# UML에 기반한 CBSD Process의 Work Flow Model

임성빈°, 송치양, 문창주, 백두권
고려대학교 컴퓨터학과 소프트웨어 시스템 연구실
e-mail: bluesol@swsys2.korea.ac.kr

# A UML-Based Work Flow Model of the Component Based Software Development Process

Sung Bin Yim°, Chi Yang Song, Chang Joo Moon, Doo Kwon Baik
Software System Lab., Dept. of Computer Science & Engineering, Korea Univ

요 약

컴포넌트라는 소프트웨어 개발 기술에 대해서 중요성이 점점 높아만 가고 있는 가운데, 이 컴포넌트 기술를 이용하여 효율적이고, 실용적으로 소프트웨어를 개발 할 수 있도록 컴포넌트 기반의 소프트웨어 개발 방법에 대하여 연구 제시한다. 즉, 컴포넌트가 가지고 있는 구성요소로서 인터페이스라는 것이 있는데 이 인터페이스와 컴포넌트를 이용해서 만들고자 하는 시스템에 대한 요구사항에 맞게 분석 및 설계를 하고, 만들고자 하는 시스템과 컴포넌트의 투명성을 보여 줄 수 있는 방법에 대해서 Work Flow Model과 각각의 프로세스에 의해서 만들어지는 산출물을 통해서 컴포넌트 기반의 소프트웨어 개발 프로세스에 대해서 제시를 한다.

## 1. 서 론

현재 소프트웨어 공학 및 산업분야에서는 구조적인 기법에서 객체지향의 기법으로 세대를 맞이하면서 변화해 왔다. 그러나, 소프트웨어의 규모가 커지면서 객체지향 기법에 대해서 많은 문제점들이 속속 들어 났다. 따라서, 재사용의 개념을 담고 있는 컴포넌트라는 기술이 만들어지고 추진 적으로 도입되고 있다. 그러나, 이 컴포넌트 기술은 과거의 개발 방법 절차로는 컴포넌트라는 새로운 개념에 대해서 충분히 반영 할 수 없고, 효율적으로 소프트웨어 개발을 할 수가 없다. 또한, 컴포넌트의 다양각색의 형태와 상이한 다수의 개발 방법론, 그리고, 복잡한 개발 절차들로 인해 소프트웨어를 개발하는데 오히려 어려움과 복잡성의 문제점만을 남겨주고 있다. 따라서, 본 논문에서는 이런 문제들을 해결하고자, 컴포넌트 기반의 소프트웨어 개발 방법 대해서 소개하고, 개발 절차를 Work Flow Model을 통해 제시해 보았다.

## 2. 관련연구

### 2.1 UML

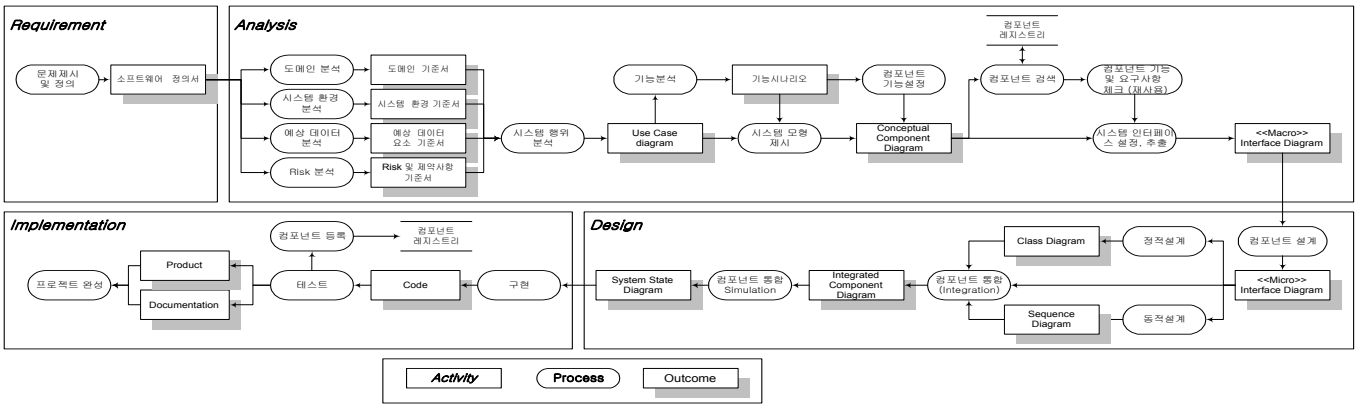
UML은 Booch, Rumbaugh, Jacobson에 의해서 개발되어진 객체지향 분석 및 설계 방법으로 객체지향 개발을 위한 통일된 모델링 언어(Unified Modeling Language)이다. 이 UML은 소프트웨어를 시각화 및 명세화 하여 생성하고 문서화하기 위한 표준화된 언어이며 구현과 상관없이 소프트웨어의 전 개발 과정에 걸쳐 프로세스에서 사용되어지는 방법이다. 특히, 클래스나 아키텍처, 객체에 대해서 관계나 상호작용, 내부행동 등을 여러 종류의 다이어그램을 통해서 표준화된 그래픽컬한 표기법을 사용하여 나타낸다.

