

Berlo와 Laat(2010)는 BIM과 지형공간 정보시스템(GIS, Geospatial Information Systems)의 통합을 연구했다. Chen(2011)은 모바일 기기에 의해 지원되는 BIM 기반 협업 플랫폼에 대한 개념적 기틀(Framework)을 보고했다. Kim등(2011)은 설계 과정에 대한 보다 상세한 이해를 제공할 수 있고 의사결정 근본원인에 쉽게 접근할 수 있는 BIM 기반의 시각화(Visualization)도구를 소개했다. Singh등(2011)은 다중 분야들의 협업 플랫폼인 BIM 데이터 허브를 이용하기 위한 기술적 요건들에 대한 이론적 기틀을 개발했다. Liu등(2013; 2014)은 건물 지식 모델링(BKM) 플랫폼을 성취하기 위해 BIM과 지식 경영(Knowledge Management, KM)을 결합하기 위한 해법(Solution)을 제안했다. Beetz 등(201)은 BIM 서버라고 불리는 IFC 데이터 서버를 개발했다. Jiang 등(2012)는 에너지 효율 건물 라이프 사이클을 지원하는 BIM 서버의 요건들에 대한 논의를 개제하였다. Yu등(2013)은 BIM 서버와 OpenStudio 간의 데이터 교환을 용이하게 하고 자동화할 수 있는 통합된 방법을 제안했다. BIM4Geo4는 3D 네비게이션과 시각 환경(viewing environment)에 IFC 내의 BIM 데이터를 통합하기 위해 BIM 서버를 연동했다(Hijazi 등 2010). Qi등(2012)은 BIM 데이터 허브로부터의 조회된 데이터 정보를 이용하여 추락사고 위험성을 점검할 수 있는 도구를 개발했다.

2.2. 부분 모델 정보요청

Mazairac와 Beetz(2013)는 건물 정보 모델 정보요청 언어(Building Information Model Query Language, BIMQL)를 제안했다. BIMQL은 IFC 형식 내의 데이터를 선별, 갱신, 삭제하기 위해 고안되었다. BIMQL의

몇 가지 특징들, 즉 공개 프로그램이고, 도메인을 규정하며 플랫폼과는 무관하다는 특징들이 많은 BIM 정보 추출과 조작의 문제들에 대한 실용적인 접근을 가능하게 했다. BIMQL은 상대적으로 단순한 구문과 사용하기 쉬운 문법으로 되어있다. 하지만 새로운 정보요청 언어로서 JAVA와 같은 다른 일반적 프로그래밍 언어들과는 다르게 사용범위에 제약이 있는 언어이기 때문에 모든 최종 사용자들이 습득하기에는 비용이 소요된다. Daum과 Borrmann(2014)은 BIM을 위한 정보요청 언어(Query Language for Building Information Model, QL4BIM)를 고안했다.

부분 모델 정보요청 언어(PMQL)는 IFC 모델 서버로부터 부분 모델 정보요청을 용이하게 하려는 초창기 연구의 산물이다(Adachi 2003). 구체적으로 PMQL 내의 정보요청 기능은 XML 데이터의 일부분이다. XML 파일은 단순 객체 접근 규약(프로토콜, protocol)(Simple Object Access Protocol, SOAP)을 통해 서로 다른 웹 서비스로 직접 전송될 수 있다. 사용자들은 IFC의 부분 모델들을 선별, 갱신, 삭제하기 위해 PMQL을 사용할 수 있다. Koonce 등(1998)은 EXPRESS 정보요청 언어(EXPRESS Query Language, EQL)를 고안하고 실행했다. 이는 객체 지향적인 EXPRESS 모델링 형식의 파일로부터 데이터를 조회하는데 사용될 수 있다.

몇몇 다른 연구자들은 새로운 언어를 고안하는 대신에 Daas(Data as a Service)를 이용하여 데이터 조회를 쉽게 하려는 시도를 하고 있다.(Dan et al. 2007; Dwivedi and Kulkarni 2008; Truong and Dustdar 2009; Olson 2009). DaaS 모델은 이러한 데이터 수준에서 소프트웨어의 상호 운용성을 촉진하려는 새로운 구조(Architecture)로서, 기존의 데이터 소스에 기반한 데이터를 제공하거나 데이터 제공자들이 데이터를 생성, 검색, 갱신, 삭제할 수 있도록 한다(Truong and Dustdar 2009; Cagle 2010).

3. 설계

3.1. 구조(Architecture)

그림 1은 본 연구의 설계 구조를 보여준다. MVD를 입력치로 하고 정보요청 코드 생성 기틀을 설계하였다. 4개의 개념적 구성요소들은 입력데이터를 어휘적으로 분석하는데 사용되는 전단부(front end), 중간 표현 생성자, 중간 표현 최적화, 그리고 목표 정보요청 코드 생성자이다. 자동 정보요청 코드 생성 과정에는 4단계가 있다. 1단계로 전단부는 먼저 입력 내용으로 EXPRESS, mvdXML, JSON 등의 형식이 될 수 있는 MVD를 받아들인 후, MVD를 분석하고 토큰 스트림(프로그램언어를 기계어로 번역할 때, 번역된 일련의 기계어, token stream)을 도출한다. 오직 전단부의 변경만으로 나머지 부분에 영향을 주지 않고 전체 기틀(Framework)이 확장될 수 있다 (4.2.2절 참조).

2단계로 중간 표현 생성자가 입력내용인 토큰 스트림으로 실행된다. 출력내용은 중간 표현(Intermediate Representation, IR)이다. 이 구성요소는 MVD 포맷, 생성된 정보요청 코드 언어 유형, 조회하기 위한 플랫폼 등에 구애받지 않는다. (4.2.3절 참조).

