CBSD 활성화를 위한 확장된 부가가치 중개 개념

심 우 곤[†]·백 인 섭[††]·이 정 태[†††]·류 기 열[††††]

요 약

본 논문에서는 컴포넌트에 기반한 소프트웨어 개발(CBSD) 환경의 전반적인 활성화 개념을 제시한다. 컴포넌트 기술에 대한 연구가 컴포넌트를 시스템 구축에 효율적으로 적용하려는 쪽으로 집중되어, 컴포넌트 시장의 활성화 방안이나 컴포넌트 자체의 개발 방식 등에 대해서는 지속적인 연구가 필요한 상태이다. 우선 CBSD 활성화로의 장애요소를 진단하고 이를 해결하기 위한 방안으로 중개개념 고려의 필요성을 강조한다. 그러나 단순한 사전적 의미로의 중개개념으로는 CBSD 활성화를 도모할 수 없으므로 중개개념에 부가적인 서비스를 추가한 "부가가치 중개 개념" 이 요구된다. 부가가치 중개 개념에는 크게 1) 도메인 아키텍처 지향의 컴포넌트 생산 촉진, 2) 지능형 컴포넌트 검색 서비스, 3) 화이트 박스 서비스의 세 가지 기능을 수행한다. 도메인 아키텍처 지향의 컴포넌트 생산 촉진은, 균형 잡힌 컴포넌트 생산과 아키텍처 중심의 대단위 재사용을 피할 수 있다. 지능형 컴포넌트 검색 서비스는, 컴포넌트 생산자와 소비자 간의 1 : 1 거래의 한계를 해결해주고 마지막으로 화이트 박스 서비스는 컴포넌트 도입의 가장 큰 걸림돌인 유지보수 문제를 보장해준다. 특히 이 개념은 국내 컴포넌트 개발의 특수 상황에 적용하여 큰 효과를 얻을 수 있을 것으로 기대된다.

The Value-Added Brokerage Concept for Steering the CBSD Environments

Woo-Gon Shim[†]·In-Sup Paik[††]·Jung-Tae Lee[†††]·Ki-Yeol Ryu[††††]

ABSTRACT

In this paper, we propose a steering concept that considers overall aspects in the CBSD (Component-Based Software Development) environments. While many researches which are concentrated on using components, market promotion and component development itself seem to be insufficient. To overcome this problem, we introduce a brokerage concept called "Value-Added Brokerage Concept" that provides the following three services : 1) domain architecture-based component promotion, 2) intelligent component search, and 3) white-box service. Domain architecture-based component promotion facilitates balancing component production and promoting architecture-level large scale reuse. Intelligent component search enables to overcome the long time search and selection problem. Finally, white-box service is for solving maintenance problems, which is one of the most critical problem in the CBSD environments. Especially, we expect this proposed concept would be well adapted to our national environments.

키워드 : 소프트웨어 컴포넌트 환경의 활성화 방안, 컴포넌트 기반 소프트웨어 개발(CBSD: Component-Based Software Development), 컴포넌트 기반 소프트웨어 공학(CBSE: Component-Based Software Engineering), 부가가치 중개개념(Value-Added Brokerage Concept), 소프트웨어 아키텍처(Software Architecture)

1. 서 론

소프트웨어 시스템의 규모와 복잡도가 급격히 증가함에 따라 전통적 개발방식에 대한 한계에 직면하게 되었다. 최근 활발히 연구가 진행되는 컴포넌트 기반 소프트웨어 개발(Component-Based Software Development : CBSD)과 소프트웨어 아키텍처는 소프트웨어 업계의 직면한 문제를 해결하는 패러다임으로 받아들여지고 있다[1-3, 5, 7, 23]. 컴포넌트 기술은, '소프트웨어 프로그래밍'에서 하드웨어 개발환경에서처럼 소프트웨어 플러그-앤-플레이(plug and play) 방식으로 시스템 구축하는 '합성을 통한 시스템구축'으로의 전환을 목적으로 한다[5, 7]. 즉, 대규모 시스템을 소규모 컴포넌트들의 합성을 통해 빠르고 높은 신뢰도를 유지하면서 구축이 가능한 획기적인 방안이다.

70년대 대규모 소프트웨어 시스템을 개발하는데 발견된 많은 문제점의 해결책으로 구조적 소프트웨어 설계(software design)가 제시되어 많은 소프트웨어 공학자들의 관심을 끌었다. 이러한 연구는 통합설계와 소프트웨어 시스템 배경분석기법(context analysis) 등을 거쳐 90년대 소프트웨어 아키텍처에 이르렀다[20]. 소프트웨어 아키텍처 기술은 복잡한 대규모 시스템의 빠르고 정확한 이해를 도모하고, 다수의 목표 소프트웨어 시스템과 이해관계에 있는 다수의 사람 또는 조

† 준 회 원 : 아주대학교 대학원 정보통신전문대학원
†† 정 회 원 : 아주대학교 정보 및 컴퓨터 공학부 교수
††† 정 회 원 : 아주대학교 정보 및 컴퓨터 공학부 교수
†††† 종신회원 : 아주대학교 정보 및 컴퓨터 공학부 교수
논문접수 : 2001년 10월 5일, 심사완료 : 2001년 11월 27일

직(stakeholder)들의 요구 사항들을 파악하고 조율함으로써 합리적 시스템 구축을 목표로 하고 있다[3, 20].

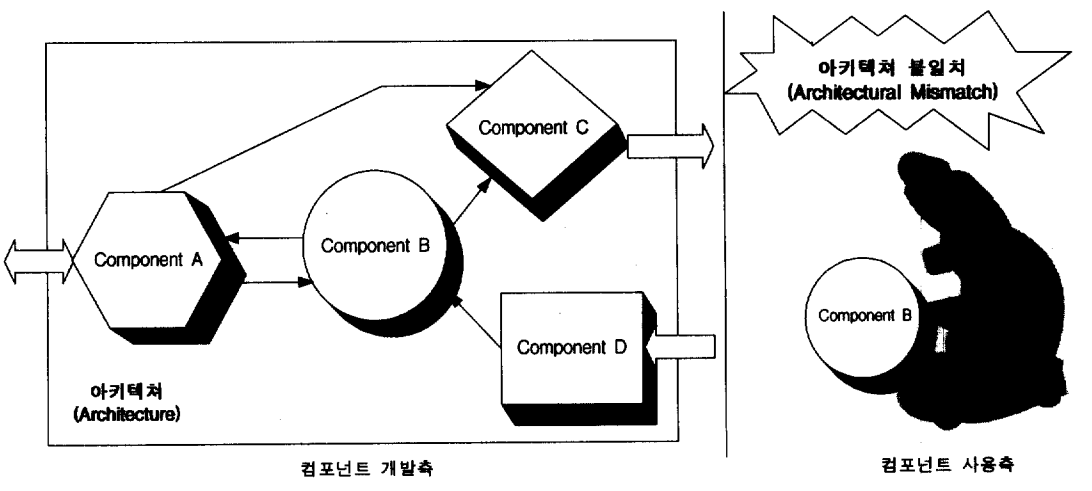
CBSD에 대한 연구는 활발하게 진행 중에 있으나, 컴포넌트의 개발, 유통, 활용의 세 부분 중 컴포넌트를 활용하여 시스템을 구축하는 부분에 초점이 맞추어져 있어서 그 이외 부분에 대해서는 연구가 부족한 상황이다. 한편 최근 컴포넌트의 개발에 대해 깊이 있는 연구가 진행되고 있는데[9] 이는 매우 바람직한 현상이다. 그러나 컴포넌트의 연구의 편중으로 인하여 컴포넌트 기술의 활성화에 많은 문제점이 발견되었다. 전통적인 개발 프로세스 모델은 처음부터 개발(develop from scratch)방식을 가정하고 있어서 재사용 중심의 개발환경에는 많은 도움을 주지 못하고 있다[1, 6, 22]. 최근 컴포넌트와 아키텍처를 접목시킨 개발 프로세스 모델이 등장하고 있는데[6, 9, 22, 27], 이는 매우 바람직한 현상이라고 할 수 있다. 그러나 일련의 모델들은 CBSD 환경에서의 개발 프로세스 모델임에도 불구하고 전통적 개발환경과는 차이가 있는 CBSD의 조직환경(Component Producer-Broker-Consumer : 컴포넌트 생산자-중개자-소비자)[17]을 반영하고 있지 못하다. 이에 CBSD의 조직환경을 감안하고 컴포넌트 기술의 지속적인 활성화를 위한 방안의 모색이 요구된다. 본 논문에서는 CBSD활성화로의 장애요소는 어떠한 것들이 있는지를 진단하고 그 해결방안으로 '부가가치 중개 개념(Value Added Brokerage Concept)'을 제안하고자 한다. 또한 제안된 개념이 국내환경의 환경을 감안하였을 때에도 컴포넌트 활성화가 도모됨을 보인다.

