소프트웨어 설계 툴의 코드자동생성능력 비교 연구

조수란O 강성원
한국정보통신대학교
{ddangly-, kangsw}@icu.ac.kr

A Comparison Study of Code Generation Capability of Software Design Tools

Sooran Jo0 Sungwon Kang
Information and Communications University

요 약

소프트웨어 설계 툴은 소프트웨어 개발을 위하여 실무에 도입되어 많이 이용되고 있다. 그러나 대부분의 소프트웨어 설계 툴이 코드를 자동으로 생성할 수 있는 기능을 제공하고 있어서 이를 잘 활용하면 개발시간을 단축하고 및 개발생산성을 향상 크게 향상 시킬 수 있음에도 불구하고, 많은 사용자들은 설계 툴을 단순한 모델링 툴로서만 이용하고 있다. 본 연구에서는 사용자들이 소프트웨어 설계 툴의 코드생성능력을 잘 활용할 수 있도록, 몇 개의 선정된 설계 툴의 코드자동생성능력의 비교를 통해 설계 툴의 코드생성능력을 비교 및 분석하였다. 자동 생성된 코드는 Java 소스코드이며, 언어지원능력, 난이도, 생성된 소스 코드의 레벨을 비교기준으로 평가를 수행하였다.

1 서론

CASE(Computer Aided Software Engineering)는 개발 방법론과 그것들을 지원하는 도구, 소프트웨어 재사용 기법, Life Cycle에 따른 유지보수 등을 총괄하며, 이러한 개발과정을 자동화하기 위하여 사용하는 소프트웨어에 도구를 의미[1]한다.

CASE 툴을 잘 사용하게 되면 사용자는 개발 과정의 일부 혹은 전체를 자동화하게 되므로 그를 통해 개발기간이 단축되게 되고, 품질을 향상할 수 있게 되며 유지보수에 필요한 노력과 비용을 절감할 수 있게 된다. CASE 툴 중에서도 특히 설계 툴은 개발 Life Cycle에서 설계의 중요성 뿐 아니라 구현 code의 상당부분을 툴에서 생성할 수 있으므로, 개발시간의 단축 및 개발생산성을 위하여 그 중요성이 매우 높다.

발표자료[2]에 따르면, 국내의 실무자들은 정보시스템을 구축하는 과정에서 81.4%로 상당히 많은 수가 설계 툴을 사용해 본 경험이 있으며 사용자의 2명 중 1명꼴인, 52.8%의 사람들은 설계 툴을 자주 사용하고 있다고 한다. 이들 사용자들은 설계 툴을 Data Modeling(52.7%)과 Processing Modeling (24.9%)로 활용하고 있다고 한다. 즉, 우리나라 실무자의 40% 이상은 설계 툴을 자주 사용하고 있으나 그 주된 용도는 소스코드 생성용이 아닌 설계의 모델링이다. 즉, 사용자들은 툴이 제공하는 기능을 충분히 활용하지 못하고 있다.

본 연구에서는 사용자들이 소프트웨어 설계 툴의 코드생성능력을 잘 활용할 수 있도록, 몇 개의 선정된 설계 툴의 코드자동생성능력의 비교를 통해 설계 툴의 코드생성능력을 비교 및 분석하였다. 자동 생성된 코드는 Java 소스코드이며, 언어지원능력, 난이도, 생성된 소스 코드의 레벨을 비교기준으로 평가를 수행하였다.

본 논문의 구성은 다음과 같다. 제 2절에서는 소프트웨어 설계 툴의 선정 방법 및 소개와 대상의 비교방법을 제시하며. 제 3절에서는 제2절을 바탕으로 실질적인 소프트웨어 설계 툴의 코드자동생성능력을 비교하며, 제 4절은 본 논문의 결론을 맺는다.

2 비교 방법

2.1 대상선정 방법

시중에서 많이 사용하는 설계 툴로는 RATIONAL-ROSE, ER-WIN, TOGETHER, RHAPSODY, POWER-DESIGNER, BP-WIN, A0-WIN 등이 있으며, 이 외에도 수십여 종이 더 사용되고 있다.

대부분의 툴이 유료로 제공 되고 있기 때문에 본 연구를 위한 대상 선정에 있어서 제약을 받게 되었다. 따라서 유료 툴로는 RATIONAL-ROSE와 RHAPSODY를 대상으로 선정하였다. 또한, 오픈소스 UML 소프트웨어 중 starUML을 대상으로 선정하였다.