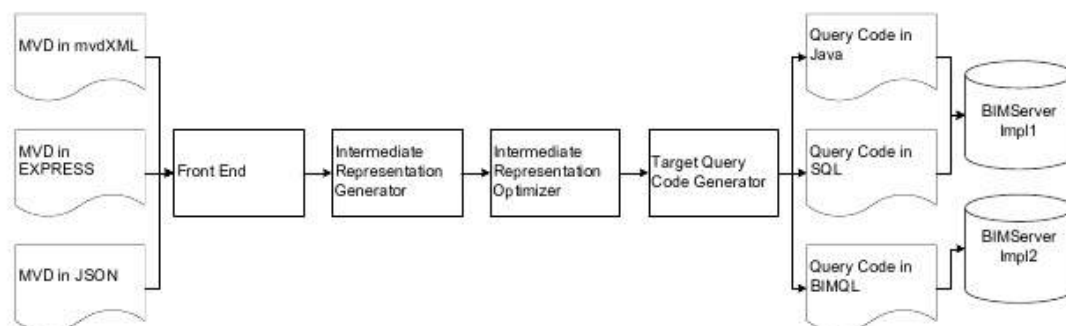
마지막으로 목표 정보요청 코드 생성자는 정보요청 코드를 구축하기 위해 최적화된 IR을 이용한다. 생성된 정보요청 코드는 바로 필요한 데이터를 가져오기 위한 특정 BIM 데이터 허브 구현단계로 보낼 수 있다. 전단부와 유사하게 이 구성요소는 실행 요건(implementation requirements)과 이용 사례(use cases)들의 적용 대상이 된다. 이 단계에서 IR은 사용자의 요건에 따라 Java, BIMQL이나 QL4BIM과 같은 다른

목표 언어들로 변환될 수 있다.

3.2. 입력내용으로서의 MVD

MVD는 특정한 사용 환경에서 모델 정보를 명확하게 정의하기 위해 설계되었다. buildingSMART international은 정보 제공 매뉴얼(Information Delivery Manual, IDM)을 일반 언어로 작성된 일련의 사용자 요구조건에 대한 문서로서 규정한다. buildingSMART international(2014b)에 따르면 “각각의 MVD 컨셉은 소프트웨어 공급 업체들에게 명확한 실행 지침을 제공한다.”고 밝히고 있다. MVD들은 분야(domain)에 대한 설명들과 BIM 데이터 스키마 사이의 연결을 구축하므로 정보요청 코드를 구성하는 명확한 정보의 원천(source)으로 사용될 수 있다. 구체적으로 특정분야의 전문가들이 먼저 MVD를 고찰하고 해석하여 관심사항들에 대한 MVD 개념들을 규정할 것이다. AEC 위원회에 몇 가지 잘 정의된 MVD들이 존재한다. 사용자들은 필요에 따라 그것들을 재사용할 수 있다.

MVD를 입력 자료로 사용하는 중요한 장점은 AEC 위원회의 기존 자산들을 이용한다는 것이다. buildingSMART international(2014b)에 따르면 많은 MVD들이 개발되고 배포되었다. 예를 들면 IFC4 Reference View, IFC4 Design Transfer View, IFC2X3 Coordination View, Space Boundary Addon View, Basic FM Handover View, Structural Analysis View 등이 있다.

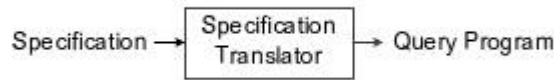


<그림 1. 자동 부분 모델 정보요청 생성 기틀 설계>

3.3. 사양 번역기

본 장에서는 프로그래밍 언어(PL)로부터 자동 정보요청 코드 생성을 분석한다. 사양 번역기는 프로그램 P(Program)로부터 사양(specification)S를 번역한다. 그림 2는 사양 번역기의 모델을 보여준다. 이론적 관점에서 본 연구 모델의 기틀은 이 모델에 속한다. 입력 내용을 사양으로 적용하는 것은 BIM의 부분 모델 정보요청의 맥락에서 몇 가지 장점들을 갖는다. 먼저 대부분의 최종 사용자들에게는 프로그래밍에 비해 사양을 명시하는 것이 더 용이하다. 사양의 명시(Specification)는 일반적으로 설명적(description) 언어로 기술되나, 많은 프로그래밍 언어들은 명령적(imperative)이다. 입력 내용인 사양을 이용하여 최종 사용자들은 “어떻게” 프로그래밍을 하는지 보다는 필요한 “무엇”만을 규정하면 된다. 반면에 자연언어(natural language)에 비해 설명적 언어는 일반적으로 모호하지 않다. 특정 법칙들을 적용하여 필요 사양

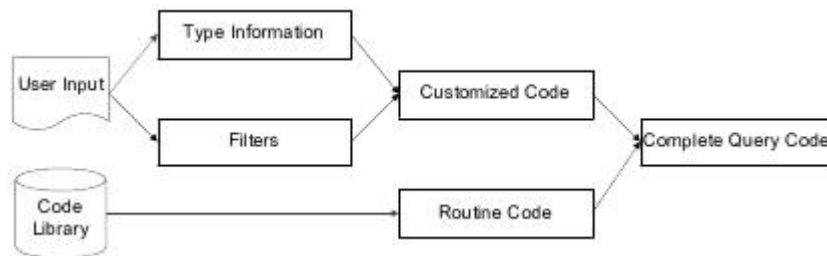
(Specification)은 기계에 의해 분석되고 번역될 수 있다.