2. CBSD 활성화에 대한 장애 요소

기존 개발환경의 문제점을 해결하기 위해 출현한 CBSD는 전통적인 방식과는 개념적으로나 실질적으로 많은 차이가 있다. 그럼에도 불구하고 아직 그 역사가 일천하여 다차원적이고 다각적인 고려가 요구된다. 실제로 대다수의 연구들은 어느 정도 성숙되고 갖춰진 컴포넌트 환경을 가정하고 그 위에서 어떻게 컴포넌트를 효과적으로 사용할 수 있을지에 관한 것들이다. 즉, 컴포넌트 환경이 활성화되기까지의 과정은 고려의 대상이 되지 않고 있다는 것이다. 또한 다수의 연구가 컴포넌트를 사용하는 소비자(통합자) 관점을 중심으로 진행되고 있어서, 컴포넌트 자체의 개발이나 컴포넌트 생산과 소비영역간 연계를 고려한 연구는 부족한 실정이다. 한편 컴포넌트의 개발에 대한 깊이 있는 최근 연구가 진행되고 있어[4] 이 부분에 대한 보강이 이루어지고 있다. 본 장에서는 언급한 문제점과 더불어 컴포넌트 환경 전반에 걸쳐 CBSD 활성화의 장애요소로는 어떠한 것들이 있는지 살펴보고자 한다. 다음은 대표적인 항목들이다.

- **아키텍처 불일치(Architectural Mismatch) 문제[11]** : 소프트웨어 아키텍처는 소프트웨어 시스템을 구성하는 컴포넌트들의 구조, 상호 관계, 설계 및 변경을 가이드하는 원칙 등을 나타내는 시스템에 대한 상위레벨의 추상화이다[3, 10, 13, 25, 26]. 일반적으로 컴포넌트의 생산자와 소비자는 전혀 다른 조직체이기 때문에 서로 상이한 아키텍처를 기반으로 컴포넌트와 컴포넌트 기반 시스템을 구축하고 있다[12]. 이러한 아키텍처 불일치를 간과하고 단지 컴포넌트의 기능적 적합성으로만 선택하는 경우, 통합 구축되는 전체 시스템은 올바르게 작동되지 않는다.

  한편 이적인 CBSD를 지향하기 위하여 컴포넌트는 특정 아키텍처에 무관하도록 구축되어야 한다[23]는 의견이 지배적이었으나, D. Perry는 컴포넌트가 아키텍처에 완벽하게 독립적일 수 없음을 지적하였다[19]. 이러한 관점을 반영하듯, 컴포넌트를 명세 시 단순히 인터페이스만을 기술하지 않고 아키텍처 수준의 고려사항을 추가하여 보완



(그림 1) 아키텍처 불일치 문제

한 기법[16]이 지속적으로 제안되고 있다.

(그림 1)의 좌측에는 컴포넌트 개발 시 고려된 아키텍처를 도식화한 것인데, 중앙에 위치한 컴포넌트-B가 아키텍처 상의 다른 컴포넌트(A, C, D)들과 데이터 혹은 제어를 주고받는 시스템을 염두하고 구축되었음을 알 수 있다. 그러나 컴포넌트의 사용측이 이러한 아키텍처 상의 상호연관관계를 고려치 않고 기능적인 적합성만으로 컴포넌트를 도입함으로써 아키텍처 불일치가 발생하게 됨을 나타내고 있다.

- **컴포넌트 개발이 상향식으로만 이루어지는 문제** : 현재 컴포넌트는, 큰 그림(아키텍처)이 없이 생산자들에 의해 이미 형성된 시장을 중심으로 '필요'와 '이윤'을 위한 상향식(bottom-up)으로 도출·개발되고 있다. 이러한 상향식 개발은, 미숙한 시장이나 생산자의 입장에서 이윤이 없다고 판단되는 영역에 대해서는 컴포넌트가 생산되지 않는 컴포넌트의 사각지대를 조장할 여지가 충분하여 '컴포넌트의 부익부 빈익빈' 현상을 초래하게 된다. 컴포넌트의 영역별로 편중된 개발양상은 컴포넌트 생산자와 소비자 모두 중복투자를 야기한다. 이윤이 높은 영역에서는 생산자간의 과다한 출혈경쟁이 예상된다. 물론 경쟁으로 인한 품질향상을 꾀할 수는 있으나 생산자들간의 중복투자가 필연적이다. 주목할 점은, 동시에 소비자는 이와 전혀 다른 각도에서 중복투자가 발생한다는 것이다. 즉, 컴포넌트가 풍부한 영역에서는 폭 넓은 선택을 꾀할 수 있으나 컴포넌트가 전무한 영역에서는 해당 컴포넌트나 해당 기능이 요구되는 모든 소비자들이 독자적으로 개발해야 하는 중복투자가 발생하기 때문이다[27].

- **컴포넌트 탐색, 선택에 드는 막대한 시간과 노력의 낭비 문제** : J. Kontio의 논문에서 언급된 "Long Time Selection[14, 15]"문제이다. 아무리 컴포넌트가 다수의 생산자들로부터 개발된다고 하더라도 실제로 원하는 기능을 수행하는 컴포넌트가 존재하는지, 실존한다면 동일 기능을 만족하는 후보 솔루션이 더 있는지, 후보 솔루션 중에 어느 컴포넌트를 선택할 것인지 등을 파악해야 하기 때문이다. 이렇게 탐색과 선택에 많은 노력과 시간이 필요하고 그 과정이 매우 어려운 경우에는 컴포넌트에 기반한 개발이 불가능해 진다.