2.2 선정한 설계 툴의 소개

1) Rhapsody

Rhapsody는 C, C++, Java, Ada 등의 언어를 지원하며 UML의 그래픽 프로그래밍과 통합되어 실제 코드 생성을 통한 테스트까지 제공하여 디자인은 곧 해석이라는 완벽한 개발환경을 제공한다. 또한 최고의 기술을 사용하여 팀 간의 협력과 재사용 가능한 디자인의 개발, 소프트웨어의 질을 향상시키는데 많은 도움을 준다.[3] Rhapsody는 개발 프로세스의 획기적인 발전과 개발 기간의 단축으로 전체 개발비용을 줄일 수 있으며 궁극적으로 시장 적응력을 향상시켜줄 하나의 Tool로 자리매김하였다.[4]

2) Rational-Rose

다음으로 Rational-Rose는 Java, C 및 C++를 위한 완전 자동화된 설계 및 코딩 변환을 제공하며 가장 강력한 모델 중심 개발 솔루션을 제공하여 준다고 한다. 또한, 런타임 모델 실행, 완벽하게 실행 가능한 코드 작성, 비주얼 디버깅도 하나의 장점이며 드라이버, 스텁, 테스트 장비, 실제 테스트 스크립트 등을 자동으로 생성한다. 이벤트 기반, 동시 및 분산 애플리케이션에 맞춰 최적화 되어 있으며 대기 시간, 처리량, 신뢰성 등에 대한 엄격한 요구사항을 준수할 수 있는 고급 모델링 구축[5]한다. 기술적으로 매우 까다로운 애플리케이션에 맞춰 설계되었다.

3) starUML

StarUML™은 UML(Unified Modeling Language)을 지원하는 소프트웨어 모델링 플랫폼이다. UML 버전 1.4에 기반을 두고 있으며, UML 버전 2.0의 표기법을 적극적으로 지원하고 있다. 총 11가지의 다양한 종류의 다이어그램을 제공할 뿐만 아니라 UML 프로파일 개념과 템플릿 기반의 문서 및 코드 생성을 지원하여 MDA(Model Driven Architecture) 접근방법을 적극적으로 지원한다. 또한 고객의 환경에 대한 맞춤 능력이 우수하고 기능에 대한 확장성이 매우 뛰어난 것이 장점이다. 소프트웨어 모델링 도구 중의 하나인 StarUML™을 사용하면 소프트웨어 프로젝트의 생산성(Productivity), 품질(Quality)이 획기적으로 높아질 것이다.

2.3. 대상비교 방법

본 논문에서는 크게 아래의 3가지의 기준에 의해서 대상을 비교하며 각 항목에 대한 평가 결과는 상, 중 또는 하가 된다.

① **툴의 언어지원능력** : 얼마나 많은 종류의 언어를 지원하여 code를 생성할 수 있는지를 비교하고 Round-trip Engineering 관점에서 순공학과 역공학[t]이 얼마나 매끄러운지를 평가 할 것이다. 다양한 언어를 지원하고 순공학과 역공학기 매끄러울수록 언어지원능력이 높다고 할 수 있다.

- 언어지원능력 1 (지원언어의 수) : 상 → 자동코드생성을 할 수 있는 언어의 수가 5개 이상인 경우, 중 → 자동코드생성을 할 수 있는 언어의 수가 3개 이상 5개 이하인 경우, 하 → 자동코드생성을 할 수 있는 언어의 수가 3개 미만인 경우
- 언어지원능력 2 (Round-trip) : 상 → 순공학과 역공학이 매끄럽게 연계되어있어 Round-trip이 매우 자연스러운 경우, 중 → 순공학과 역공학이 매끄럽지는 않으나 연계되어있어 Round-trip을 지원하는 경우, 하 → 순공학과 역공학을 지원하기는 하나Round-trip을 지원하지 않는 경우 혹은 아무 것도 지원하지 않는 경우