<그림 2. 사양 번역기의 모델>

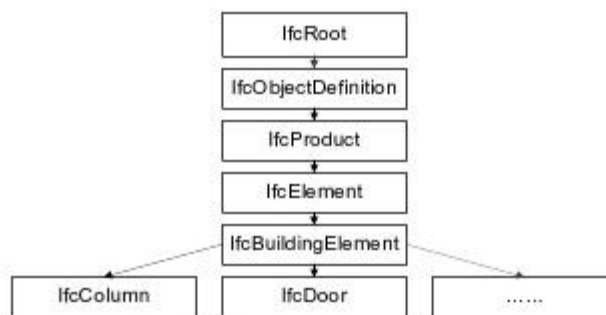
우리가 생성하려는 프로그램은 정보요청 코드이다. 이러한 코드 생성 과정은 두 가지 부분으로 나뉜다. 첫 번째는 사용자에게 의해 구체화될 필요가 있는 코드를 생성하는 것이다. 나머지는 루틴 코드(routine code)를 생성하는 것이다. 사용자들(즉, 분야의 전문가들)에 의해 구체화되어야 하는 코드는 두 가지 정보 유형들이 구체화되어야 한다. 하나는 문, 벽, 기둥과 같은 유형 정보이고, 다른 하나는 특수한 속성들(Attributes)에 대한 제약사항(Constraints)들이다.

루틴 코드의 생성에는 “패턴 매치” 방법이 사용된다. 각 정보요청 패턴에 부합하는 루틴 코드 템플릿이 루틴 코드 풀(pool)에 저장되어 있다. 어떤 시나리오들이 탐색되면 입력 내용인 MVD에 따라 매칭 루틴 코드 템플릿이 그 풀로부터 로딩 되어 완전한 정보요청 코드를 생성하기 위해 제작된 코드와 조합된다 (그림 3).



<그림 3. 루틴 코드와 제작(Customized) 코드>

여기서 “패턴”이 IFC/MVD 객체 분류를 일컫는다는 것은 주목할 만하다. 예를 들면 그림 4에서와 같이 거의 모든 정식 건물 구성재들의 IFC 클래스들은 *IfcBuildingElement*의 하위클래스라는 것이다. 게다가 그들은 개괄적 건물 도식(an abstract building graph)에서 동일한 위상(topology)을 갖고 있으므로 (그림 5), 그들은 동일한 루틴 코드를 공유할 수 있다. 서로 다른 루틴 코드들은 서로 다른 IFC 상속 클래스 위계와 개괄적 건물 도식(an abstract building graph)에서 위치가 다를 경우에 대해서만 개발되어야 한다. 이러한 두 가지 관계들을 보여주는 그림 4와 그림 5의 차이는 대단히 중요하다. 그림 4는 IFC 데이터 스키마에서 “~는 ~이다.”의 관계를 보여준다. 예를 들면 *IfcDoor*는 *IfcBuildingElement*이다. 그림 4는 전형적인 건물 구조에서 “~을 갖는다”의 관계를 보여준다. 예를 들면 *IfcBuildingStorey*는 하나 이상의 *IfcDoor(s)*를 갖는다. 이러한 두 가지 관계는 “패턴 매치”법의 기초이자 객체지향 설계의 기초적인 관계이다.



<그림 4. IFC 데이터 스키마에서 IfcDoor 상속 클래스의 위계>



<그림 5. 전형적인 건물 구조>

3.4. 추가(Advanced) 필터

경우에 따라서 사용자들은 추가로 몇 가지 객체들을 부분 모델들로부터 속성 값에 기초하여 선택하기를 원할 수 있다. 예를 들면 사용자들이 건물의 모든 문에 대한 정보를 확보하기 보다는, 높이가 2미터 이상인 문에 대한 정보를 원할 수 있다. 그러한 경우에 구체적 객체 이름과 몇 가지 개략적인 제약사항들만으로는 불충분하다. 프로그래밍 언어에서 조건명령문(conditional judgement)을 수행하기 위한 술어들(predicates, 예 “ $x > 2$ ”)을 이용하는 것과는 달리, 사양 데이터 스키마(Specification data schema)에서는 사양에 기반을 둔 방식으로 필터(filter)들을 도입한다. 그림 6은 JAVA로 작성된 정보요청 코드의 사례를 보여준다. 진하게 표시된 코드는 사용자가 입력한 필터를 기반으로 생성된 것이다. 사용자들의 입력 데이터 스키마에서 몇 가지 특별한 속성값 들에 기초한 정보요청 구문 내에 조건명령문을 생성할 필요가 있다. 그것이 어떻게 작동하는지 보여주기 위해 JSON을 이용한다.

```

@Override
public void query(IfcModelInterface model, PrintWriter out) {
    out.println("Running doors example");
    List<IfcBuildingStorey> stories = model.getAll(IfcBuildingStorey.class);
    Map<Double, IfcBuildingStorey> orderedStories = new TreeMap<Double, IfcBuildingStorey>();
    for (IfcBuildingStorey storey : stories) {
        orderedStories.put(storey.getElevation(), storey);
    }
    if (orderedStories.size() > 1) {
        IfcBuildingStorey firstFloor = stories.get(1);
        for (IfcRelContainedInSpatialStructure rel : firstFloor.getContainsElements()) {
            for (IfcProduct product : rel.getRelatedElements()) {
                if (product instanceof IfcDoor) {
                    IfcDoor ifcDoor = (IfcDoor)product;
                    if (ifcDoor.getOverallHeight() > 2) {
                        out.println(ifcDoor.getName() + " " + ifcDoor.getOverallHeight());
                    }
                }
            }
        }
    }
}
  
```

<그림 6. BIM 서버의 추가된 정보요청 사례>

```

{ "doors": [
  { "height": {
    "type": "integer",
    "unit": "metres",
    "minimum": 2
  }}
]}

```

<그림 7. 공기흐름 분석을 위한 JSON 입력 사례>

그림 7은 JSON에서 문의 최소 높이를 지정한 단순화된 IfcDoor 데이터 스키마를 보여준다.