- **컴포넌트의 유지보수 문제** : 컴포넌트 환경에서 시스템의 수명주기는 다음과 같이 정의될 수 있다. 컴포넌트에 기반한 시스템을 S 소프트웨어 컴포넌트를 C라 하고, S는 N개의 컴포넌트로 구성되어 있다고 하자. 또한 컴포넌트의 통합이나 자체 개발을 목적으로 한 코드를 user code라 한다. 이 때, S : = {C_1, C_2, ···, C_N, *user code*}이며 함수 LT(x)는 소프트웨어 x의 수명주기(Life Time)를 의미한다.

$$LT(S) = \min\{LT(C1), LT(C2), \cdots, LT(CN), LT(usercode)\}$$

즉, 통합 시스템의 수명주기는 구성 컴포넌트의 수명주기 중에서 최소 값을 갖는다. 따라서 전술한 몇 가지 문제나 위험요소(risk)를 감안하여 컴포넌트를 도입한다고 하더라도 컴포넌트를 사용하여 구축한 시스템의 수명주기 동안 도입한 모든 컴포넌트에 대한 지속적인 유지보수 지원이 보장되지 않는 경우에는 전체 시스템의 수명이 중단되기 때문에, 컴포넌트의 유지보수 문제는 결정적 중요성을 갖는다. 더구나 컴포넌트가 상용이나 비상용이냐를 떠나 다수의 컴포넌트가 보편적으로 소스코드를 제공하지 않는 블랙박스(black-box) 형태를 띄는 경우가 대부분이기 때문에, 컴포넌트 생산자로부터 유지보수 중단이 일어났을 때 대처가 불가능하게 된다. 이 문제는 특히 영세 컴포넌트 생산자로부터 도입한 컴포넌트에서 심각하게 고려되어 진다.

이러한 장애요소들을 극복하기 위해서는 보다 발전적 기술대안(아키텍팅 기술, 컴포넌트 정의 및 검색 기술 등)뿐만이 아니라 새로운 개념 및 정책적 대안 등 비기술적 대안들도 마련되어야 할 것이다. 더구나 현재의 CBSD 환경에서 보면 언급한 장애요소들을 컴포넌트 개발자(또는 생산자) 측면에서 풀거나 컴포넌트 이용자(또는 소비자) 측면에서 풀어야 하는데, 이는 양측 모두에게 막대한 부담이 되기 때문에 사실상 불가능한 상황이다. 따라서 본 논문에서는 이러한 다양하고 어려운 장애요소들을 생산과정이나 소비과정이 아닌 유통과정을 합리적으로 지능화함으로써 극복할 수 있는 방안을 제시하고자 한다.

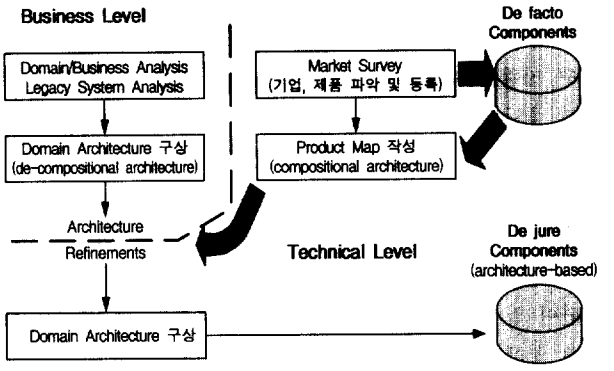
3. 부가가치 중개 개념

의미 그대로만 본다면 생산자와 소비자간을 연결시켜주는 중개를 담당한다고 할 수 있다. 중개자체가 중요하고 독립적으로 고려되어야 할 필요는 분명히 있지만, 중개의 사전적 의미만으로는 CBSD의 활성화를 꾀할 수 없다. 사전적 의미의 중개로는 2장에서 언급한 '컴포넌트 탐색, 선택에 드는 막대한 시간과 노력의 낭비문제'에 대해서만 효과를 얻을 수 있기 때문이다. 따라서 본 장에서는 CBSD 활성화를 위한 중개개념의 확장 안을 제안하고 이를 "부가가치 중개 개념"이라 하였다. 부가가치 중개 개념에서 수행하는 부가 가치적 업무는 다음과 같다.

3.1 도메인 아키텍처 지향의 컴포넌트 생산 촉진

컴포넌트의 상향식 생산은 영역별 불균형한 분포와 당장 이윤이 남지 않는 분야에 대한 투자부족으로 인한 컴포넌트 사각지대가 발생한다. 결국 생산자와 소비자 모두 중복투자로 인한 손해를 입게 된다. 따라서 컴포넌트의 생산을

상향식뿐만 아니라 하향식도 함께 접목할 필요가 있다 [8, 12, 23]. 본 논문에서 제안하는 진행과정은 (그림 2)와 같다. 세로 방향으로 시간의 흐름을 나타내고 동일 수평선상에 있는 일들은 병행하여 수행하는 것을 나타낸다.



(그림 2) 도메인 아키텍쳐 기반 컴포넌트 생산

좌측을 중심으로 먼저 설명하면, 우선 대상 도메인을 설정하고 분석한다. 분석과정에서 이미 사용되고 있는 기존 시스템(legacy system)을 분석하는 것이 함께 수행되어져야 한다. 기존시스템은 도메인과 영역의 필요한 업무흐름을 구현해서 사용했던 것이므로 이를 분석하는 것이 결과적으로는 대상 도메인과 영역을 분석하는 것으로 활용된다. 그 다음 분석결과를 바탕으로 업무 수준에서 도메인 아키텍쳐를 구상한다[4]. 이와 동시에 수행되어져야 할 업무가 있는데, (그림 2)에서 우측에 나타나있는 것들이다. 현재 시장에 어떠한 기업의 어떤 제품이 출시되어있고 유통되고 있는지 시장조사를 실시한다. 종종 하향식으로 개발한다고 하여 독자적인 아키텍쳐를 구상한 후에, 아키텍쳐에 기술된 부분에 대응되는 컴포넌트를 찾는 방식으로 진행하는 경우가 많다 [6]. 이 경우, 실제 유사한 컴포넌트가 있더라도 적용하지 못하고 새롭게 개발할 가능성이 매우 크다. 따라서 컴포넌트의 재사용성을 증대시키기 위해서는 현재 시장에 존재하는 컴포넌트를 식별, 분류하는 작업을 수행하여야 한다. 그 후 식별된 컴포넌트들을 사용하여 도메인 내에서 어떠한 응용들을 구축할 수 있을지 상향식의 아키텍쳐(프로덕트 맵)를 구상한다. 이 아키텍쳐를 물리적 컴포넌트에 종속적인 Compositional Architecture로 볼 수 있다(이 때, 자연발생적으로 구축되어 컴포넌트 시장에 존재하는 컴포넌트를 de facto 컴포넌트라 할 수 있다).