② **Tool의 난이도** : 이 항목은 매우 개인적인 의견이 추가될 수 있는 항목이다. 소프트웨어 설계 툴의 사용 경험이 거의 없는 필자가 Tool을 사용해본 후 사용법과 익히는 시간 등의 난이도를 기준으로 평가하는 것이다. 난이도가 낮을수록 좋다고 할 수 있지만 너무 어렵지만 않다면 괜찮다고 생각한다. (상 : 처음 접하는 사용자가 일주일 이내에 마스터하기 어려울 뿐만 아니라 관련된 reference 역시 찾기 어려울 경우, 중 : 처음 접하는 사용자가 툴의 기능을 일주일 이내에 에 마스터 하기는 어려우나 tool에 관련된 reference를 쉽게 찾을 수 있는 경우, 하 : 처음 접하는 사용자가 일주일 이내에 툴의 기능의 70%이상을 마스터 할 수 있으며 tool에 관련된 reference를 쉽게 찾을 수 있을 경우)

③ **생성된 소스 코드의 레벨** : 자동 생성된 코드가 얼마나 사용자가 바라는 결과와 일치하고, Code Redundancy가 적으며, 이해하기 쉽게 작성되어 있는지, Time Complexity가 효과적인지, 또한 사용자가 얼마나 자의적으로 참여할 수 있는지를 기준으로 자동 생성된 소스 코드의 레벨을 평가할 것이다.

- 결과의 일치성 : 상 → 사용자가 원하는 결과를 출력하였을 경우, 중 → 비슷한 결과를 출력하기는 하나 사용자가 원하는 결과가 아닐 경우, 하 → 결과가 출력되지 않고 컴파일 에러가 나왔을 경우
- Code Redundancy : 상 → 손으로 직접 짠 소스코드보다 20배 이상일 경우, 중 → 손으로 직접 짠 소스코드보다 10배 이상일 경우, 하 → 손으로 직접 짠 소스코드보다 10배 미만일 경우
- 빠른 실행 속도 : 상 → 3초 이내에 실행 될 경우, 중 → 5초 이내에 실행될 경우, 하 → 5초 이상 시간이 걸릴 경우
- 소스코드의 이해도 : 상 → 사용자의 입력이 거의 없이도 자동 생성된 부분의 코드가 함수명, 변수명 등이 하는 역할과 관련하여 잘 작명되어 있으며 구조가 효율적일 경우, 중 → 자동 생성된 부분의 코드가 비록 잘 작명되어있고 구조가 효율적이기는 하나 사용자의 입력을 많이 필요로 하는 경우, 하 → 사용자의 입력을 많이 필요로 하며 그럼에도 불구하고 작명 또는 구조에 손색이 있을 경우
- 사용자의 참여도(복잡한 경우에만 시행) : 상 → access level, 변수타입, 어레이형 변수, final 변수, static 변수, 초기 값 지정이 가능한 경우, 중 → access level, 변수타입, final변수, static 변수, 초기 값 지정이 가능한 경우, 하 → access level, 변수타입, 초기 값 지정이 가능한 경우

이 세 가지 항목의 점수를 비교 한 뒤 전체 합산을 내어 어느 부분에서는 어느 툴이 유용하며 어떠한 사용자에게는 어떤 툴이 효율적이라는 것을 추천해 줄 것이다.

3 소프트웨어 설계 툴의 코드자동생성능력 비교

비교 기준 설정에 있어서 UML과 Java 중심의 구조적 소프트웨어 공학과 객체-지향소프트웨어공학을 참고[7]하였다.

3.1. 선정한 설계 툴의 언어지원능력

첫째로, 현재 널리 사용되고 있는 언어 중 몇 개의 언어를 지원하여 code를 생성할 수 있는지를 기준으로 비교 할 것이다.

다음으로, 모델에 포함된 정보로부터 소스 코드를 생성하는 코드생성기(순공학)로서 역할과, 생성된 코드로부터 시스템 설계를 표현하는 모델을 재구성 하거나 구성하는 역할(역공학)을 충실히 하는지를 비교하고 순공학과 역공학 사이의 연결이 얼마나 매끄러운지를 비교한다.

1) Rhapsody

첫째로, Rhapsody는 네 가지 프로그래밍 언어를 선택할 수 있다. C, C++, Java, 그리고 Ada가 바로 그것이다. 하지만, Rhapsody in C, Rhapsody in Java처럼 하나의 프로그래밍 언어만을 지원하는 제품으로 서로 다른 여러 언어를 혼용해야 하는 대규모 시스템의 경우에는 적합하지 않을 수 있다.