4. 실행

4.1. BIM 서버의 정보요청 메카니즘

BIM 서버의 초기 입력 파일들은 IFC와 IFCXML이다. 이 파일들은 Berkley DB에 저장된다. BIM 서버는 이클립스 모델링 프레임워크(Eclipse Modelling Framework, EMF)를 통해 각각의 IFC 클래스를 부합하는 자바(Java) 클래스 안에서 실행된다(Steinberg et al. 2008). BIM 서버가 가동하기 시작하면 데이터베이스로부터 IFC 객체들의 데이터를 읽어서 그것들을 자바 객체들로 구체화(instantiate)한다. 다양한 건축 요소들을 표현하는, 이러한 자바 객체들은 도식화된 건물을 구성하는 절점(node)으로 메모리에 저장되는데 이러한 도식화된 건물은 사전 정의된 여러 관계들을 나타내는 형태로 연결되어 있다. 이러한 구조물의 도식화는 구조물의 BIM 개요(abstraction)가 된다. BIM 서버의 정보요청 코드는 자바로 기록된다. 정보요청 코드의 주요 논리구조는 건물 요소 도식(graph)에 대한 탐색을 수행하는 것이다.

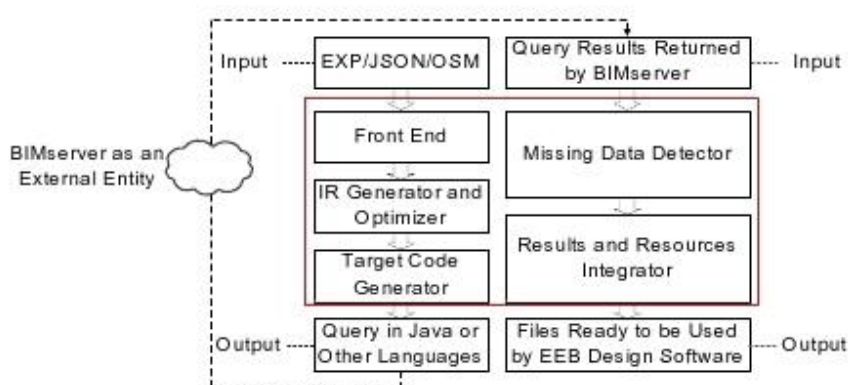
사용자가 BIM 서버에 등록한 자바 정보요청 코드는 정보요청 상호작용을 실행하는 자바 클래스이다. 이러한 정보요청 코드는 컴파일 되어 실행되기 위해 BIM 서버로 보내진다. 클래스들의 명칭들은 표 1과 같다.

<표 1. org.bimserver.querycompiler 패키지 내의 자바 클래스들>

Class name	Source
CompileException	Predefined in the BIMserver
Query	Submitted by a User
QueryCompiler	Predefined in the BIMserver
QueryInterface	Predefined in the BIMserver, Must Be Implemented in Query by a User
VirtualClassLoader	Predefined in the BIMserver
VirtualFile	Predefined in the BIMserver
VirtualFileManager	Predefined in the BIMserver

4.2. 쿼리 생성기

BIM 서버 정보요청 프로그램을 생성기를 자바로 실행했다 (<https://github.com/triangelfish/QueryGeneratorforBIMserver0.2>). 그림 8은 정보요청 프로그램 생성기의 실행 내용을 보여준다.



<그림 8. 실행 개요>

정보요청 프로그램 생성기의 전단부는 이용사례 탐색기와 어휘 구문 분석기의 두 부분으로 구성된다. 이용사례 탐색기는 두 가지 용도로 쓰인다. 첫째, 입력 형식을 검색하고 특수한 어휘 구문 분석기로 구문 분석 작업을 전달한다. 둘째, 입력물의 이용사례를 검색한다. 이러한 이용사례 정보를 기반으로 후속 작업 흐름의 조정이 필요할 수도 있다. 모든 입력 파일들을 일괄적으로 처리하기 위해 정보요청 프로그램 생성기가 사용자들이 하나씩 입력 파일들을 지정하게 하는 대신에 모든 입력파일들이 포함된 저장소를 지정하도록 유도한다. 분야별 전문가들은 직접 MVD 문서들을 입력자료로 사용할 수 있다. 그들은 또한 사용자 맞춤형 입력 방식으로 검토대상의 속성들을 규정할 수 있다. 현재는 EXPRESS 언어, JavaScript Object Notation (JSON) 형식, 기타 분야별 특수 스키마 파일들에서 MVD 구문분석이 지원된다. 정보요청 프로그램 생성기는 이러한 MVD의 컨셉들을 IFC 객체와 대응시킴(mapping) 것이다. JSON 형식은 널리 이용되는 독립 플랫폼 파일 형식이다. 가벼운 파일 구조와 사람들이 이해하기 용이한 내용과 같은 특징들은 JSON을 입력에 적절한 형식이 되게 했다. 많은 분야별 전문가들이 이미 JSON을 데이터 교환을 수월하게 하기 위해 채택해 왔다. 따라서 JSON의 지원은 정보요청 프로그램 생성기가 AEC 위원회에서 기존의 자산들을 활용하도록 해준다. 특정 분야의 스키마 파일은 특정 이용사례에 적용된다. 정보요청 프로그램 생성을 위한 일반적 해법을 제공하는 EXPRESS 언어와 JSON 파일 형식과 달리 특수 분야의 입력 형식은 향후 개발되고 해당 상황에 따라 적용되어야 한다. 그러나 이러한 특수 분야 입력 형식들은 설계자들에게 대단히 효율적 도구가 될 것이다. 이 파일들은 특수 분야의 스키마에 대상 객체들의 목록을 만든다. mvdXML과 같은 기타 형식들도 널리 사용된다. 향후 더 많은 형식들을 지원할 것이지만 이 단계에서는 EXPRESS 언어나 JSON 형식과 몇 가지 특수 형식 내의 MVD만을 이용한다.

