영역기반 아키텍처 모델에 관한 연구

오영배

수원여대 컴퓨터응용학부

A Study on the Domain Based Architecture Model

Young Bae Oh

Division of Computer Application, Suwon Women's College

E-mail : yboh@suwon-c.ac.kr

요 약

최근의 소프트웨어 위기를 해결할 수 있는 최적의 대안으로서 컴포넌트 기술을 개발, 산업계에 확산하기 위하여 다양한 영역의 비즈니스 컴포넌트를 개발 보급하여야 한다. 비즈니스 컴포넌트를 개발하기 위하여는 컴포넌트의 조립 모델을 정의한 개념적 아키텍처를 우선적으로 설정하여야하며 이 모델에 기반한 체계화된 명세 방법을 정의하여 각 컴포넌트가 일관적 체계에 따라 명세화 되어 개발할 수 있는 체계를 구축하여야 한다.

본 연구에서는 정통부 사업의 일환으로 수행된 공용 컴포넌트 뱅크 구축을 위한 영역 아키텍처 모델을 제시하고, 컴포넌트 뱅크에 구축할 컴포넌트 개발을 위한 명세 방법의 개념 모델과 명세 단계를 제시하고 이를 비즈니스 응용 영역에 적용하여 개발한 명세화 접근 방법 및 개발 결과에 대하여 논한다.

1. 서론

향후 거대한 시장을 형성할 것으로 예상되는 비즈니스 컴포넌트의 확대 보급을 위하여 MS ADCU(Microsoft Application Developers Customer Unit), OMG(Object Management Group) 및 OAG(Open Application Group)등에서는 비즈니스 각 영역별로 컴포넌트 사양의 표준화를 추진하고 있다[1][2]. MS, Oracle, SUN 등은 컴포넌트 플랫폼 모델을 기반으로 하여 비즈니스 컴포넌트 어플리케이션을 구현할 수 있는 비즈니스 컴포넌트 아키텍처를 개발하여 제시하고 있다.

본 연구에서는 이에 대응하는 우리의 컴포넌트 명세를 개발하기 위한 단계적 접근을 시도하며, 명세 개발을 위한 영역 선정은 다음 기준에 따랐다.

● 업무의 정형화 정도

소프트웨어의 컴포넌트화는 업무 프로세스를 부품화 하는 과정이므로 프로세스의 정형화 및 표준화가 전제되어야 한다. 업무의 정형화 정도가 높은 영역부터 컴포넌트화를 추진하며 점차적으로 비정형적인(Ad-hoc) 프로세스의 정형화를 유도하여 컴포넌트화 한다. 궁극적(장기적)으로는 기존 시스템(Legacy System)의 재구조화(Reengineering)를 유도해야 할 것으로 본다.

● 개발 제품의 사용자 구체성

컴포넌트 개발은 개발 제품의 상품화를 위한 것이기 때문에 제품의 사용자에 대한 명확한 정의가 전제되어야 한다. 이를 위해서 컨소시엄의 사용자 그룹 의견을 반영하여야하며 영역 선정 및 개발과정 상에서 사용자와의 상호작용(interaction)이 필요하다.

