클라이언트-서버 구조와 상황 모델에 대한 재고

최원영 * · 전성현 * * · 이 춘 열 * * *

Rethinking Client/Server Architecture and Contingency Model

Won Young Choi, Sung-hyun Juhn, Choon Yeul Lee

요 약

Edelstein[1994]이 제시한 클라이언트/서버 아키텍쳐가 표준으로 자리잡은 가운데 최근 국내에서는 김영걸, 박영면[1995]과 문태수,정윤[1997]교수가 클라이언트/서버 채택을 위한 상황모델을 제시하였다. 그런데 최근 정보기술의 급속한 발전 상황(특히, 데이터베이스 관리 시스템 분야, 클라이언트/서버 기술 등)은 현재 클라이언트/서버 아키텍쳐와 클라이언트/서버 채택을 위한 상황모델을 설명하는 데 있어서 약간의 어려움을 주고 있다. 따라서 본 논문에서는 클라이언트/서버 아키텍쳐의 보완을 제안하고, 현재까지의 상황 모델에 대한 해석에 있어서의 오류를 지적하여 새로운 모형과 그 해석을 제시하고자 한다. 그 전에 현재까지의 구조와 모델을 고찰 및 비판할 것이다.

Key words : client/server , contingency model, database

제 1 장. 서 론

최근의 객체-관계형 데이터 베이스 관리 시스템은 데이터 베이스 내에 데이터 뿐만 아니라 서버용 프로그램도 스토어드 프로시져(stored procedure), 스토어드 펑션(stored function), 테이블 트리거(table trigger)라는 이름하에 자체의 데이터 딕셔너리(data dictionary)를 이용하여 저장하고 있다. 그리하여 비즈니스 로직(business logic)과 데이터 로직(data logic) 전부를 데이터 베이스 내에서 보유하고 있는 것이다. 따라서 기존의 아키텍쳐는 그대로 유용하되 표기 방법에 있어서는 다소의 수정이 있어야 된다고 생각한다.

또한 현재의 상황 모델은 데이터의 양과 업무의 복잡성이라는 2 개의 변수에 의한 2 차원 모델로 표현되고 있으나 정작 중요한 결정요소는 서버라는 하드웨어의 제한된 능력과 데이터베이스 관리 시스템의 최대 허용 사용자수에 기인한 사용자수 라고 보아진다. 따라서 데이터의 양, 업무의 복잡성, 사용자수에 의한 3 차원 모델로 표현될 수 있어야 한다고 본다.

또 하나, 최근에 소개된 제품의 경우 100% 서버에서 수행되는 클라이언트/서버 시스템 [Citrix,1999] 이 출시되고 있다. 하지만 응용 로직이 클라이언트에 존재하지 않는다는 것이지 표현서비

* 국민대학교 정보관리학과 박사과정

* * 국민대학교 정보관리학과 교수

* * * 국민대학교 정보관리학과 부교수

스는 클라이언트에 존재하기 때문에 이것 역시 클라이언트-서버 아키텍쳐의 하나로 보아야 할 것이다. 컴퓨팅 파워의 분산이라는 대명제하에 출발했던 클라이언트-서버는 관리상의 어려움 때문에 거센 도전을 받고 있는 것이다.

제 2 장. 선행 연구에 대한 고찰

2-1. Edelstein (1994)에 대한 고찰

Edelstein(1994)은 클라이언트-서버 구조를 파일서버, 원격데이터, 원격표현, 로직분할, 3 단계구조와 같이 5 가지 유형으로 구분하였다. 파일서버 구조는 파일 서비스만 한 노드에서 처리하는 경우를 말하며, 원격데이터 구조는 한 노드에서 데이터 서비스와 파일 서비스 역할을 하는 경우이고, 원격표현 구조는 표현 로직과 표현 서비스가 한 노드에 있는 경우를 말하며, 로직분할 구조는 업무 로직과 데이터 로직이 두 노드에 있는 경우를 말한다. 그리고 마지막으로 3 단계구조는 서버를 두 노드로 분리하여 데이터 서버와 응용 서버로 분리하는 경우를 말한다.

2-2. 김영길,박영면(1995)에 대한 고찰

김&박(1995) 은 기업의 정보 처리적 특성에 영향을 줄 수 있는 환경 요소를 11 가지로 구분했으며 각 요소를 정보 처리에 기여하는 용도에 따라 클라이언트-서버 아키텍쳐 선정에 직접 영향을 주는 결정 요소(업무 처리 형태, 업무 단위별 상호 참조성, CPU 의존도, 데이터 유형, 트랜잭션 수, 사용자 상태, 필드의 수)와 실제 사용시 고려해야 할 구현 요소(현 정보기술의 유용성, 자원 지원 능력, 개발자의 능력, 개발 강도)로 구분하였다. 그리고 7 가지 결정 요소는 업무의 복잡도(업무 처리 형태, 업무 단위별 상호 참조성, CPU 의존도)와 데이터 량(데이터 유형, 트랜잭션 수, 사용자 상태, 필드의 수)이라는 2 가지 기준으로 구분하였다. 그리고, 업무의 복잡도와 데이터량의 고-저(high-low)에 따라 4 가지의 업무처리특성 유형을 구상하였다. 그래서 이 4 가지의 상황적인 업무처리 유형에 따라 클라이언트-서버 구조를 설계할 수 있다고 주장하였다.