### 2.2 Component

컴포넌트는 독립적인 부분으로 교체나 결합 가능한 시스템의 한 부분이라는 의미로서 블랙박스의 형태를 띠고 있으면서 내부구조를 알 수 없는 물리적으로 독립된 소프트웨어를 말한다. 즉, 논리적인 요소를 물리적으로 패키지 화하여 정의가 명확한 인터페이스를 통해서 정보를 전달 할 수 있게 한 응집력이 있는 구조이다. 다시 말해서, 컴포넌트는 내부 구현상황이나 구조에 대해서 알지 못해도 인터페이스와 약간의 명세정보만을 가지고 활용 할 수 있다.

## 3. CBSD Process의 Work Flow Model

본 논문은 <그림1>에서 보여주는 Work Flow Model과 같이 UML을 이용하고 확장시켜서 컴포넌트 기반의 소프트웨어 개발 방법(CBSD)의 Work Flow Model을 제시한다. 일반적으로 새로운 개발 방법론을 이루기 위한 요소로 주로 방법(Method)과 프로세스(Process), 그리고, 지원도구의 툴(Tool)이 있다. 따라서, 본 논문에서 제시하고 있는 CBSD 방법은 먼저, UML이라는 모델링 방법을 중점 구성요소로 놓고, 새로운 다이어그램을 UML에 추가시켜 확장된 형태의 방법을 가진다. 그리고, 개발 프로세스로는, 일반적으로 폭포수 모형의 프로세스를 따랐지만, 컴포넌트의 개념을 포




<그림 1> CBSD Process의 Work Flow Model

함 할 수 있는 서브 프로세스들로 세분화하였다. 그리고, 지원도구로는 컴포넌트라는 것이 만들어지게 된 목적 중에 한 가지로서 재사용(Reuse)의 개념을 지원해주는 컴포넌트 레지스트리[5]라 불리 우는 공유환경에서 컴포넌트를 관리하고 정보를 제공 할 수 있는 도구를 이용하였다. <그림1>에서 CBSD의 Work Flow Model에 관한 세부사항을 나타내었다. 참고적으로, 본 논문에서는 개발 방법 프로세스 진행 중에 불만족한 결과가 나온 경우는 백트레킹(Backtracking)이 필요하다. 그러나, <그림1>에서는 이 과정을 도식화하지 않았다.

### 3.1 Requirement 단계

<그림1>에서 보는 것과 같이 요구(Requirement) 단계에서는 만들고자 하는 소프트웨어에 대한 문제를 확인하고 정의하는 단계이다. 따라서, 문제 제시 및 정의라는 간단한 프로세스를 가지고 있고 이 프로세스는 소프트웨어 정의서 라는 산출물(outcome)을 만든다.

### 3.2 Analysis 단계

분석(Analysis) 단계는 소프트웨어의 본격적인 개발을 하기 위한 시작 단계로서 먼저 소프트웨어의 정의서를 보고 소프트웨어에 관한 도메인, 시스템 환경, 예상되어지는 데이터, 위험도(Risk) 및 제약사항을 조사하여 개발에 필요한 기준서를 만든다. 이렇게 정립하여 서술되어진 기준서를 가지고 만들고자 하는 소프트웨어에 대한 시스템 내의 행위를 뽑아내어 Use Case Diagram을 이용해서 표현한다. Use Case Diagram에서는 시스템의 행위를 기능중심의 동작(Use-Case)으로 보고 작성한다. 이때, 엑터(Actor)라는 요소를 이용해서 시스템의 외부 환경과의 관계를 표현을 한다. 만들어진 Use Case Diagram을 중심으로 전 단계에서 작성되어진 기준서를 참조하여 시스템의 행위에 대한 세부 기능을 분석하고 기능 중심의 시나리오를 작성한다. 기능 중심의 시나리오는 Main Flow, Sub Flow, Alternative Flow라는 3가지의 흐름으로 기능에 대해서 서술한다.