다음으로, Rhapsody는 Round-trip Engineering 관점에서 우수한 기능을 가지고 있다. 다이나믹 모델/코드결합(Dynamic Model/Code Associativity)을 하나의 주요기술로 내세울 만큼 UML 모델과 코드가 동적으로 결합되어 있어서 수정이 매끄럽다. 또한, 수정한 부분이 하이라이트 표시되어 복원도 쉽다.

따라서 랩소디는 비록 혼용 언어 지원은 하지 않고 있지만 현재 많이 사용되고 있는 C, C++, Java, Ada를 지원하며, 순공학과 역공학의 관계가 밀접하여 코드와 모델이 매끄럽게 연결되어 있다.

언어지원능력 1 (지원언어의 수) : 중

언어지원능력 2 (Round-trip) : 상

2) Rational Rose

첫째로, Rational Rose는 여러 가지 edition이 있다. 모델러 에디션의 경우는 프로그래밍 언어를 지원하지 않는 제품이며, 프로페셔널 에디션은 하나의 프로그래밍 언어를 지원하는 제품으로 C++, Java, Visual Basic, Smalltalk, Ada, Power Build, Delphai 버전이 있다. 또한, 엔터프라이즈 에디션은 C++, Java, Visual Basic, Power Build, Oracle 8등을 동시에 지원하는 제품으로 하나의 모델 설계에서 서로 다른 여러 언어를 동시에 혼용해야 하는 대규모 시스템에서 사용하기 적합하다.

다음으로, Rational Rose는 반복적인 최초의 설계 모델과 구현으로부터 역공학 된 새로운 설계모델 사이의 차이점을 나타내주는 디퍼런싱 툴로서의 역할도 하고 있다. 하지만, 로즈에 의해 생성된 코드가 아닌 경우에는 역공학만 가능하다. 또한, 경우에 따라서 내용이 보존되지 않기도 하는데 그러한 경우로는 역공학에 의해 모델을 생성하고 난 후 다시 소스 코드로 갱신 할 경우에 소스 코드에 있는 멤버 함수 내의 코딩 내용이 보존되지 않는 경우가 있다. 즉, 역공학과 순공학 지원에 있어서코딩 내용이 보존되지 않는 등 매끄럽지 않은 경우가 많다.

언어지원능력 1 (지원언어의 수) : 상

언어지원능력 2 (Round-trip) : 중

3) starUML

starUML은 여러 언어로 확장 가능하지만 소스 코드 생성의 경우, 템플릿 기반으로 각종 문서와 코드를 생성할 수 있게 해주는 Generator 모듈을 제공해야만 가능하다. starUML은 Java 언어의 프로파일과 J2SE/J2EE 프레임워크, 코드생성 및 역공학을 지원하는 C++ 언어 모듈을 제공하며, C++언어의 프로파일과 MFC 프레임워크, 코드생성 및 역공학을 지원하는 C++ 언어 모듈을 제공한다. 즉, Java와 C++의 자동소스코드 생성을 지원한다. 다음으로, starUML은 역공학을 지원하기는 하지만, Round-trip Engineering을 지원하지 않는다.

언어지원능력 1 (지원언어의 수) : 하

언어지원능력 2 (Round-trip) : 하

3.2. Tool의 난이도

1) Rhapsody

Rhapsody의 경우 유료 툴인 만큼 다른 객체지향 툴에 비하여 다양하고 많은 기능을 제공한다. 따라서 단순히 일반적인 객체지향 툴의 기능뿐만 아닌, 실제 비교해보고자 하는 코드 생성, 문서 생성 등의 기능을 이용하기 위해서는 매우 많은 시간을 요구한다. 또한, 아직 값도 비싸고 널리 사용되는 툴이 아니기 때문에 사용법에 관한 reference나 예제 등을 찾기도 어려워 습득하는 시간이 오래 걸린다.

따라서 적은 reference와 복잡한 기능으로 인해서 Tool의 난이도는 높은 편이라고 할 수 있다.

Tool의 난이도 : 상

2) Rational Rose

Rational Rose의 경우 Rhapsody와 마찬가지로 유료 툴인 만큼 다른 객체지향 툴에 비하여 다양한 기능을 제공한다. 역시 Rhapsody와 마찬가지로 많은 시간을 요구한다. 하지만, Rational Rose의 경우 Rhapsody에 비하여 현재 많은 마니아 층을 확보하고 있기 때문에 사용법에 관한 reference를 찾기 쉽다. 또한 출시된 지 오래되었기 때문에 Rational Rose에 대한 예시등도 많이 제공되기 때문에 하고자 하는 업무와 관련된 예시를 찾을 가능성도 높다.