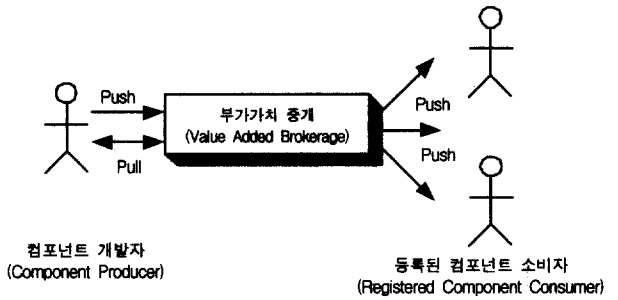
업무수준의 도메인 아키텍쳐와 기존 컴포넌트를 바탕으로 구상된 상향식 아키텍쳐(프로덕트 맵)의 구상이 완료되면, 이들을 바탕으로 실질적인 도메인 아키텍쳐를 하향식으로 구상한다. 이 단계에서는 업무분석에 따라 분석된 정보와 어떠한 컴포넌트들이 어떻게 합성되어 응용을 구축할 수 있는지에 대한 시장조사가 수행되었으므로, 하부수준을 무시 혹은 가정한 상태에서 진행하는 일방적인 하향식 방식에서 흔히 겪을 수 있는 문제점을 극복할 수 있다. 도메인에 필요한 요소들을 도출하고 도출된 요소들을 컴포넌트에 할당하고 세분화하는 과정을 반복적으로 수행하기 때문에 De-compositional Architecting으로 볼 수 있다. 물론, 프로덕트 맵의 정보를 사용하여 되도록 현존하는 식별된 컴포넌트를 재사용할 수 있도록 세분화하여야 한다. 또한 컴포넌트의 중복, 사각, 상충이 발생하지 않도록 세심한 배려를 기울여야 한다.

도메인 아키텍쳐의 구상을 완료하고 나서 컴포넌트 생산자에게 아키텍쳐에 기술된 컴포넌트들을 생산하도록 한다면, 다음과 같은 이득을 꾀할 수 있다. 생산자의 입장에서는, 시장조사 및 분석을 중개개념을 담당하는 측이 수행해 주기 때문에, 이러한 능력이 부족한 군소 개발자들을 활성화시킬 수 있다. 또한 동종의 컴포넌트를 생산하는 생산자들에게 다른 분야의 컴포넌트를 생산하도록 유도함으로써 과열경쟁을 예방할 수 있다. 소비자의 입장에서는, 생산되는 컴포넌트들이 우후죽순처럼 상향식으로 구축되지 않고 잘 정의된 하향식 아키텍쳐를 만족(confirm)하는 컴포넌트로 생산될 것이므로 컴포넌트의 고른 도메인별 분포로 인하여 컴포넌트를 선택할 수 있는 기회가 늘어나게 된다(이렇게 아키텍쳐에 기반하여 구축되는 컴포넌트를 de jure 컴포넌트라 할 수 있다). 이는 컴포넌트의 사각지대가 발생하는 영역에 대해서는 소비자마다 해당 영역의 기능을 독자적으로 구현하는데 따른 중복투자를 예방할 수 있다. 또한 기존에는 컴포넌트 단위의 재사용만을 고려했으나 아키텍쳐 수준의 대규모 재사용을 꾀할 수 있게 된다. 즉, 소비자가 구상하는 아키텍쳐가 중개 측의 도메인 아키텍쳐의 서브셋(subset)이라면, 소비자는 중개 측의 아키텍쳐에 맞게 구축된 컴포넌트(architecture-based component)를 모두 재사용할 수 있게 된다는 것이다. 이는 컴포넌트의 사용에 있어서 고질적인 문제인 long-time selection문제를 해결할 수 있을 것으로 기대된다. 물론, 생산자와 소비자에게 중개 측에서 구상한 도메인 아키텍쳐 정보가 제공 또는 공유되어야 한다. 이러한 일련의 작업들이 표준화 측면에서 이루어진다면 그 효과는 극대화 될 것이다.

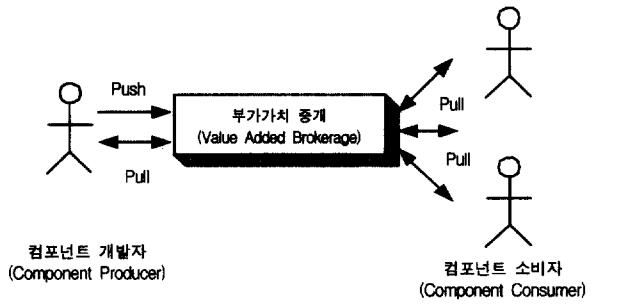
3.2 지능형 컴포넌트 검색 서비스

소프트웨어 컴포넌트 중개와 가장 부합되는 업무로, 컴포넌트 생산자와 컴포넌트에 대한 정보를 보유하고 있으면서 소비자가 요구하는 컴포넌트를 검색하여 최적의 조건들(가격, 기능, 성능, 보안, 이식성, 기업의 신뢰도, 시장 점유율, … 등)을 고려하여 제공하는 서비스이다. 이 서비스를 효과적으로 제공하기 위해서는, 컴포넌트를 식별하고 분류하는 기준이 되는 근거(컴포넌트 명세)를 정의하는 것이 핵심이다. 현재 이 부분에 대한 연구는 ISO 9126과 같은 일반적인 소프트웨어 분류 기준과 컴포넌트에 대한 명세기준으로

OMG와 기타 다른 연구기관에서 활발히 수행 중에 있다. 지능형 컴포넌트 검색 서비스는 마치 인터넷의 검색 서비스와 유사하며, 얻을 수 있는 장점은 다음과 같다. 생산자의 입장에서는, 생산한 컴포넌트의 홍보나 판매 대행을 중개 측이 수행해 줄 수 있기 때문에 다양한 홍보전략을 세울 수도 있고 컴포넌트 생산에 전념할 수 있도록 해준다. 또한 소비자의 입장에서는, 모든 소비자들이 공통적으로 컴포넌트 선택에 들이는 노력을 독립적으로 분리하여 전문화시켰기 때문에, 신속·편리하고 값싸게 최적의 컴포넌트를 사용할 수 있게 된다. 언급된 장점으로 인하여 생산자와 소비자들은 중개 측을 통해 컴포넌트를 유통하게 되므로, 자연스럽게 중개 측은 현재 어떠한 컴포넌트가 개발되고 있으며 어떠한 컴포넌트를 요구하는 지를 파악할 수 있게 된다. 파악된 정보는 도메인 아키텍처에 피드백(feedback) 되어 아키텍처를 진화(evolution) 시킬 수 있도록 한다.



(a) Push 방식



(b) Pull 방식

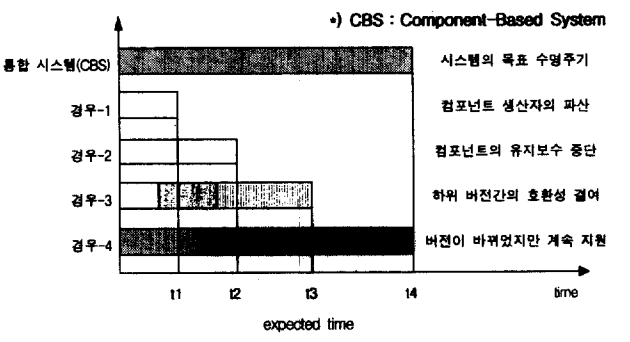
(그림 3) 컴포넌트 변경의 통보방법 (Push, Pull 방식)

전술한 컴포넌트 검색 서비스의 효과를 극대화하기 위해 다음의 두 가지 서비스를 기본적으로 제공하여야 한다. 첫째, 컴포넌트의 품질인증 획득을 촉진하는 것과 둘째, 컴포넌트 검색 결과의 현시성을 유지하는 것이다. 컴포넌트 기반 시스템은 다양한 개발조직으로부터 개발된 다수의 컴포넌트를 결합하여 구축하게 되므로 시스템을 구성하는 컴포넌트 중 어느 하나라도 요구하는 품성적 조건을 만족하지 못하면 전체 시스템 품질에 악영향을 끼치게 된다. 따라서 각각의 컴포넌트의 품질이 보장되어야 할 필요가 있다. 이를 위하여 컴포넌트 등록 시 품질인증을 유도하고 인증 된 제품에 대해서는 검색 시 우선 순위를 부여하거나 인증제품의 사용자에게 혜택을 제공하여 컴포넌트의 품질인증을 유도할 수 있다. 보통 품질인증은 공정성과 전문성을 기하기 위하여 제3의 전문 기관을 통하여 수행된다. 국내에서는 최근 ETRI에서 SQEC(Software Quality Evaluation Center)[28]를 설립하여 소프트웨어의 품질인증 서비스를 수행중에 있다.