어휘 구문 분석기는 입력 내용으로부터 문자의 흐름을 읽어 들어서 토큰들을 출력하고 기호(symbol) 테이블을 구축한다. 토큰은 “일종의 어휘 단위를 표현하는 추상적 기호이다.” 패턴은 “어떤 토큰의 어휘들이 취할 수 있는” 형태를 설명한다. 패턴은 일반적인 표현으로 기술될 수 있다. 어휘는 “토큰에 대한 패턴이 일치하는” 문자들의 시퀀스이다. 표 2는 IFC 토큰들과 어휘들의 몇 가지 사례들을 보여준다.

본 연구진은 단순화되고 맞춤화된 자바 문맥 자유 문법 (Java Context-Free Grammar, CFG)인 “QueryJava”를 정의하고, 그림 9에서는 이를 Backus-Naur 형식 (BNF) (Backus, J. W. 1959)으로 보여준다. CFG는 일련의 생성 법칙(production rule)들로 구성된다. 각각의 법칙은 헤드(좌측에 목록화 된 기호들)와 헤드를 교체하기 위해 사용될 수 있는 기호들로 구성된다. 헤드는 최소 하나의 비 종결 기호(non-terminal symbol)를 담고 있다. 헤드는 종결 기호들과 비 종결 기호들에 의해 교체될 수 있다. 종결 기호들은 자바의 키워드인 “void”, “public”과 같은 CFG의 기본 요소들이다. 비 종결 기호들은 생성 법칙에 따라 비 종결 기호와 종결 기호 집단(Group)에 의해 교체될 수 있다. 종결 기호, 비 종결 기호, 생성 법칙들은 서로 다른 추상적 수준에서 언어의 문법을 표현하는 나무와 같은 데이터 구조를 구성한다. 종결 기호들은 나뭇잎들에 해당한다. 그림 9에서 종결 기호들은 굵은 글자로 강조된다. 그림 9에 QueryJava CFG의 핵심부분을 수록했다. QueryJava CFG를 규정할 때, 본 연구진은 BIM 서버 정보요청 프로그램의 요구에 맞도록 전체 자바 문법을 재조정하고 축소하였다. 재조정 측면에서 IR 생성기는 구문 분석기로부터 통과된 IFC 클래스 명칭들을 다른 자바 클래스 명칭등과는 별도로 독특한 어휘 요소로 다룰 수 있었다. 게다가 “MainClass”로 시작하는 일반적인 자바 프로그램과 달리 본 연구에서 생성된 코드는 하나의 자바 클래스 “Query”만을 갖는다는 알 수 있다. 이러한 “Query” 클래스는 고정된 종결 기호들과 4개의 비 종결 기호들(식별자, 구문, 필드 선언, 클래스 선언)로 구성된다. 축소(reduction) 라는 측면에서는 QueryJava CFG는 사용되지 않은 문법 요소들을 삭제한다.

최종 사용자들에게 IR 디자인과 실행은 완벽하게 감추어진다는 것을 주목하는 것이 중요하다. 다른 말로 사용자들은 본 연구 성과물을 사용하기 위해 IR을 알아야할 필요가 없다.

사용자들이 제공한 정보는 두 가지 특징들(관심 있는 객체 유형 정보와 적합한 객체들을 선별하기 위해 사용되는 필터 정보)을 갖고 있다. 자바 생성기는 IR에서 IFC 객체들의 유형 정보에 따라 변수 선언과 관련된 구문과 표현을 생성하는 반면에 필터 정보는 주로 조건문과 순환문으로 설정된다. 목표 코드 생성기는 오류 없이 호환되는 완벽한 자바 소스 코드나 기타 목표 언어들을 구성하기 위해 사전 저장된 루틴 코드와 생성된 구문들을 배치(interleave)한다. 이러한 프로그램의 배치 및 채워(fill)나가는 프로세스의 정확성은 QueryJava CFG에 의해 보장된다.

BIM 서버에 의해 정보요청 코드가 실행된 후에 결과물들이 JSON 파일로 익스포트된다. 결과물들은 사용자들에 의해 자유롭게 사용될 수 있다. 예를 들면, 직접 통계 분석이나 시뮬레이션을 수행할 수 있다. 하지만 여전히 데이터 처리 과정에서 데이터 누락이나 형식 불일치와 같은 문제점이 나타날 수 있다. BIM 서버 쿼리 생성기의 목적은 BIM 서버에서 건설 설계 어플리케이션들에 의해 사용될 수 있는 파일 형식으로 IFC 모델을 조회하는 것이므로 사용자들이 누락된 데이터를 처리하고 목표 소프트웨어 어플리케이션을 위한 결과물들을 구성하는 것을 보조하기 위해 두 가지 구성요소들(즉, 누락 데이터 탐색기와 결과물들과 리소스들의 통합기)을 추가로 실행하였다.

누락데이터 탐색기는 결과물들이 MVD에 포함된 모든 데이터 속성들을 관장하고 있는지 점검한다. 사용자들이 비어있는 필드에 데이터를 할당할 수 있게끔 누락데이터 탐색기는 인터페이스 패널을 이용하

여 사용자들에게 알려준다. 인터페이스 패널은 각 이용사례별로 개발되어야 한다. 사용자 할당 데이터는 리소스 파일로 저장된다.