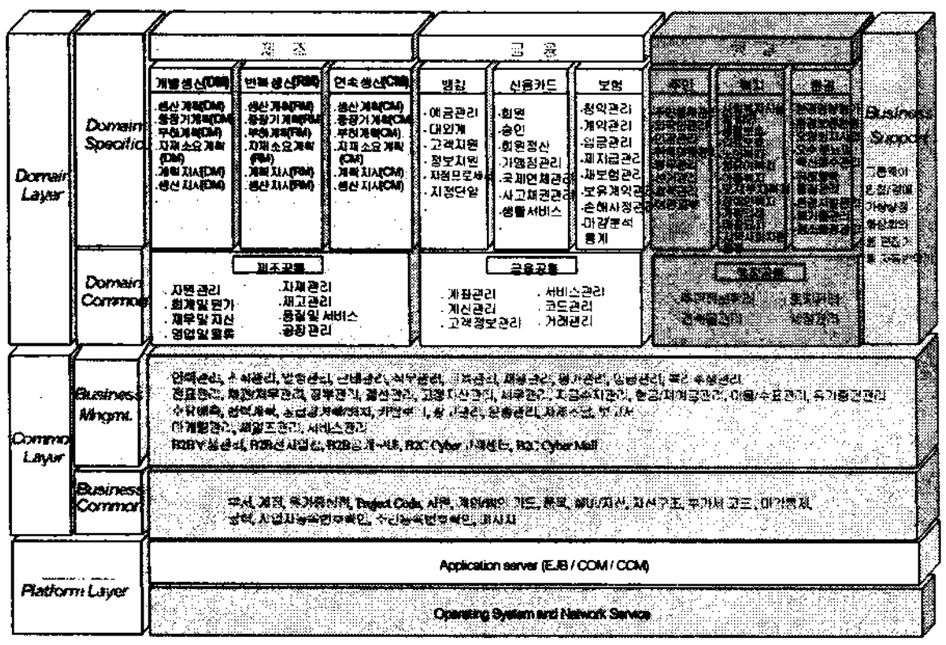
● 개발 제품의 파급 효과

컴포넌트는 영역 분석을 통한 공통성 있는 프로세스의 구현을 지향한다. 이는 개발된 컴포넌트의 사용의 극대화를 도모하는 것으로 파급 효과가 높은 분야를 선정한다. 파급 효과와 사용자 구체성은 서로 상충되는 면을 가지고 있는데, 예를 들면 비즈니스 공통 컴포넌트는 파급효과는 높지만 사용자 구체성이 떨어지고 사용하기 어려운 점이 있고 영역별 컴포넌트는 사용자 구체성이 높고 사용하기 쉬우나 파급효과는 떨어진다.

● 국제표준과의 적합성

정보기술(IT)환경의 국제화(global)에 따라 비즈니스 프로세스도 세계표준과 맞추어져 가고 있다. 또한 컴포넌트의 수출을 위해서도 국제표준과의 호환성을 염두에 두어야 한다. OMG, MS ADCU등은 선정된 영역별로 컴포넌트 표준화 작업을 진행하고 있다. 이들 표준이 컴포넌트 시장에 영향력을 발휘할 것으로 예상되며, 이들 영역 표준과의 적합성을 영역 선정의 기준에 반영할 필요가 있다.

2. 공용 컴포넌트 아키텍처



<그림 2.1> 공용 컴포넌트 아키텍처

본 연구에서 명세한 공용 컴포넌트 뱅크는 기업에서의 활용성 및 재사용성이 높은 응용인 제조, 금융 및 행정 영역을 주 대상으로 하며 컴포넌트의 아키텍처는 크게 3개로 구성되는 계층 구조를 이루고 있다.

2.1 응용 계층(Domain Layer)

응용 계층은 제조, 금융 및 행정의 3개 영역을 대상으로 하며 각 영역의 단위 업무를 수행하는 컴포넌트로 이루어진다. 응용 계층의 컴포넌트는 3개 계층 중에서 단위 컴포넌트의 크기(granularity)가 가장 크며 단위 업무의 응용 로직(logic)을 구현한 단위 프로세스(process)를 이루고 있다.

응용 계층은 세부 응용 업무를 수행하는 세부 응용(Domain Specific) 컴포넌트와 각 영역별로 공통적으로 재사용될 수 있는 응용 공통(Domain Common) 컴포넌트로 구분된다.

2.2 공통 계층(Common Layer)

공통 계층은 응용 컴포넌트를 구성할 수 있는 공통 컴포넌트로 구성된다. 공통 컴포넌트는 영역에 특정되지 않는 컴포넌트로서 응용 컴포넌트 및 특정 응용을 개발하기 위하여 재사용 빈도가 높은 컴포넌트들로 구성되는 계층이며, 컴포넌트의 크기는 응용 계층에 비하여 작다.

공통 계층은 영역에 특정되지 않고 공통으로 사용되는 단위 업무들로 구성된 단위 업무(Business Management) 컴포넌트와 단위 업무로서보다는 업무를 구성하는 부품으로서 재사용 빈도가 높은 업무 공통(Business Common) 컴포넌트로 구성된다.

2.3 플랫폼 계층(Platform Layer)

플랫폼 계층은 컴포넌트를 연결하고 컴포넌트 간의 통신 기능을 제공하는 등의 컴포넌트 수행 인프라(infra) 기능을 실행하는 계층이다. 이 계층은 EJB, COM, CCM 등의 컴포넌트 실행 플랫폼 및 운영체제와 네트워크 계층으로 구성되며 본 연구의 명세에 포함되지 않는다.