한편, 컴포넌트 검색 서비스는 검색결과의 현시성의 유지가 필수적이다. 즉 컴포넌트의 수정이나 소유권, 라이센스의 변경 등을 지속적으로 모니터링 하여 변동사항을 반영해야만 한다. 뿐만 아니라 이러한 변동사항을 이미 컴포넌트를 도입하여 시스템을 구축 및 운영하고 있는 소비자들에게도 통보하여야 하는데, 컴포넌트의 변화는 컴포넌트에 기반한 시스템의 유지보수 및 운영관리 문제를 야기할 수 있기 때문이다. 통보방식은 크게 Push방식과 Pull방식으로 고려될 수 있으며, Push방식은 정보의 발생·변경 측이 해당 정보를 필요로 하는 측에 능동적으로 제공하는 형태를, Pull방식은 정보가 필요한 측에서 정보의 발생·변경 측에 요청하여 취득하는 형태를 의미한다(그림 3). 부가가치 중개가 컴포넌트의 변동사항을 인지하는 방법은 컴포넌트 생산자가 변경할 때 중개에 알리거나 부가가치 중개가 등록되어 있는 컴포넌트 생산자들을 모니터링 하여 감지하는 방법이 있는데, 전자의 방법은 Push방식 후자의 방법은 Pull방식이라 할 수 있다. 그러나 (그림 3)에서는 표현의 간략화를 위하여 '컴포넌트 생산자↔부가가치 중개'간의 정보교환 방식은 혼용·병기하였고, 다만 컴포넌트 소비자가 현시정보를 취득하는 관점에 준하여 Push방식과 Pull방식으로 분류하였다.

3.3 화이트 박스 서비스 (White-Box Service)

CBSD에서 고려되어야 하는 가장 중요한 문제점은 컴포넌트의 쉽고 경제적인 도입보다 컴포넌트를 도입한 후의 상황인데, 컴포넌트를 도입한 시스템의 유지보수 문제[24]가 그것이다. (그림 4)는 컴포넌트를 도입하여 구축한 시스템과 컴포넌트들 간의 수명주기 관계를 도식화한 것이다.



(그림 4) 컴포넌트 기반 시스템(CBS)과 컴포넌트간의 수명주기 관계

경우 1. 컴포넌트 생산자의 파산으로 인하여 유지보수가 중단되는 경우를 도식화 한 것이다. 주로 군소 컴포넌트 생산자로부터 컴포넌트를 도입한 시스템에서 발견될 수 있는데, 이는 최악의 상황으로 어떠한 형태로든 유지보수를 받을 수 없는 상태다. 결국 시스템 운영자는 유지보수가 중단된 컴포넌트를 동종의 다른 컴포넌트로 대체하던지 컴포넌트의 기능을 직접 구현함으로써 이 문제를 극복해야 한다. 대체가 불가능한 경우 전체 시스템의 운영이 중단된다.

경우 2. 생산자의 경영방침이나 기술적인 요인으로 컴포넌트에 대한 유지보수가 중단되는 경우다. 유지보수가 중단된 컴포넌트로부터 문제점이 발생하게 되면, 유지보수가 불가능하므로 시스템 생명주기를 단축시키게 된다. 경우 1과의 큰 차이점은, 생산자가 존속해 있어서 일정기간은 야기된 문제들은 해결의 여지가 있다는 것이다.

경우 3. 일반적인 소프트웨어는 하위버전과의 호환성을 유지하지만, 제품의 혁신을 꾀하거나 호환성을 지속적으로 유지하더라도 임의의 시점에는 모든 하위버전과의 호환이 불가능한 경우가 발생된다. 이 때, 보통 상위버전으로의 변경을 권고하는 경우가 많은데, 시스템이 안정화 상태에 놓인 경우에는 새로운 버전이 나왔다고 하더라도 교체를 꺼리는 경우가 많다. 즉, 새로운 버전의 컴포넌트에 의한 아키텍쳐 불일치 문제는 구축된 시스템의 안정성을 침해할 우려가 크기 때문이다. 더구나 버전간의 호환성이 깨지게 되면, 대개 하위버전에 대한 유지보수를 중단하게 되므로 경우 2의 상황으로 전이될 수 있다. 그러나 경우 2와는 달리 유지보수는 보장되지만, 상위버전으로 전환하면서 기존 시스템의 생명주기가 단축될 수 있다는 점이 다르다.

경우 4. 가장 이상적인 경우로 컴포넌트의 버전이 바뀜에도 불구하고 여전히 이전 버전과의 호환성을 유지한다. 뿐만 아니라 지속적인 지원으로 인하여 유지보수가 원활하게 이루어진다.

따라서, 컴포넌트 생산자의 전략적 문제나 기타 여건에 의하여 컴포넌트에 대한 지원을 지속할 수 없는 경우라도 해당 컴포넌트를 도입한 소비자들을 보호하기 위하여 유지보수 서비스의 계속성 유지가 필수적이라 할 수 있다. 중개측에서 유지보수에 문제가 생기는 컴포넌트의 소스코드와 소스코드 사용에 대한 권한을 양도받아 이를 보유하고 있다가 추후 소비자들의 요청에 따라 중개 측이 유지보수 서비스를 해주거나 또는 부득이한 경우 소스코드를 제공하여 소비자 스스로 유지보수를 감당하도록 함으로써 유지보수의 계속성을 보장하도록 한다. 상용 및 비상용 컴포넌트들이 대다수 블랙 박스 형태로 제공되기 때문에 발생하는 문제를 해결하기 위해 제안하는 서비스인 만큼, 대조적인 용어를 사용하여 "화이트 박스 서비스"라 명명하였다. 만약 누군가 해당 컴포넌트의 상품적 가치를 인정하고 소유권을 사게 된다면, 굳이 중개 측이 개입할 필요는 없지만 누가 소유하게 되는 지에 대한 정보는 계속 파악해서 3.2의 지능형 컴포넌트 검색 서비스에 반영하여야 한다. 화이트 박스 서비스의 개념은 기존에 없는 개념이므로 이를 현실화시키기 위해서는 다음과 같은 몇 가지 방안을 고려해 볼 수 있다. 중요한 것은, 언급될 방안들이 상호 경쟁적 관계가 아니라 상호 보완적인 관계라는 점이다.