Goal	::=	Query<EOF>
Query	::=	"class" Identifier "{" "@Override" "public" "void" "Query" "(" "IfcModelInterface" Identifier "," "PrintWriter" identifier ")" "{" statement "}" {field_declaration} {class_declaration} "
statement	::=	variable_declaration (expression ";") (statement_block) (if_statement) (for_statement) (try_statement) ("return" [expression] ";") ("throw" expression ";") (identifier "." statement) ("break" [identifier] ";") ("continue" [identifier] ";") (";")
identifier	::=	"a..z,\$,_" { "a..z,\$,_.0..9,unicode character over 00C0" }
class_declaration	::=	{ modifier } "class" identifier ["extends" class_name] ["implements" interface_name { "," interface_name }] "{" { field_declaration } "
field_declaration	::=	((method_declaration constructor_declaration variable_declaration)) static_initializer ";"
variable_declaration	::=	{modifier} type identifier { "[" "]" } ["=" variable_initializer] ";"
variable_initializer	::=	expression ("{" [variable_initializer { "," variable_initializer } [","]] "}")
expression	::=	numeric_expression testing_expression logical_expression string_expression bit_expression casting_expression creating_expression literal_expression "null" "super" "this" identifier ("(" expression ")") (expression ("(" [arglist] ")") ("[" expression "]") ("." expression) ("," expression) ("instanceof" (class_name interface_name)))
statement_block	::=	"{" { statement } "}"
if_statement	::=	"if" "(" expression ")" statement ["else" statement]
for_statement	::=	"for" "(" (variable_declaration (expression ";") ";") [expression] ";" [expression] ";") statement
try_statement	::=	"try" statement { "catch" "(" parameter ")" statement } ["finally" statement]
parameter	::=	type identifier { "[" "]" }
type	::=	type_specifier { "[" "]" }
type_specifier	::=	"boolean" "char" "short" "int" "float" "long" "double" class_name interface_name ifc_objects_name
method_declaration	::=	{ modifier } type identifier "(" [parameter_list] ")" { "[" "]" } (statement_block ";")
modifier	::=	"public" "private" "protected" "static" "final" "transient"
class_name	::=	identifier (package_name "." identifier)
interface_name	::=	identifier (package_name "." identifier)
ifc_objects_name	::=	"Ifc" Identifier ("org.bimserver.models.ifc2X3/4" "Ifc" Identifier)

<그림 9. QueryJava 자유문맥(Context-free) 문법(Grammar)>

결과물들과 리소스들의 통합기는 정보요청 결과물들과 리소스 파일들을 이용 사례 정보에 따라 하나의 파일로 통합한다. 이 파일은 EEB 설계 소프트웨어에 의해 사용될 준비가 된다.

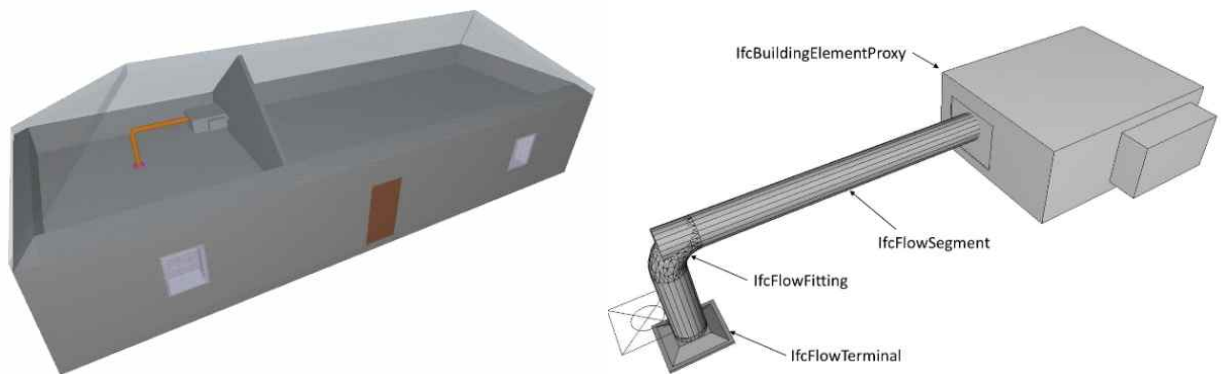
4.3. 정보요청 프로그램 생성 프로세스 요약

요약하면 정보요청 프로그램 생성기는 다음의 기능들을 수행한다. 첫째, 입력내용으로서 구조적 MVD 파일들을 사용한다. 둘째, 일련의 토큰들과 기호 테이블을 생성하기 위해 입력내용에 대한 어휘 분석을 수행한다. 그 다음으로 코드 라이브러리로부터 루틴 코드를 로드한다. 이러한 루틴 코드는 IR을 생성하기 위해 이전 단계의 토큰 및 기호 정보와 결합한다. 그 다음에 IR을 기반으로 구성 목표 코드 생성기가 원하는 언어들로 정보요청 프로그램 코드를 생성한다. 특정 이용사례들에 따라서 몇 가지 후속 또는 특수 응용 단계가 필요하다.