3. 컴포넌트 명세 방법

본 연구에서는 컴포넌트 명세 개발자의 영역 지식(Domain Knowledge)을 바탕으로 한 영역 분석(Domain Analysis)을 통한 시스템 아키텍처를 도출한 후, 영역 분류를 하여 후보 컴포넌트를 식별하고 컴포넌트 상호 관계도를 분석하여 세부 기능별 컴포넌트간의 상호관계를 파악하여 컴포넌트를 식별하고 컴포넌트 다이어그램 및 컴포넌트 인터랙션 다이어그램 등을 통하여 컴포넌트 인터페이스 명세를 도출하는 하향식 접근 방법(Top-Down Approach)을 사용하였다.

컴포넌트를 식별하기 위하여 사용한 원칙은 다음과 같다.

● 컴포넌트는 잘 정의된 외부 인터페이스를 가지는 소프트웨어 단위이다.

● 컴포넌트는 다양한 소프트웨어 제품의 제작을 위해 다른 컴포넌트들과 결합될 수 있다.

● 컴포넌트는 구현과 관련되는 특정한 플랫폼과 미들웨어에 독립적이다.

● 컴포넌트는 응용 영역의 업무 프로세스(Business Process)를 기초로 단위 업무를 중심으로 식별되었다.

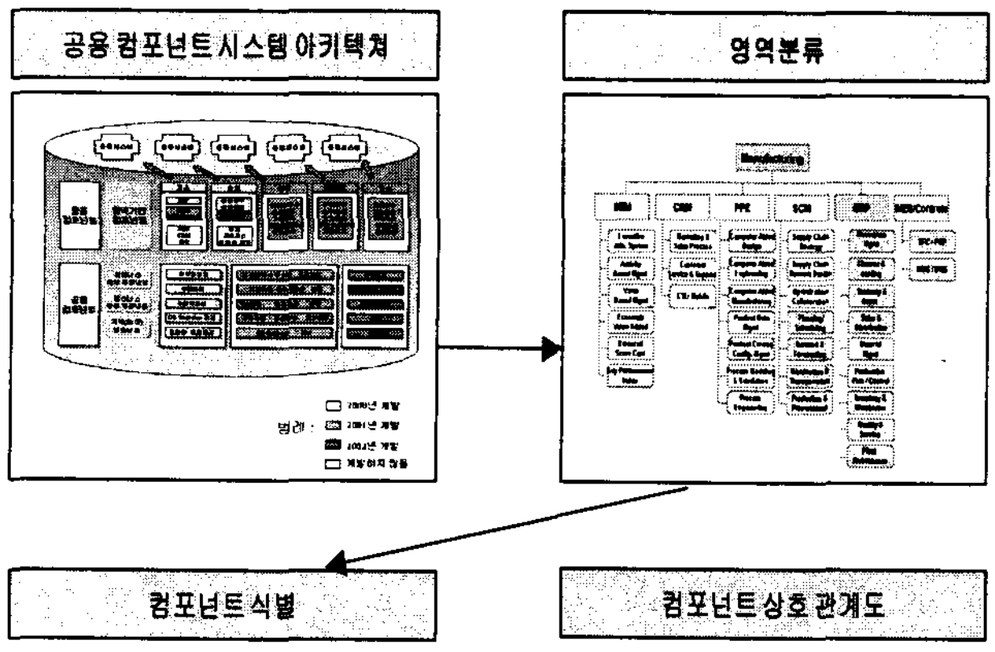
컴포넌트 명세는 다음과 같이 8개의 단계를 거쳐 도출된다.