- **소비자들이 많이 사용하는 컴포넌트를 중개 측에서 구입 및 판매**

  지능형 컴포넌트 검색 서비스를 통해서 중개 측은 소비자들이 임의의 컴포넌트를 얼마만큼 원하고 또한 사용하는지 알아 낼 수 있다. 따라서 생산자의 변화가 발생하는 경우, 이를 감지하고 소비자가 많은 컴포넌트의 경우 중개 측에서 구입한다. 일개 개인이나 조직이 구입하게 되면, 불특정 다수에 재사용(유지보수의 계속성)의 기회를 박탈하게 된다. 뿐만 아니라 구입 시에 드는 비용은, 다수의 고객이 확보되어 있는 중개 측보다 개인이 구입하는 비용부담이 훨씬 크다. 이 방법은 가장 자본주의 논리에도 부합되며 실현가능성이 높은 방법이다. 다만, 생산자의 변화를 중개 측에서 능동적으로 감지하던지 생산자 측에서 중개 측에 변화를 통보하는 체계가 확립되어야 한다. 또한 소비자의 요구에 합당하는 검색결과를 제시하는 것으로 끝나는 것이 아니라, 소비자의 요구성향이나 도입한 컴포넌트들에 대한 정보를 수집하여 통계를 내는 작업이 선행되어야 한다. 이 방법은 소비자뿐만 아니라 생산자에게도 이득을 주게 되는데, 파산이나 새로운 버전의 출시 등과 함께 사장(死藏)되는 기존의 컴포넌트를 중개 측에서 타당한 금액으로 인수하게 됨으로써 컴포넌트 공급시장을 활성화시킬 수 있기 때문이다.

- **법·제도적인 장치를 마련**

  중개 개념이 도출되어 독립적으로 존재하고 언급한 서비스들을 제공한다고 하더라도, 생산자와 소비자의 참여를 이루지 못한다면 무용지물로 전락하게 된다. 소비자는 전술한 업무들을 사용하는 것이 실질적으로 이득이 되기 때문에 참여를 유도하는 데에는 큰 어려움이 없을 것으로 판단된다. 그러나 컴포넌트 생산자들의 경우는 단기적으로 볼 때, 직접적인 이득이 그리 많지 않기 때문에 생산자가 중개 측에 생산한 컴포넌트의 정보를 등록하고 갱신하는 절차를 수행하는데 소극적일 소지가 있다. 따라서 이러한 부가적 절차 수행을 보다 적극적으로 유도하는 법·제도적인 장치가 필요하다. 예를 들자면, 중개 측에 협조를 약속하고 세금 감면이나 인허가에 있어서의 혜택 그리고 특정 성격을 띄는 지원자금

제공 등이다. 여기서의 협조란, 생산자가 생산한 컴포넌트 자체와 생산자 조직 자체의 변화를 주기적으로 중개 측에 통보하는 것과 향후 해당 컴포넌트에 대하여 더 이상의 지원을 보장할 수 없게 되었을 때는 컴포넌트 가격의 일정 비율로 중개 측에 양도한다는 식의 계약 서명 등을 들 수 있겠다. 국내적 특수성을 감안하면, 등록된 컴포넌트의 불법 복제와 같은 지적재산권, 라이선스 문제 등을 보다 엄격하게 다룬다는 '생산자 보호' 조항을 두는 것도 필요하다.

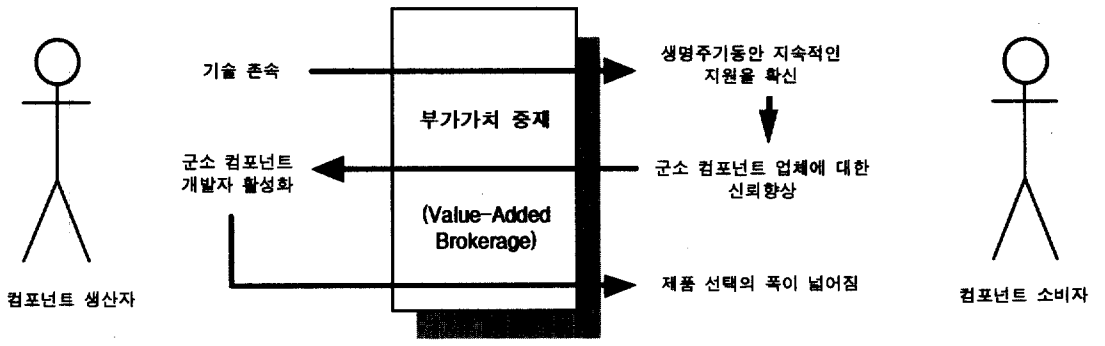
- **상 도덕(Business Moral) 조성**

  가장 근본적인 문제이며, 지속적인 노력이 필요한 부분이다. 애써 개발한 기술이 사장되는 바에는 존속시키기 위하여 무상으로 중개 측에 기증하는 것을 그 첫 번째로 들 수 있다. 국내외의 몇몇 소프트웨어 기업들은 의도적으로 소스코드를 공개하는 경우가 있으며, 특히 GNU 선언문을 따르는 프로그래머들은 애써 개발한 응용의 소스코드를 아무런 대가 없이 공개한다. 물론 종종 상업적인 의도를 그 저변에 두고 실행하기도 하지만, 이는 매우 바람직하다 하겠다. 특히 기업이 파산하면서 생산했던 컴포넌트를 휴지조각으로 만들 수도 있는 상황에서 이를 중개 측에 기증하는 것은 상도덕이 어느 정도 성숙되지 않고서는 기대할 수 없다. 두 번째로는 소비자가 컴포넌트를 적법한 절차로 사용하는 상도덕의 조성이 필요하다고 하겠다.

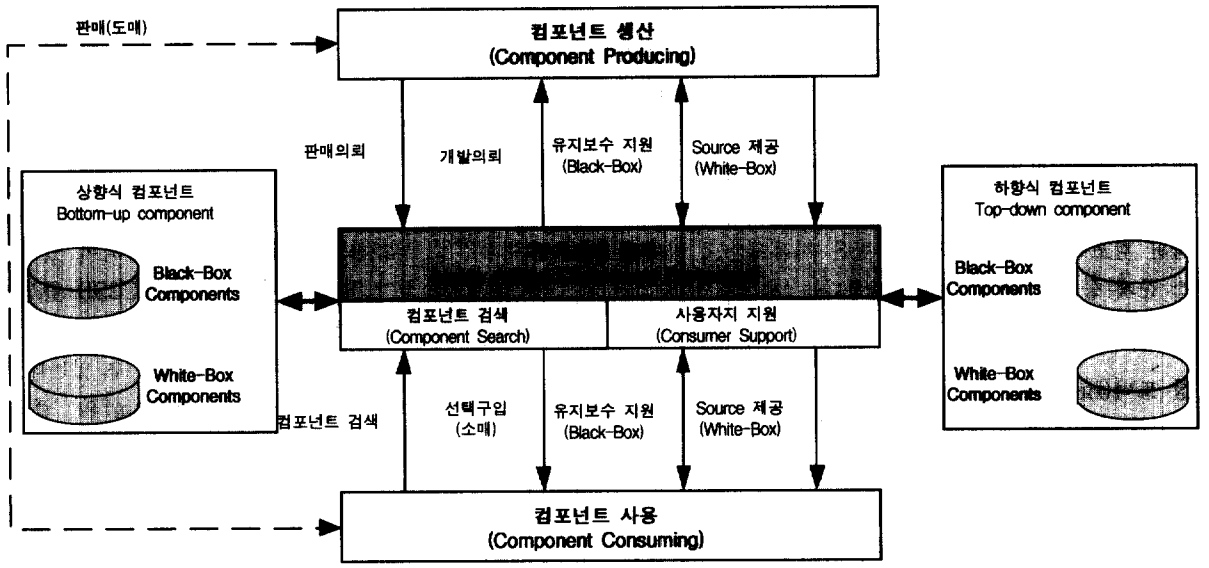
이제 전술한 화이트 박스 서비스를 수행함으로써 얻을 수 있는 장점들을 살펴보도록 한다. 개발자의 입장에서는 애써 개발한 제품이 사장되는 것을 막을 수 있으며, 중개 측이 개발자의 존속여부·지원여부와 무관하게 지속적 유지보수를 보장하게 되므로 소비자의 컴포넌트 사용에 관한 신뢰도가 향상되는 것은 물론 군소 컴포넌트 생산자들을 활성화시킨다. 소비자의 입장에서는 시스템의 생명주기 동안에 지속적인 유지보수 지원이 보장되며, 군소 컴포넌트 활성화로 인하여 소비자가 선택할 수 있는 컴포넌트의 선택의 폭이 넓어진다. (그림 5)는 화이트 박스 서비스를 통하여 얻어지는 장점들 간에 시너지 효과가 있음을 도식화하고 있다.