5. 사례연구

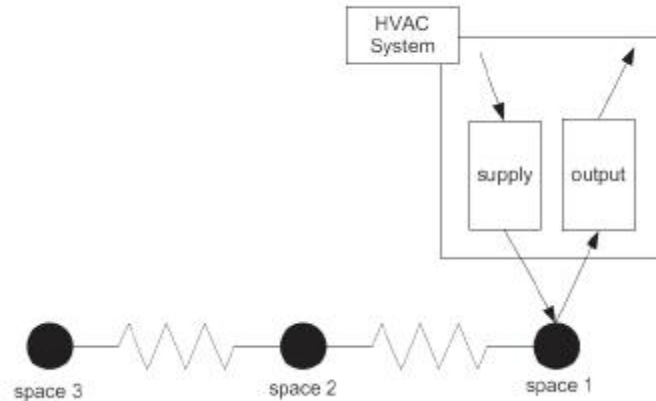
5.1. 실험 개요

사례연구는 정보요청 프로그램 생성기가 CONTAM (NIST 2013)을 위한 입력 파일을 자동으로 생성하여 공기 흐름 분석 작업을 위한 데이터 교환을 용이하게 하는지 시험하기 위해 수행되었다. 다중 영역 공기 흐름 분석은 에너지 효율 건물 설계를 위한 중요한 작업이다. CONTAM은 공기흐름 시뮬레이션을 위해 가장 널리 이용되는 도구들 중 하나다. 쿼리 생성기를 간단한 HVAC 설비를 갖춘 2층 규모의 간단한 건물 모델에 적용하여 시험했다. 자세한 사항들은 표3에 목록화 하였다. 이 모델은 Autodesk사의 Revit으로 개발하여 IFC 포맷으로 변환하여 도출하였다(그림 10 참조).

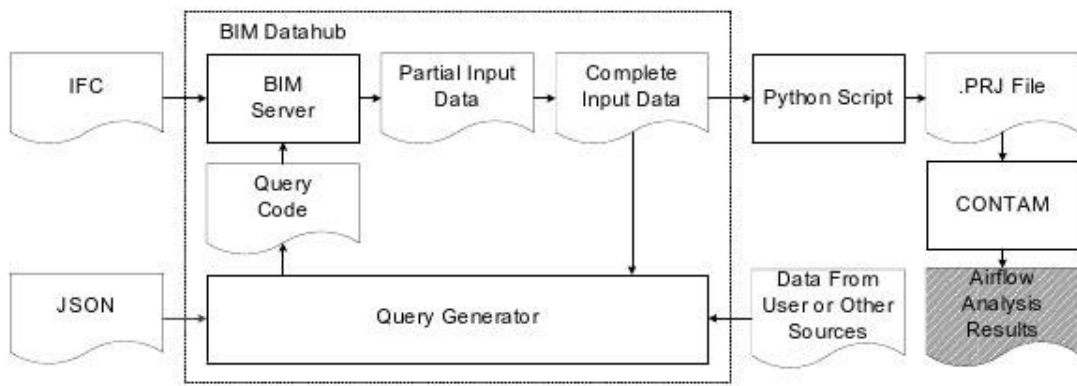


<그림 10. 건물 모델과 그 HVAC system의 상세들>

공기흐름 분석에 필요한 정보는 건물의 위상(topology)에 관한 것이 더 많다. 그림 11은 1층의 위상을 보여준다. 그림 11의 절점들은 공간이나 HVAC 설비를 나타내며, 썸기형상(edge)은 인접한 공간들 간의 공기 누출이나 공간들과 HVAC 설비 사이의 공기흐름을 나타낸다.



<그림 11. 연결 및 건물의 노드들>



<그림 12. BIM 서버 정보요청 코드 생성기를 이용한 연결 및 건물의 노드들>

5.2. 쿼리 생성기를 이용한 공기 흐름 분석의 작업 흐름 개요

그림 12는 쿼리 생성기를 이용한 전반적인 공기 흐름 분석 작업 흐름을 보여준다. 작업 흐름은 CONTAM에서 4단계(1단계: BIM 서버로부터 그린빌딩의 부분적 모델 조회, 2단계: 누락 데이터 할당 (선택사항), 3단계: 완전한 입력 파일 구축을 위한 부분적 모델 데이터와 누락 데이터의 결합, 4단계: 입력 파일 (.PRJ 파일)의 CONTAM 프로그램 로딩으로 구성된다.

쿼리 생성기는 사용자들이 전체적인 작업 흐름을 단계별로 마무리하도록 유도한다. 사용자는 쿼리 생성기가 자바 쿼리 코드를 생성하는 폴더와 입력 파일을 로드하는 폴더의 명칭을 지정해야 한다. 이 사례 분석에서 사용된 입력 파일 형식은 JSON이다. 그림 13은 CONTAM의 시뮬레이션 실행에 필요한 데이터를 조회하기 위한 입력 파일의 사례이다.

```
{
  "levels": [{"name": "", "elevation": , "height": }],
  "ahs": [{"name": "", "outdoorAirPercent": }],
  "zones": [{"name": "", "volume": , "level": "",
    "supplyAir": [{"system": "", "flow":
      {"value": , "units": ""}],
    "returnAir": [{"system": "", "flow":
      {"value": , "units": ""}}]},
  "walls": [{"name": "", "area": , "elevation": ,
    "zones": [""],
    "azimuth": }]}

```

<그림 13. 데이터 조회를 위한 입력 파일 사례>

생성기의 현재 버전이 BIM 서버 코드와 분리되어 있기 때문에 사용자는 생성된 코드를 복사해서 BIM 서버에 붙여넣기 해야 한다. 이 코드가 지정된 출력 폴더 내에 한 두 개의 파일들을 내보내기(export) 위해 BIM 서버를 구동시킨다. 첫 번째 파일은 BIM 서버로 부터의 조회 결과들을 포함하고 있다. 만약 누락된 데이터가 있으면 두 번째 파일은 누락된 데이터의 지정값(default value)를 포함시킬 것이다.

본 사례의 그린빌딩에 대한 IFC 모델은 필요한 몇 가지 정보 유형들을 담지 못한다. BIM 서버 정보요청 프로그램 코드 생성기는 이러한 사실을 탐지해서 사용자가 그러한 누락된 속성들을 할당하도록 해주는 패널을 팝업창에 띄운다.