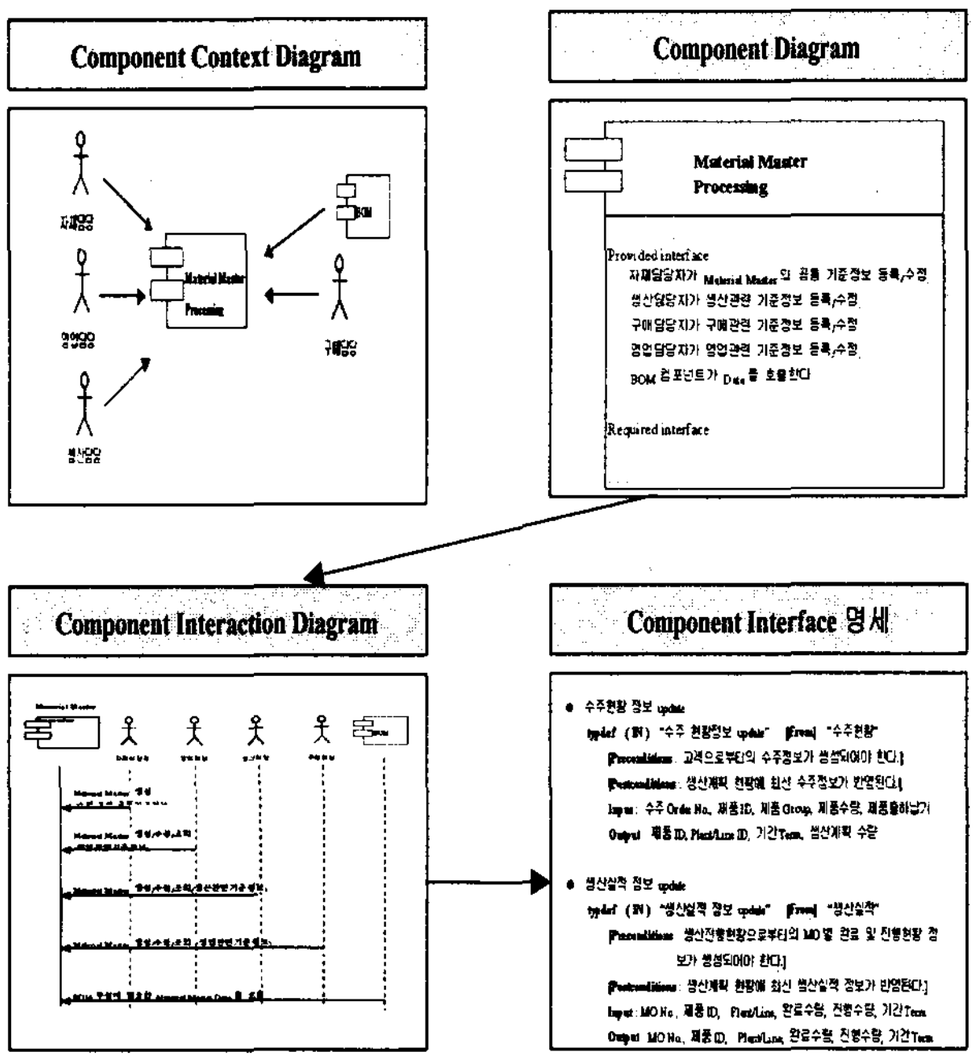
3.1 공용 컴포넌트 계층 구성도

명세하고자 하는 컴포넌트들의 범위, 목적, 기능 등을 기술한 다이어그램(diagram)이며 영역 분석을 통한 컴포넌트의 계층 구조를 나타내고 응용 컴포넌트와 공통 컴포넌트를 구분하고 각 계층의 정의 및 구분 기준을 제시한다.

3.2 영역 분류도

영역의 기능을 하향식으로 분류하여 트리 구조를 가지는 다이어그램을 구성한다. 루트 노드(Root Node)는 영역 이름이 되며 노드의 수준(level)이 증가함에 따라 영역내의 분야 또는 기능들을 세분화한





<그림 3.1> 컴포넌트 명세방법

다. 단말 노드는 컴포넌트 분류표의 제일 왼쪽 항목 즉 기능을 나타내며 단말 노드들에 속하는 컴포넌트들에 대한 명세서의 개발 시기를 알아보기 쉽도록 구분하여 단말 노드에 표시한다.

3.3 컴포넌트 분류표

영역내에서 도출되어 명세하고자 하는 컴포넌트들을 기능별로 분류한다. 분류표의 항목은 기능, 세부기능, 컴포넌트로 구성한다. 분류표의 제일 왼쪽 항목인 기능은 영역 분류도에 단말 노드로 나타나며 분류표의 모든 항목에는 이름과 분류코드를 포함한다. 분류코드는 영역별로 각 컴포넌트를 식별하기 위한 고유한 식별자(ID)이다.