4. 결 론

본 논문에서는 CBSD 환경에서 다수의 연구가 컴포넌트 소비자영역(통합자)또는 생산영역에 편중되어 있어 야기되는 문제점들을 지적하고 해결방안으로 CBSD 활성화를 위한 부가가치 중개 개념을 제안하였다. 현재 컴포넌트 중개 서비스가 제공되고 있는 사례를 살펴볼 수 있는데, 대표적으로 '컴포넌트소스(componentsource.com)'와 '플래시라인(flashline.com)'의 서비스를 꼽을 수 있다. 이들은 컴포넌트의 기능과 적용분야에 따라 분류를 제공하며, 컴포넌트 검색기능을 통해 원하는 컴포넌트를 찾을 수 있도록 한다. 그러나 이들 서비스는 본 논문에서 제안한 부가가치 중개 개념의 세 가지 업무중에서 두 번째 업무에만 초점이 맞추어져 있고 도출되는 결과가 다소 제한적이라 평가할 수 있다. 이들 서비스와 제안한 개념과의 차이점을 다음과 같이 세 가지로 정리할 수 있다. 첫째, 기능 및 적용분야에 대한 분류를 제공하고는 있으나 도메인에 대한 아키텍처를 제공하는 수준은 아니라는 것이다. 기존에 존재하는 컴포넌트를 취합하여 분류하고 있으나 취합된 컴포넌트로부터 구축 가능한 도메인의 아키텍처 제공은 물론 상향식 아키텍처의 제공이 고려되고 있지 못하다. 둘째, 검색서비스를 제공하고는 있으나 주로 기능적 측면의 검색만을 지원하고 있어 아키텍처 불일치를 해소할 수 없다. 한편, 검색결과가 라이선스, 구입비용, 플랫폼 등 다양하게 제시되고 있지만 컴포넌트의 아키텍처에 기반한 제약조건을 제시하지는 않고 있다. 즉, 종속성을 갖는 다른 컴포넌트나 필요한 외적 기능 등의 제시가 필요하다. 셋째, 제공되는 서비스는 주로 유통과 공급에 치중되어 있어 핵심적 사안인 유지보수에 대한 부분의 관리 및 정보제공은 전혀 이루어지고 있지 못하다. 물론, 이 부분에 있어서는 기술적 측면보다는 비기술적 측면의 문제가 우선하지만, 정보기술 측면에서 정보제공 노력이 요구된다 할 수 있다. 현재까지 파악된 바로는 아직



(그림 5) 화이트 박스 서비스를 통한 시너지 효과



(그림 6) 부가가치 중개 개념

제안된 개념을 지원하는 서비스는 없다하겠다.

최근 컴포넌트 육성을 위한 국가 시책으로 인하여 상당수의 컴포넌트가 개발되고 있는데, 중요한 점은 각 컴포넌트들이 서로 다른 조직으로부터 독립적으로 개발되고 있기 때문에 컴포넌트의 중복, 상충, 사각이 발생할 여지가 많다는 것이다. 이 부분에 대해서는 다행히 현재 '한국 소프트웨어 컴포넌트 컨소시엄(KCSC)' 중심의 활발한 활동으로 다소 해결책을 모색하고 있는 듯하다. 하지만, 현재 다수의 컴포넌트들은 연구소에서 개발되고 있는 실정이어서 제작이 완료된 컴포넌트의 판매는 물론 지속적인 유지보수를 보장할 주체도 불명확한 상황이다. 즉, 장기적인 컴포넌트 시장의 활성화에 대한 고려는 부족하다고 할 수 있다. 결국 이러한 위험요소(risk)는 막대한 투자를 통해 제작된 컴포넌트가 사용자들로부터 외면을 받고 사장될 뿐만 아니라 나아가 컴포넌트 시장 자체가 축소 또는 소멸되어 컴포넌트 산업이 존폐의 위기까지 치달을 수 있다.

제안된 부가가치 중개 개념을 도입하였을 때 얻을 수 있는 장점은 다음과 같다. 도메인 아키텍처 지향의 컴포넌트 생산 촉진으로 컴포넌트의 중복, 상충, 사각을 예방할 수 있으며, 대규모 재사용을 꾀할 수 있는 것은 물론 지능형 컴포넌트 검색을 통하여 컴포넌트의 유통을 촉진하게 된다. 또한 블랙 박스형태의 컴포넌트의 경우에는 사용자와 생산자를 연결하여 유지보수를 지원할 수 있도록 하고 화이트 박스 서비스를 통해서는 도입한 컴포넌트에 대해 지속적인 유지보수를 보장할 수 있으므로 안전한 컴포넌트 도입환경을 조성할 수 있을 것으로 기대된다 (그림 6). 향후 연구과제로는 기술적 측면과 기술외적 측면을 고려해 볼 수 있는데, 기술 측면으로는 부가가치 중개 개념을 지원할 수 있는 컴포넌트 생산자, 사용자, 중개 개발 프로세스, 아키텍처를 고려한 컴포넌트 명세, 컴포넌트 기반 아키텍팅, 컴포넌트 평가(evaluation), 컴포넌트 검색, 아키텍처 명세 기술 등에 관한 구체적인 연구가 필요하다. 한편 기술외적 측면으로는 제안된 개념을 지원할 수 있는 정책과 모랄(moral)의 개발 및 구현에 관한 연구가 요구된다.

참 고 문 헌

[1] M. Aoyama, "New Age of Software Development : How Component-Based Software Engineering Chages the Way of Software Development," in Proc. of the First International Workshop on Component-Based Software Engineering-In Cooperation with the 20th ICSE, Kyoto, Apr. 1998.

[2] Andersen Consulting, "Understanding Components," http : //www.ac.com/services/eagle/eagle_thought1.html, 1998.

[3] L. Bass, P. C. Clements, and R. Kazman, "Software Architecture in Practice," Addison-Wesley, 1998.

[4] P. Bengtsson and J. Bosch, "Scenario-based Software Architecture Reengineering," in Proc. 20th ICSE, IEEE, Jun. 1998.

[5] F. Brosard, D. Bryan, W. Kozaczynski, E. S. Liongorari, J. Q. Ning, A. Olafsson, and J. W. Wetterstrand, "Toward Software Plug-and-Play," in Proc. of the 1997 Symposium on Software Reusability, pp.12-29, 1997.

[6] A. W. Brown and K. C. Wallnau, "Engineering of Component-Based Systems," Component-Based Software Engineering, IEEE CS Press, pp.7-15, 1996.