| | |
|---|---|
| Main Flow | 메뉴의 주메뉴처럼 주요 기능의 흐름을 서술 |
| Sub Flow | 주메뉴 아래의 부메뉴 처럼 부기능(sub-function)의 흐름을 서술 |
| Alternative Flow | 예외상황에 관한 흐름을 서술 |

기능시나리오는 시스템의 개념적(Conceptual) 컴포넌트 다이어그램을 만들기 위한 시스템 모형설정에 도움을 주며, 또한 컴포넌트 다이어그램에서 사용되는 하나하나의 컴포넌트에 대해서 컴포넌트를 명세하기도 한다. 즉, 개념적 컴포넌트 다이어그램내의 하나의 컴포넌트에 대해서 기능시나리오 하나 이상이 대입이 되고 UML의 컴포넌트 다이어그램과 같은 방법으로 표현한다.

개념적 컴포넌트 다이어그램이 완성이 되면, 이 다이어그램을 가지고 본격적으로 컴포넌트 기반의 시스템 설계를 하기 위한 구체화 단계로서 요구하는 기능에 맞은 컴포넌트에 대해서 재사용이 가능한지 컴포넌트 검색을 하고 컴포넌트의 외부 인터페이스(External Interface)를 설정한다.

이때, 컴포넌트 레지스트리[5]라는 도구를 이용해서 컴포넌트 저장소(Repository)에 보관되어 있는 컴포넌트를 검색하여 재사용 가능 유무를 판별하고, 사용 가능한 컴포넌트들은 Macro Interface Diagram을 작성하는데 추가를 시킨다. 만약 검색되어진 컴포넌트가 요구사항과 정확하게 맞으면, 그 자체로 가져다가 사용하면 되는데, 요구사항에 맞는 정확한 컴포넌트를 찾기가 매우 어렵다. 따라서, 컴포넌트 레지스트리에서 제공하는 컴포넌트 명세 정보와 컴포넌트가 필요로 하는 제약사항이나 요구사항을 고려해서, Conceptual Component Diagram에서 결정한 컴포넌트와 컴포넌트 레지스트리를 통해서 선택한 컴포넌트를 조합하여 하나의 통합된 새로운 인터페이스나 분할된 인터페이스를 설정한다. 그리고, 다른 요구사항이나 제약조건에 맞게 추출되어진 외부 인터페이스를 가지고 인터페이스의 연관 관계도인 Macro Interface Diagram을 작성한다. Macro Interface Diagram에서는 주로 컴포넌트를 대표 할 수 있는 외부 인터페이스에 관한 사항을 도식화한다.

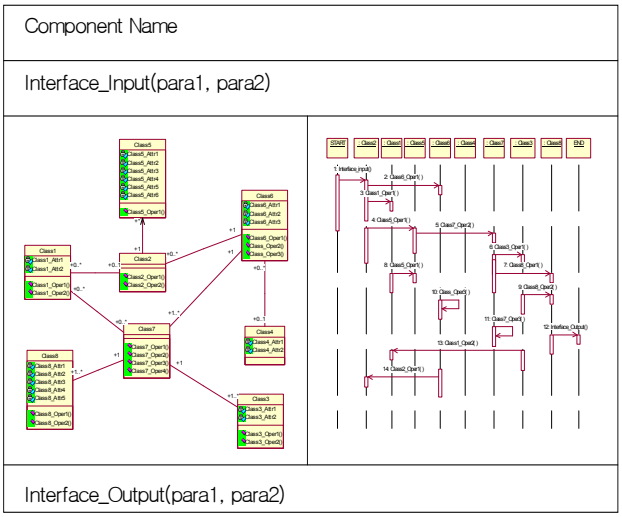
Macro Interface Diagram은 설계의 단계에서나 나올 듯한 다이어그램이지만, 이 단계를 거치면서 만들고자 하는 소프트웨어에 대해서 전반적인 전체 윤곽을 제시할 뿐만 아니라, 시스템의 환경적, 구조적, 기능적 등의 사항들에 대해서 세부적으로 분석된 사항을 표현한다.