따라서 Tool 자체의 난이도는 높은 편이지만, 관련 자료가 많기 때문에 어느 정도 쉽게 배울 수 있다.

Tool의 난이도 : 중

3) starUML

starUML의 경우 오픈 소스 툴인 만큼 기능이 제한적이다. 기능이 제한적이라 모든 기능을 마스터 하는 시간도 적게 걸릴뿐더러 오프소스 프로그램이기 때문에 마니아층을 단시간에 많이 확보하여 관련 reference를 찾기 쉽다.

따라서 Tool 자체의 난이도도 낮은 편이며 관련 자료가 많기 때문에 쉽게 Tool을 익힐 수 있다.

Tool의 난이도 : 하

3.3. 생성된 소스 코드의 레벨

3.3.1. 코드가 단순한 경우

비교 기준은 위에서 밝혔던 사용자가 바라는 것과 일치하는 결과를 출력하는지, Code Redundancy 정도는 얼마나 되는지, 실행 time이 얼마나 걸리는지(Time Complexity)를 기준으로 비교한다. 하지만, 이 경우에는 단순한 경우인 만큼 사용자가 바라는 결과와 일치하지 않는 경우는 없고 time도 millisecond 정도로 매우 적게 차이가 나기 때문에 생략하고 Code Redundancy만을 비교하도록 하겠다. 추가적으로 소스코드의 이해도도 비교해보도록 하겠다.

"Hello JSR"을 연속으로 출력하는 Java code를 생성하는 경우를 통하여 비교를 해보겠다. 일반적으로 JAVA code로 이와 같은 프로그램을 작성 하는 것은 10줄 정도의 길이의 code가 필요하다.

1) Rhapsody

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 매우 크다. 위에서 언급 했듯이 실제로 코딩하면 10줄이면 되는 코딩이지만, 코드를 생성할 경우 220여 줄이 생성 되었다.

③ 3초 이내에 결과를 출력하였다.

④ Rhapsody가 생성한 코드는 효율적인 구조로 되어있다. 일반적으로 자동생성 된 코드를 'Black-Box' 코드라고 한다. 중요도가 높아서가 아니고 그만큼 속을 알 수 없기 때문에 붙여진 이름이다. Rhapsody의 경우 'Black-Box' 코드라고 보기에는 무리가 있다. 여느 프로그래머가 만든 코드처럼 쉽게 읽힐 수 있어서 손으로 쓴 코드라 생각하고 일반적인 코드 테스트를 해 보아도 손색없을 정도이며, 바로 deploy 될 수 있는 코드를 생성한다. 주석, 함수명, 변수명 등이 모두 손색없이 잘 작명되어 있다.

결과의 일치성 : 상

Code Redundancy : 상

빠른 실행 속도 (Time Complexity) : 상

소스 코드의 이해도 : 상

2) Rational Rose

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 매우 크다. 위에서 언급 했듯이 실제로 코딩하면 10줄이면 되는 코딩이지만, 코드를 생성할 경우 280여 줄이 생성 되었다.

③ 3초 이내에 결과를 출력하였다.

④ Rational Rose가 생성한 코드 역시 제법 효율적인 구조로 되어있다. 주석, 함수명, 변수명 등이 Rhapsody와 마찬가지로 잘 작명되어 있으며 구조 및 메소드가 효율적으로 구성되어 있다.

결과의 일치성 : 상

Code Redundancy : 상

빠른 실행 속도 (Time Complexity) : 상

소스 코드의 이해도 : 상

3) starUML

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 매우 크다. 위에서 언급 했듯이 실제로 코딩하면 10줄이면 되는 코딩이지만, 코드를 생성할 경우 180여 줄이 생성 되었다. starUML 자체로는 매우 큰 code redundancy이지만 Rhapsody와 Rational Rose에 비해서는 적은 편이다.

③ 3초 이내에 결과를 출력하였다.

④ starUML의 경우에는 상당히 많은 부분을 user가 입력해야만 한다. 사용자가 많은 부분을 입력하기 때문에 이해도가 높다고 생각할 수 있으나, 그것은 사용자가 입력한 코드이지 발생한 코드가 아니다. 따라서 이해도가 높게 평가되기는 하나 이 점이 이해도 높은 코드를 생성했다고 보기에는 문제가 있다.