3.4 컴포넌트 상호 관계도

영역내에서 도출되는 각 컴포넌트간의 관계를 나타낸다. 관계의 복잡도, 세부 기능 분류에 따라 적절한 수의 컴포넌트들로 구성함으로써 컴포넌트간의 관계를 파악하고자 하는 것이 목적이다.

컴포넌트 분류표에 따라 기능과 관련된 세부기능을 표시하며 다이어그램은 컴포넌트와 컴포넌트간의 호출관계를 나타내는 화살표로 이루어진다. 패키지 형식으로 묶여진 컴포넌트들은 동일한 세부기능 내에 있는 컴포넌트들을 의미한다. 표시된 기능 및 세부기능에 관련된 모든 컴포넌트가 다이어그램에 표시되어야 하며 컴포넌트 개수가 많아질 때는 여러 페이지에 걸쳐 분리하여 표현할 수 있다.

3.5 컴포넌트 문맥 다이어그램(Component Context Diagram)

명세하고자 하는 컴포넌트와 그 컴포넌트의 외부 개체와의 관계를 그림으로 표현한다. 외부 개체는 행위자(actor) 혹은 컴포넌트로 표현하며 행위자는 컴포넌트로 명확하게 나타나지 않는 경우에 표시한다. 컴포넌트로 들어오는 화살표는 외부 개체가 컴포넌트가 제공하는 인터페이스(Provided Interface)를 요청한다는 것을 나타낸다. 컴포넌트가 사용하는 다른 컴포넌트의 인터페이스(Required Interface)는 컴포넌트에서 나가는 화살표로 나타낸다.

3.6 컴포넌트 다이어그램

컴포넌트 자체의 명세를 간략하게 설명하기 위하여 컴포넌트 이름과 컴포넌트 인터페이스로 구성된다.

여기에서는 컴포넌트의 불변 성질(invariant)과 가변성(variability)을 기재하는데 불변 성질은 컴포넌트 기능 수행을 위하여 변경되지 않아야 할 특성으로 컴포넌트의 모든 인스턴스에 대해 만족되어야 하는 조건을 정의한다. 다시 말하면, 컴포넌트 자체에 관한 사항으로 변하지 않고 지켜야 하는 조건을 명시한다. 가변성은 컴포넌트가 가질 수 있는 여러 가지 가능한 경우가 있을 경우를 명시하는 것이다.

3.7 컴포넌트 상호 작용도(Component Interaction Diagram)

컴포넌트와 외부 개체들이 어떤 식으로 상호작용 하는가를 표현한다. 해당 컴포넌트와 수직선(Vertical Bar)은 진하게 표시되며 가장 왼쪽에 위치하고, 관련된 컴포넌트들은 점선의 수직선과 함께 차례대로 나열한다. 행위자도 마찬가지로 점선의 수직선과 함께 나타내고 인터페이스들 간에 순서가 있는 경우는 번호를 붙여서 순서 관계를 명시한다. 컴포넌트 내에 시나리오별로 여러 순서(sequence)의 사용예가 있는 경우는 각각 다른 다이어그램으로 기술하며 인터페이스 기술은 매개변수(parameter)를 생략하여 명세를 단순화한다. 요구 인터페이스는 "->" 화살표로 나타내고 제공 인터페이스는 "<-" 화살표로 나타내며, 인터페이스를 정확하게 기술하기 위해 선택사항으로 [ ]내에 조건(guard)을 사용할 수 있다.

3.8 컴포넌트 인터페이스 명세

컴포넌트 인터페이스 명세는 다음과 같은 문법으로 작성한다.

```
{Required|Provided} Interface "인터페이스_이름"
TO {"external components"|"Actor"}
  Description: <description>
  Preconditions: {<description> | None}
  Postconditions: {<description> | None}
  Input: {<input data> | None}
  Output: {<output data> | None}
```

[그림 3.2] 컴포넌트 인터페이스 명세

각 항목에 대한 설명은 다음과 같다. 여기에서 해당사항이 없는 경우에는 **None**으로 표시한다.