### 3.3 Design 단계

설계(Design) 단계는 Macro Interface Diagram을 구성하고 있는 각각의 컴포넌트, 단위 컴포넌트에 관한 세부사항 설계 및 설계되어진 단위 컴포넌트에 대해서 통합하여 모의

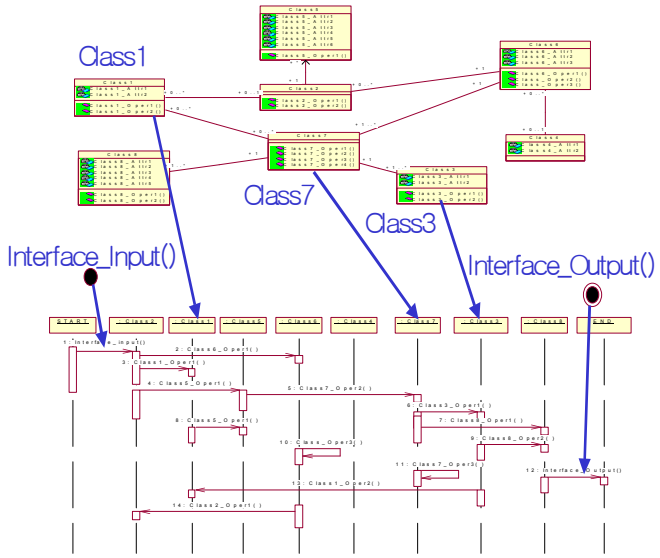
실험인 시뮬레이션(Simulation)을 수행하는 프로세스로 이루어진다.

컴포넌트 설계 단계의 첫 번째 프로세스로 단위 컴포넌트에 관한 설계이다. 본 논문에서는 단위 컴포넌트를 설계하고 명세하기 위해서 <그림2>와 같은 컴포넌트의 Micro Interface Diagram을 제시한다.




<그림 2> Micro Interface Diagram

컴포넌트를 설계하기 위한 참고사항으로 컴포넌트는 크게 외부를 위한 기능과 내부의 기능으로 나누어진다. 외부기능은 다른 환경이나 다른 컴포넌트와의 연결을 목적으로 하는 외부 인터페이스의 기능이다. <그림2>에서 볼 때, Interface_Input()과 Interface_Output()이라는 부분은 컴포넌트의 외부인터페이스를 구성하고 있는 입출력 함수에 해당되는 부분이다.

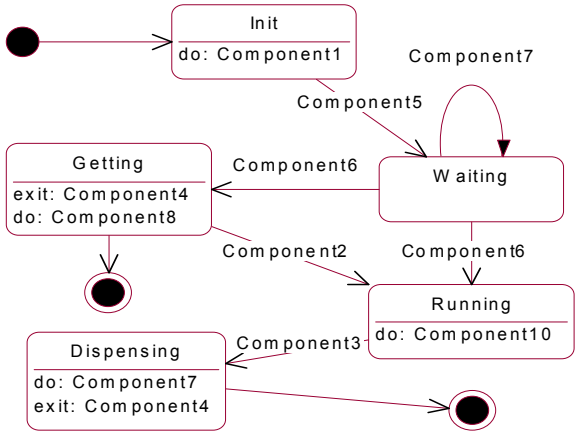



<그림3> Micro Interface 구성하는 요소에 대한 연관 관계도

다음으로 내부기능이 있는데 이는 컴포넌트 밖의 환경과 관계없이 외부 인터페이스를 통해서 들어온 데이터나 전달요소를 가지고 컴포넌트 내에서 자체적으로 수행하는 기능을 말한다. 본 논문에서는 이 컴포넌트의 내부기능을 설계하기 위한 표현수단으로 UML의 Class Diagram과 Sequance Diagram을 사용하였다. 컴포넌트를 제작하기 위해서는 내부적으로 객체기반의 기술이 이용되어진다. 따라서, 컴포넌트의 내부 정적 상태를 위해서 Class Diagram을 사용하여 설계를 하고, 컴포넌트의 동적인 면을 설계하기 위해서는 시간적 흐름 관계도인 Sequence diagram을 이용하여 컴포넌트 내의 정적인 클래스들이 어떻게 동적으로 서로 메시지를 주고 받고 어떤 이벤트 흐름을 가지고 있는지를 표현한다. 실제 UML에서는 Sequence Diagram은 Use Case의 시나리오를 표현하기 위한 수단으로 주로 사용되지만, 본 논문에서는 Sequence Diagram을 이용해서 컴포넌트를 구성하는 클래스들의 동적인 흐름을 나타낸다. 위의 <그림 2>는 단위 컴포넌트를 설계하기 위한 표현 방법으로 단위 컴포넌트가 가지는 모든 기능을 명세하고 있다고 말 할 수 있다.