결과의 일치성 : 상

Code Redundancy : 중

빠른 실행 속도 (Time Complexity) : 상

소스 코드의 이해도 : 중

3.3.2. 코드가 복잡한 경우

이 경우에는 UML을 이용하여 10개의 class를 설계한 후 자바코드로 변환 해 볼 것이다.[8] Class를 여러 개 생성하는 이유는 time을 비교하기 위해서 이다. 실제 코딩 시 400여줄 정도 되는 분량이다.

1) Rhapsody

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 보통이다. 위에서 언급 했듯이 실제로 코딩하면 400여 줄이다. 이 경우 생성된 소스코드는 500여 줄로 code redundancy가 크지 않다는 것을 알 수 있다.

③ 3초 이내에 결과를 출력하였다.

④ Simple case에서와 마찬가지로 Rhapsody가 생성한 코드는 제법 효율적인 구조로 되어있다. 여느 프로그래머가 만

든 코드처럼 쉽게 읽힐 수 있어서 손으로 쓴 코드라 생각하고 일반적인 코드 테스트를 해 보아도 손색없을 정도이며, 바로 deploy 될 수 있는 코드를 생성한다. package 지정이 가능하며 클래스다이어그램에 표현한 것만이 아니라, 자동으로 여러 라이브러리를 import하며, 주석, 함수명, 변수명 등이 모두 손색없이 잘 작명되어 있다.

⑤ access level, 변수타입, 어레이형 변수, final 변수, static 변수, 초기 값 지정이 가능하였다.

결과의 일치성 : 상
Code Redundancy : 하
빠른 실행 속도 (Time Complexity) : 상
소스 코드의 이해도 : 상
사용자의 참여도 : 상

2) Rational-Rose

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 보통이다. 위에서 언급 했듯이 실제로 코딩하면 400여 줄이다. 이 경우 생성된 소스코드는 550여 줄로 code redundancy가 크지 않다는 것을 알 수 있다.

③ 3초 이내에 결과를 출력하였다.

④ Rational Rose가 생성한 코드 역시 제법 효율적인 구조로 되어있다. 주석, 함수명, 변수명 등이 Rhapsody와 마찬가지로 잘 작명되어 있으며 구조 및 메소드가 효율적으로 구성되어 있으나 나 여러 라이브러리를 자동으로 import하지는 않으며, 클래스다이어그램에 표현한 것만 정확히 코드로 변환한다.

⑤ access level, 변수타입, final변수, static 변수, 초기 값 지정이 가능하였다.

결과의 일치성 : 상
Code Redundancy : 하
빠른 실행 속도 (Time Complexity) : 상
소스 코드의 이해도 : 상
사용자의 참여도 : 중

3) starUML

① 바라는 결과를 그대로 출력했다.

② Code Redundancy가 크다. 위에서 언급 했듯이 실제로 코딩하면 400여 줄이다. 이 경우 생성된 소스코드는 700여 줄로 code redundancy가 크지 않다는 것을 알 수 있다.

③ 5초 이내에 결과를 출력하였다. Rhapsody나 Rational Rose에 비하여 많은 시간이 걸렸다.

④ StarUML의 경우에는 상당히 많은 부분을 user가 입력해야만 한다. 사용자가 많은 부분을 입력하기 때문에 이해도가 높다고 생각할 수 있으나, 그것은 사용자가 입력한 코드이지 발생한 코드가 아니다. 따라서 이해도가 높게 평가되기는 하나 이 점이 이해도 높은 코드를 생성했다고 보기에는 문제가 있다. 또한, 여러 라이브러리를 자동으로 import하지는 않으며, 클래스다이어그램에 표현한 것만 정확히 코드로 변환한다.

⑤ access level, 변수타입, 초기 값 지정이 가능하였다.