그 다음 단계는 정보요청 코드에 의한 부분적 입력 데이터와 사용자 입력데이터를 최종 입력 파일에 통합시키는 것이다. 그림 15는 그린빌딩의 최종 정보요청 프로그램 결과의 일부를 보여준다.

<그림 14. 누락된 데이터를 수집하기 위한 사용자 입력 패널>

```
{
  "walls": [
    {
      "area": 22.360693,
      "azimuth": 0,
      "elevation": 1.5,
      "name": "Basic Wall: Interior - 4 7/8 Partition (1-hr): 158353",
      "zones": ["1", "2"]
    },
    {
      "area": 20.10463,
      "azimuth": 90,
      "elevation": 1.5,
      "name": "Basic Wall: Exterior - Brick on CMU: 157673",
      "zones": ["3"]
    },
    {
      "area": 60.96375,
      "azimuth": 180,
      "elevation": 1.5,
      "name": "Basic Wall: Exterior - Brick on CMU: 157571",
      "zones": ["1", "2"]
    },
    {
      "area": 22.360693,
      "azimuth": 0,
      "elevation": 1.5,
      "name": "Basic Wall: Interior - 4 7/8 Partition (1-hr): 158433",
      "zones": ["2", "3"]
    },
    {
      "area": 54.890507,
      "azimuth": 270,
      "elevation": 1.5,
      "name": "Basic Wall: Exterior - Brick on CMU: 157723",
      "zones": ["1", "2"]
    },
    {
      "area": 20.10463,
      "azimuth": 0,
      "elevation": 1.5,
      "name": "Basic Wall: Exterior - Brick on CMU: 157953",
      "zones": ["1"]
    }
  ],
  "levels": [
    {
      "elevation": 3,
      "height": 0,
      "name": "Level 2"
    },
    {
      "elevation": 0,
      "height": 3,
      "name": "Level 1"
    }
  ],
  "description": null,
  "name": "Project Number",
  "ahs": [
    {
      "name": "ahs",
      "outdoorAirPercent": 35
    }
  ],
  "zones": [
    {
      "level": "Level 1",
      "name": "3",
      "returnAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 280,
            "system": "ahs"
          }
        }
      ],
      "supplyAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 180,
            "system": "ahs"
          }
        }
      ],
      "volume": 78.13969
    },
    {
      "level": "Level 1",
      "name": "2",
      "returnAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 380,
            "system": "ahs"
          }
        }
      ],
      "supplyAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 380,
            "system": "ahs"
          }
        }
      ],
      "volume": 80.64787
    },
    {
      "level": "Level 1",
      "name": "1",
      "returnAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 380,
            "system": "ahs"
          }
        }
      ],
      "supplyAir": [
        {
          "flow": {
            "units": "scfm",
            "value": 380,
            "system": "ahs"
          }
        }
      ],
      "volume": 78.13969
    }
  ]
}
```

<그림 15. JSON 형식의 데이터>

6. 한계

쿼리 생성기의 시제품(prototype)은 현재 몇 가지 한계가 있다. 첫째 제안된 방식은 몇 가지 가능한 입력 포맷들을 제안했다. 그들 중 몇 가지는 아직 이 시제품에서 실행되지 않는다. 둘째, 이 제품은 추가 필터들을 지원할 수 없다. 이 두 가지 항목은 추가적인 공학적 노력을 필요로 한다.

BIM 서버 정보요청 프로그램 코드 생성의 자동화는 단편적인 소프트웨어 어플리케이션들과 EEB 설계 분야의 다양한 참여자들을 데이터 수준에서 통합시키는데 기여할 수 있다. 이 연구는 데이터 수준의 통합을 개선하여 수평적으로 확장될 수 있고 적용 소프트웨어의 통합을 달성하여 수직적으로 확장될 수 있다. 데이터 수준에서 제안되는 향후 연구는 다음과 같다. 첫째 더 많은 설계 시나리오들(예: 주광 분석 및 에너지 소비분석)이 개발되고 탐색될 수 있다. 둘째 gbXML과 STEP 파일과 같은 더 많은 데이터 포맷들을 지원할 수 있다. 셋째 코드 생성 프로세스를 간소화하고 통합하기 위해 IR이 더 세밀하게 조정될 수 있다. 실무적용 수준에서는 정보요청 코드 자동 생성 이외에도 자동적 정보요청 코드 가져오기와 내 보내기 기능이 건물 설계 과정을 보다 용이하게 할 수 있다. 현재의 BIM 데이터 서버 기능들과 다른 어플리케이션들의 웹 상에서 실행을 구현하는 것은 어플리케이션 간 프로그램 함수 호출 문제를 해결해

줄 수 있다.

7. 결론 및 향후 연구

개방형 오픈 데이터 표준들과 BIM 데이터 허브 실행물 들을 조사하여 현재의 부분적 모델 정보요청 메카니즘들로 인해 건축 설계자들과 분석가들이 BIM 모든 장점들을 취하지 못한다는 사실을 알아냈다. 이러한 문제를 다루기 위해 다음과 같은 성과들을 얻어냈다. 첫째 정보요청 언어 상세물 들과 BIM 데이터 허브 내부 데이터 구조 실행물을 사용자들에게 명확히 인식하도록 하는 정보요청 프로그램 생성방안을 제안하였다. 둘째 코드 생성 프로세스를 구동하기 위해 맞춤화되고 간소화된 IR을 정의했다. 셋째 정보요청 생성기의 시제품을 구축하여 실제 다중 영역 공기흐름 분석에 이 방법을 적용했다. 이러한 성과들을 EEB 설계 작업 흐름에서 간접비를 절감시킬 수 있을 것이다. 향후에는 보다 정량적인 분석을 수행할 수 있도록 사용자 중심의 연구를 수행하고자 한다.