단위 컴포넌트 설계가 끝나면, 요구되어진 기능을 가지는 시스템을 위한 조립의 단계로 단위 컴포넌트간의 통합(Integration)을 수행한다. 요구사항과 분석 및 설계 단계를 거치면서, Update되거나, 개발, 변형되어진 컴포넌트에 대해서 조립을 함으로서 원하는 시스템을 위한 Integrated Component Diagram을 작성한다. 분석단계의 Conceptual Component Diagram이 개략적인 다이어그램이라면, 이 Integrated Component Diagram은 컴포넌트에 대해서 상세 설계하고 통합시킨 시스템의 모든 것을 표현한 다이어그램으로, 지금까지의 모든 프로세스에 의해서 만들어진 총 결과의 산출물이라고 말할 수 있다.

다음으로 Integrated Component Diagram을 기반으로 System State Diagram을 만든다.




<그림 4> System State Diagram

System State Diagram은 요구되진 시스템을 위해서 통합되어진 컴포넌트들이 기능을 수행할 때, 서로 어떠한 상태(State)로 컴포넌트들이 수행되고 있는지 보여 줄 수 있는 상태 표현도이다. 표현법은 UML과 같은데 UML에서는 하나의 클래스 내에서 클래스의 상태를 나타내기 위해서 사용한다. 그러나, 본 논문에서는 통합되어진 컴포넌트들이 어떠한 연관 동작을 하고, 어떤 상태로 활동을 하는지 표현하기 위해서 사용을 한다. 그리고, 이 System State Diagram을 이용하여 시스템 내의 컴포넌트들간의 흐름을

비교하여 시스템이 요구한 사항에 맞게 운영이 되는지, 또는, 정확한 기능을 수행하는지에 대해서 가상운영 기법인 정형적(Formal) 시뮬레이션[6]을 수행한다. 단계별 트레킹(tracking)과 가상 데이터의 대입 방법으로 시스템의 기능과 동작에 대한 상태를 분석 및 시뮬레이션[6] 함으로써 소프트웨어가 만들어지기 전에 정형검증을 하고, 품질과 성능에 대해서 평가를 할 수 있게 한다.

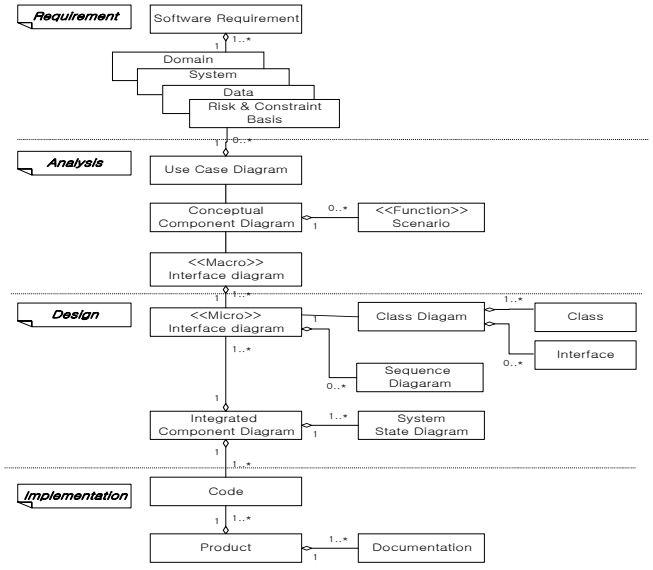
### 3.4 Implementation 단계

마지막으로 분석과 설계를 마친 산출물들을 가지고 구현(Implementation)의 단계에 들어간다. 구현 단계는 최후의 단계로서 충분히 분석되어지고 설계된 컴포넌트에 대해서 코드를 만들고, 완성품에 대한 테스팅(testing) 과정을 거쳐 최종적으로 소프트웨어를 만들어 내게 된다. 테스팅을 거쳐 품질이 검증된 컴포넌트는 다시 재사용을 하기 위해서 컴포넌트 레지스트리[5]라는 곳에 등록을 시켜 다음에 다른 소프트웨어 개발에 재사용을 한다. 본 논문에서는 분석 및 설계에 중점을 두었기에 마지막 단계인 구현 단계는 일반적인 개발 프로세스만을 사용한다.