결과의 일치성 : 상
Code Redundancy : 하
빠른 실행 속도 (Time Complexity) : 중
소스 코드의 이해도 : 중
사용자의 참여도 : 하

표 1. 코드 생성능력 비교표

비교항목 \ 툴목록		Rhapsody	Rational Rose	starUML
언어지원능력	지원언어의 수	중	상	하
	Round-trip	상	중	하
Tool의 난이도		상	중	하
코드의 레벨 (단순한 경우)	결과의 일치성	상	상	상
	Code Redundancy	상	상	중
	빠른 실행속도	상	상	상
	코드의 이해도	상	상	중
코드의 레벨 (복잡한 경우)	결과의 일치성	상	상	상
	Code Redundancy	하	하	하
	빠른 실행속도	상	상	중
	코드의 이해도	상	상	중
	사용자의 참여도	상	중	하

3.4. 생성 능력 비교 분석

이와 같이 3개의 툴을 비교한 결과가 표 1에 요약되어 있다. 전체적인 면에서 보면 Rhapsody가 가장 무난하게 사용할 수 있다. 하지만 사용자가 모델링한 구조가 올바르다면 올바른 소스코드가 생성되기 때문에 소스코드 생성 능력을 비교하는 것은 의미가 없다. 툴에 따라서 사용자에게 많은 기능을 제공해 주기도 하고 적은 기능을 제공해주기도 하지만, 많은 기능은 때로는 세부적인 설정을 할 수 있어서 유익하기도 하지만 사용자가 복잡한 기능을 설정해야 하므로 번거로울 수도 있다. 툴의 종류에 따라서 소스코드의 생성능력 자체에는 큰 차이가 없기 때문에, 사용자는 자신의 특성에 따라서 많은 기능을 제공해 주는 툴을 사용할 것인지 적은 기능을 제공해 주는 툴을 사용할 것인지를 결정하면 된다. 툴들 간의 소스코드 생성 능력 자체의 차이를 염려하는 것보다는 오히려, 모델링의 올바른 구조를 잡는 것이 더욱 중요한 일일 것이다.

4 결론

본 논문의 연구결과에 따르면, 소프트웨어 설계 툴의 코드 생성 능력은 언어지원능력, Tool의 난이도, 이해도의 경우는 약간의 차이를 보였고 생성된 코드의 레벨에 있어서는 사용자의 참여도를 제외하고는 큰 차이가 없었다. 언어지원능력 및 Tool의 난이도의 경우 개발자들의 편의 및 목적에 의해 제공된 것이며 기본적인 비교항목이기 때문에 생성되는 소스코드의 질과는 큰 연관이 없다. 또한, 사용자의 참여도는 툴이 제공해주는 기능과 연관된 것이기 때문에 소스코드의 질 보다는 툴이 제공하는 기능의 옵션에 가까운 것이다. 즉, 툴에 따른 특성에 의하여 기본적인 항목에 있어서 차이를 보이기는 하였지만 질적 비교인 소스코드의 이해도에 있어서는 Rhapsody와 Rational Rose가 starUML에 비하여 코드의 이해도가 높은 것을 제외하면 생성된 코드의 질의 좋고 나쁨의 차이는 거의 없었다.

그러므로 설계 툴의 선정에 있어서 생성된 코드의 효율성은 중요한 요소라고 보기는 힘들며, 오히려 모델링 측면의 효율성이 중요한 요소라고 보아야 한다. 또한 툴의 종류에 상관없이 코드생성능력을 활용하느냐의 여부가 개발시간단축 및 개발생산성을 위하여 결정이 필요한 사항이라고 보아야 한다.

5 References

[1] 두산편집부, 두산세계대백과-ENCYBER DELUXE, 두산, 2002
[2] http://cs.yonsei.ac.kr/%7Esanghyun/class/graduate/se/case.ppt
[3] http://www.i-logix.co.kr/rhap/rhap_main.html
[4] http://www.onjava.com/pub/a/onjava/2001/01/25/uml_rhapsody.html
[5] Michael Boggs, Wendy Boggs, UML과 Rational Rose 비주얼 모델링, 제 2장, 2003
http://www-8.ibm.com/software/kr/rational/products/XDEDeveloper.html
[6] 장 옥 배, 유 철 중, 이 병 걸, etc, 소프트웨어공학, 도서출판한산, 2000p492
[7] Stephen R.Schach, UML과 Java 중심의 구조적 소프트웨어 공학과 객체-지향소프트웨어공학, p125, p149~156, 2001
[8] Cheesman John, Daniels John, and Addison-Wesley, UML components, 2001
[9] 정 국 환, 조 용 길, 송 현 선 , "관리기법/1과 CASE도구간의 연계활용을 위한 기초 연구 ", 한국전산원, 1994. 12
[10] 한국정보산업연합회, 중앙대학교, "효과적 CASE 도입. 운용전략", 정보산업전략연구회 제7회 세미나 자료, 1996. 9