[7] P. C. Clements, "From Subroutines to Subsystems : Component-Based Software Development," Component-Based Software Engineering, IEEE CS Press, pp.3-6, 1996.

[8] P. C. Clements and L. M. Northrop, "Software Architecture : An Executive Overview," Technical Report (CMU/SEI-96-TR-003), Pittsburgh, PA : Software Engineering Institute, Carnegie Mellon University, Feb. 1996.

[9] S. Cohen, B. Gallagher, M. Fisher, L. Jones, R. Krut, L. Northrop, W. O'Brien, D. Smith, and A. Soule, "Third DoD Product Line Practice Workshop Report," Technical Report(CMU/SEI-2000-TR-024), Pittsburgh, PA : Software Engineering Institute, Carnegie Mellon University, Jul. 2000.

[10] C. Gacek, A. Abd-Allah, B. Clark, and B. Boehm, "On the Definition of Software System Architecture," in Proc. of the First International Workshop on Architectures for Software Systems - In Cooperation with the 17th ICSE, D. Garlan (ed.), Seattle WA, pp.85-95, April, 1995.

[11] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural Mismatch (Why it' s hard to build systems out of existing parts)," in Proc. of the 17th ICSE, Seattle, pp.179-185, April, 1995.

[12] W. M. Gentleman, "Architecture for Software Construction by Unrelated Developers," In Software Architecture. TC2 First Working IFIP Conference on Software Architecture (WICSA1), pp.423-435, Feb. 1999.

[13] IEEE Architecture Working Group, "IEEE P1471 Recommended Practice for Architectural Description, Draft 5.2," Dec. 1999.

[14] J. Kontio, G. Caldiera and V. R. Basili, "Defining Factors, Goals and Criteria for Reusable Component Evaluation," in Proc. of CASCON '96, Toronto, Canada, pp.12-14, Nov. 1996.

[15] J. Kontio, S-F Chen, K. Limperos, R. Tesoriero, G. Caldiera and M. Deutsch, "A COTS Selection Method and Experiences of Its Use," in Proc. 20th Software Engineering Workshop, NASA Software Engineering Laboratory, Greenbelt, MD, 1995.

[16] B. Meyer, "Applying Design by Contract," IEEE Computer, Vol.25, No.10, pp.40-51, Oct. 1992.

[17] J. Q. Ning, "A Component-Based Software Development Model," In Proceedings of 20th COMPSAC '96, Seoul, Korea, pp.389-394, Aug. 1996.

[18] D. E. Perry A. L. Wolf, "Foundations for the Study of Software Architecture," ACM SIGSOFT Software Engineering Notes, Vol.17, No.4, pp.40-52, Oct. 1992.

[19] D. E. Perry, "System Compositions and Shared Dependencies," 6th Workshop on Software Configuration Management, ICSE 18, Berlin Germany, Mar. 1996.

[20] J. Sametinger, "Software Engineering with Reusable Components," Springer-Verlag, 1997.

[21] D. Sprott and L. Wilkes, "Component-Based Development : Application Delivery and Integration Using Componentised Software," Butler Group, Sept. 1998.

[22] V. Tran, "Component-based Integrated Systems Development : A Model for the Emeging Procurement-centric Approach to Software Development," in Proc. of 22nd COMPSAC, Vienna, Austria, pp.128-135, Aug. 1998.

[23] M. R. Vidger and J. Dean, "An Architectural Approach to Building Systems from COTS Software Components," Technical Report, National Research Council, Canada, http : //wwwsel.iit.nrc.ca/abstracts/NRC40221.abs. 1997.

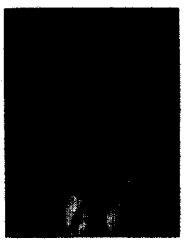
[24] J. Voas, "Maintaining Component-Based Systems," IEEE Software, Vol.15, No.4, pp.22-26, July/August, 1998.

[25] 국토개발연구원, "국가 ITS사업의 핵심공유 기반기술 연구 (최종보고서)", 1997.

[26] 최미숙, 심우곤, 백인섭, "아키텍처에 기반한 소프트웨어 프로세스 모델 연구", 정보과학회지 춘계학술대회, pp.593-595, 1999.

[27] 심우곤, 백인섭, "컴포넌트 재사용성 제고를위한 포괄적 프로세스 모델", 제2회 한국 소프트웨어공학 학술대회논문집, 제2권 제1호, pp.211-220, 2000.

[28] 한국소프트웨어진흥원, S/W품질인증과 경쟁력 강화 전략 워크샵, (ETRI S/W 시험센터 : SQEC-http : //sqec.etri.re.kr), Jul. 2001.



심 우 곤

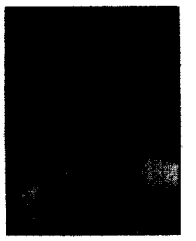
e-mail : startear@infoeng.ajou.ac.kr

1998년 아주대학교 컴퓨터공학 학사

2000년 아주대학교 컴퓨터공학 석사

2000년~현재 아주대학교 정보통신전문대학원 박사과정

관심분야 : 소프트웨어공학, CBSE, 소프트웨어 아키텍처, 지능형교통시스템(ITS) 등



백 인 섭

e-mail : ispaik@madang.ajou.ac.kr

1969년 서울대학교 전기공학 학사

1975년 한국과학원 전산학과 전산학 석사

1981년 프랑스 그레노블대학 전산학 박사

1970년~1976년 한국 과학기술 연구소 선임연구원

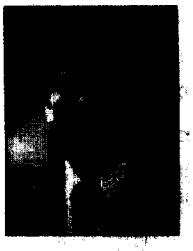
1981년~1983년 프랑스 국립통신연구소 연구원

1983년~1988년 한국 데이터 통신 연구소장

1988년~1992년 한국 전산원 연구위원

1992년~현재 아주대학교 정보 및 컴퓨터공학부 교수

관심분야 : 소프트웨어공학, 지능형교통시스템(ITS), 데이터베이스, 정보기술 아키텍처 등



이 정 태

e-mail : jungtae@madang.ajou.ac.kr

1979년 서울대학교 농과대학 학사

1981년 서울대학교 자연과학대학 계산학 이학석사

1988년 서울대학교 자연과학대학 계산학 이학박사

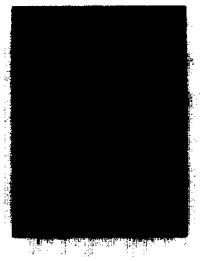
1982년~1983년 울산공과대학교 전산학과 전임강사

1983년~1988년 아주대학교 공과대학 전자계산학과 전임강사

1988년~1997년 아주대학교 정보통신대학 조교수

1997년~현재 아주대학교 정보통신대학 부교수

관심분야 : 객체지향시스템, 프로그래밍 언어, 컴포넌트 기술 등



류 기 열

e-mail : kryu@madang.ajou.ac.kr

1985년 서울대학교 컴퓨터공학 학사

1987년 한국과학기술원 전산학 석사

1992년 한국과학기술원 전산학 박사

1992년~1993년 한국과학기술연구원 연구원

1993년~1994년 동경대학 객원 연구원

1994년~현재 아주대학교 정보 및 컴퓨터공학부 부교수

관심분야 : 객체지향시스템, 프로그래밍 언어, 컴포넌트 기술, 메시징 시스템 등