- Required Interface: 컴포넌트가 사용하는 다른 행위자(actor)나 컴포넌트의 인터페이스를 의미하며, TO 다음에 다른 행위자나 컴포넌트를 기술한다.
- Provided Interface: 컴포넌트가 외부에 제공하는 인터페이스를 의미하며, TO 다음에 상대 컴포넌트나 행위자를 기술한다.
- Description: 개요에 해당하는 것으로 컴포넌트의 전체적인 기능을 간단, 명료하게 설명한다.
- Preconditions: 컴포넌트 수행 전에 반드시 만족해야 하는 사전 조건을 나타낸다. 만일 컴포넌트가 기대되는 값을 가지지 않을 경우 기능이 수행되지 않는다. 이 조건을 이용하여 최소한의 처리 오버헤드로도 오류를 검출할 수 있다.
- Postconditions: 컴포넌트의 연산 수행 후에 반드시 만족해야 하는 사후 조건을 나타낸다. 컴포넌트 기능의 수행 후의 결과를 나타낸다. 이 조건을 이용하여 인터페이스와 구현을 분리하여 컴포넌트가 어떻게 수행되는지 기술하지 않고도 연산을 나타내는 유용한 수단으로 오류를 검출할 수 있다.
- Input: 컴포넌트로 입력되는 데이터를 나타낸다.
- Output: 컴포넌트로부터 외부로 출력되는 데이터를 나타낸다.

4. 공용 컴포넌트 명세

본 연구에서 개발한 컴포넌트 명세 방법에 따라서 제조 및 금융 등의 응용 영역 및 공통 컴포넌트에 대한 명세를 개발하였다[3]. 각 영역별로 우선적으로 100개 정도의 컴포넌트를 식별하여 총 500개 정도의 컴포넌트에 대한 명세를 개발하였다. 각각의 영역에 대하여는 영역별 명세서를 작성하였으며 일관된 표기규약을 적용하였다.

명세서를 작성할 때 고려한 원칙은 다음과 같다.

● 컴포넌트 명세는 컴포넌트의 외부 동작(External Behavior)을 인터페이스로서 기술한다.

● 컴포넌트 명세는 컴포넌트 인터페이스에 대한 제한 사항을 전제 조건(preconditions)과 사후 조건(postconditions) 등을 사용하여 기술한다.

● 컴포넌트 명세는 인터페이스의 행위, 순서 관계, 그리고 존재하는 부작용(Side-Effects)을 명세화한다.

● 명세는 정확하고 기능적으로 완전해야 하며, 숨겨진 인터페이스가 없어야 한다.

● 컴포넌트가 구현한(제공하는) 인터페이스와 컴포넌트가 사용하는 인터페이스를 구분한다.

● 기존 명세에 대한 변경 또는 확장이 가해지는 경우에는 그 정당성을 입증하고 충분히 명세화해야 한다.

● 여러 문맥으로 사용될 수 있는 기능들을 분해하고 인터페이스들을 분리하여 명세화한다.

● 다른 인터페이스 명세에 대한 종속성을 최소화하고, 기존의 표준 문서들에 적합하도록 인터페이스를 명세화한다.

● 구현에 대한 융통성을 최대로 고려해야 하고, 결합될 다른 컴포넌트나 어플리케이션과의 상호운용성(interoperability)도 고려한다.

5. 결론 및 향후 계획

본 연구에서는 비즈니스 응용 영역별로 활용가능한 컴포넌트를 개발하기 위하여 컴포넌트를 식별하고, 컴포넌트를 명세하기 위한 명세 방법을 개발하며 이를 적용하여 컴포넌트를 명세 하였다.

본 연구에서 개발한 컴포넌트 명세방법은 비즈니스 컴포넌트를 명세 하는 기법을 제시하고 이를 영역별 컴포넌트 명세 개발에 적용함으로써 산업계에서 활용할 수 있게 했다는 점에서 큰 의의를 가지며, 향후 이를 보완하여 개발 프로세스에 대한 품질을 예측하고 평가하는 수단을 제공하고, 시스템의 관계당사자에게 시스템을 분석 평가할 수 있는 프레임워크를 제공하고, 개발후 또는 유지보수시에 시스템을 이해하고 분석할 수 있는 표현수단을 개발할 예정이다.

[참고문헌]

[1] OMG, OMG Special Interest Groups, Task Forces and The Architecture Board, https://www.omg.org/techprocess/sigs.html#dtc, Aug. 1999

[2] OAG, Open Applications Group Integration Specification, Release 6.1, Dec. 1998

[3] 오영배 외, 영역별 컴포넌트 명세서, 한국전자통신연구원, 1999년 12월