## 4. 프로세스 산출물

### 4.1 산출물들간의 관계도

지금까지 크게 요구사항과 분석 및 설계, 구현 단계를 거치면서, 각 단계별의 세부 프로세스를 알아보았고 각 프로세스별로 생성되는 산출물(outcome)에 대해서 알아보았다. 본 논문에서는 UML에서 사용하는 다이어그램을 재배치 및 확장을 시켜 주로 이용했는데, 이 다이어그램, 프로세스에 의한 산출물들은 서로 연관 관계를 가지고 있다.  이 연관 관계를 <그림5>에서 나타내었다.




<그림5> Work Flow Model 내의 산출물의 관계도

### 4.2 기대효과

소프트웨어는 규모가 커지면서 실코드와 실행 파일뿐만 아니라 개발에 따른 많은 산출물들이 필요하다. 그래서, 위의 <그림 5>에서 보는 것처럼 각각의 프로세스별로 생산된 산출물에 대해서 서로의 연관 관계가 있으므로 이 관계를 고려해서 분류를 한다면 보다 효율적으로 개발 관리를 할 수 있다고 본다. 그리고, 객체보다는 크지만, 작은 단위로서 컴포넌트라는 것이 투명성이 없는, 즉, 내부적으로 볼 수 없는 블랙박스의 형태를 나타내고 있으므로 개발되어지거나, 개발할 컴포넌트에 대한 컴포넌트의 투명성을 확실하게 보장해 줄 수 있는 명세서가 필요하다. 이 컴포넌트의 투명성을 보장해주는 명세서들을 모으면, 이 명세서들은 다시 컴포넌트보다 큰 의미의 소프트웨어의 투명성을 보장하는 것이 된다. 따라서, 본 논문에서 제시한 컴포넌트 기반 개발 방법 프로세스 및 산출물들은 소프트웨어를 개발하면서, 도메인이나 시스템 등의 요구사항에 대해서 유연성(Flexibility)을 제공하며, 무엇보다도 UML에서 사용하는 다이어그램을 이용하여 사용하기 쉽고(Use to Easy), 유지(maintenance)하기가 쉽다. 게다가 프로젝트를 관리하는데 좋은 지지(Good Support)를 제공하며 일관된 관리(Consistency)를 할 수 있게 한다.

## 5. 결론

본 논문에서는 컴포넌트라는 기술을 접목시켜서 가장 일반적으로 소프트웨어를 개발 할 수 있는 방법 절차에 대해서 세분화된 프로세스의 Work Flow Model과 산출물(Outcome)에 대해서 알아보았다. 본 논문은 세부 프로세스에 대해서 깊은 절차와 양식에 대해서는 언급하지 않았고 Case Study의 예제를 보여주지는 않았다. 그러나, 컴포넌트라는 기술 개념을 도입한 소프트웨어 개발에 접목시킬 수 있는 방법으로 충분하다고 본다.

앞으로 추후 연구과제로는 컴포넌트 기반의 소프트웨어 개발 방법을 중심으로 이 방법 절차와 산출물을 지원해주는 개발 툴을 만들고 현재 컴포넌트를 관리하는 컴포넌트 레지스트리[5]와 같은 도구들을 접목시켜 개발부터 품질평가, 저장 및 관리 등의 기능을 수행하는 CBDMS(Component Based Development & Management System)를 개발하는 것이다.

## 6. 참고문헌

[1] Desmond F. D'souza and Alan C. Wills, "Objects, Components, and Frameworks with UML", Addison-Wesley, 1998.

[2] Grady Booch, James Rumbaugh, Ivar Jacobson, "The Unified Modeling Language User Guide", Addison-Wesley, 1999

[3] "The Proceedings of International Workshop on Component-Based Software Engineering", LA, USA, 1999 ( http://www.sei.cmu.edu/cbs/icse99 )

[4] "The Proceedings of International Workshop on Component-Based Software Engineering", Kyoto, Japan, 1998 ( http://www.sei.cmu.edu/cbs/icse98 )

[5] 임성빈, 문창주, 백두권, “통합된 컴포넌트 공유환경 구축을 위한 컴포넌트 레지스트리 설계”, 한국정보과학회 ‘99 가을 학술발표논문집 제26권 2호, pg441-443, 1999

[6] "StateMate Magnum Quick Reference Guide Release 2.0", I-Logix, 1999