2-3. 문태수,정윤(1997)에 대한 고찰

문&정(1997) 은 업무 처리의 복잡도와 기존 정보시스템 규모에 따라 2 가지 축의 클라이언트-서버 아키텍쳐를 선택하는 상황 모형을 제시하였다. 그리고, 업무 처리의 복잡도와 기존 정보시스템 규모의 고-저(high-low)에 따라 4 가지의 업무처리특성 유형을 구상하였다. 그래서 이 4 가지의 상황적인 업무처리 유형에 따라 클라이언트-서버 구조를 설계할 수 있다고 주장하였다.

제 3 장. 체계적인 클라이언트-서버 구조와 상황 모델

3-1. 체계적인 클라이언트-서버 구조

최근의 객체-관계형 데이터 베이스 관리 시스템은 데이터 베이스 내에 데이터뿐만 아니라 서버용 프로그램도 스토어드 프로시져(stored procedure), 스토어드 펑션(stored function), 테이블 트리거(table trigger)라는 이름하에 자체의 데이터 딕셔너리(data dictionary)를 이용하여 저장하고 있다. 즉 비즈니스 로직(business logic)과 데이터 로직(data logic)전부를 데이터 베이스 내에서 보유하고 있는 것이다. 이것에 대한 장점은 이미 파싱(parsing)이 되어 있으므로 인한 수행 성능의 향상에 있다. 또한 모듈화가 용이하므로 소프트웨어의 재사용성에 특히 유리하다. 따라서 기존의 아키텍쳐는 그대로 유용하되 표기 방법에 있어서는 다소의 수정이 있어야 된다고 생각한다.

또 하나, 최근에 소개된 제품의 경우 100% 서

버에서 수행되는 클라이언트/서버 시스템 [Citrix,1999] 이 출시되고 있다. Citrix Systems, Inc. 에서 발표한 1999 년 White Paper "Server-based Computing"에 의하면, ICA(Independent Computing Architecture)와 클라이언트 디바이스에서 서버로 응용 처리를 옮겨주는 ICA 프로토콜인 MultiWin 이라는 기능을 제공하므로서 많은 사용자들이 동시에 서버 상에서 수행되는 응용 프로그램들을 접근 가능하게 하고 있다. 즉, 2tier, 3tier 클라이언트/서버 시스템이 100% 서버 상에서 수행되고 관리되어 진다.

Citrix 는 클라이언트 디바이스에 응용 프로그램의 인터페이스를 위한 표현 로직을 서버에서 분배하는 방법을 제공한다. 하지만 컴포넌트들이 동적으로 네트워크를 통해서 클라이언트 디바이스로 다운로드되는 네트워크 컴퓨팅 아키텍쳐(Network Computing Architecture)와는 다르다. 결국 호스트 컴퓨팅의 장점과 퍼스널 컴퓨팅의 장점을 동시에 취하고자 하는 것이다. 부가적으로 필요한 장비는 WBT(Windows Based Terminal)이라는 클라이언트 하드웨어 장비를 필요로 할 뿐이다. WBT 가 기존의 클라이언트용 장비와 다른 점은 응용 프로그램의 다운로드가 없다는 점과 클라이언트에서 응용 프로그램의 수행이 없다는 점이다.

Citrix 모델의 배경에는 무엇보다도 클라이언트/서버 컴퓨팅의 관리에 있어서의 문제점들, 예들들면, 수천대의 클라이언트에 각 업무별로 다른 또는 동일한 프로그램들이 존재하므로 인해 클라이언트 프로그램의 배포 및 관리에 있어서의 어려움과 같은 문제에 봉착하는 많은 사용자를 위한 고려가 우선되었다고 보여진다. 컴퓨팅 파워의 분산이라는 대명제를 갖고 시작한 클라이언트/서버 컴퓨팅은 그 자체의 또 다른 문제점에 대해 거센 반발에 부딪히고 있는 것이다.

새로이 제시되는 아키텍쳐는 기존의 아키텍쳐 (Edelstein,1994)에 새로이 추가되는 또 다른 하나의 아키텍쳐를 포함한다. 즉, 클라이언트에는 표현서비스만 존재하는 가상프리젠테이션(Virtual Presentation)이 추가된다. 또 하나 특기할 점은 비즈니스 로직과 데이터 로직이 별도로 존재하는 외에 데이터 서비스에 포함되어 존재하기도 한다는 것이다. 이것에 대한 표기는 ( )로서 표기하기로 한다. [주:예를 들면 (데이터로직)와 같이 하는 것이다.] 그림 1 에 가상 프리젠테이션의 개념을 나타내었다.

3-2. 체계적인 상황 모델

김&박(1995) 은 그들의 논문 pp.170~172 에서 기업의 정보 처리적 특성에 영향을 줄 수 있는 환경 요소를 11 가지로 구분했으며 각 요소를 정보처리에 기여하는 용도에 따라 클라이언트-서버 아키텍쳐 선정에 직접 영향을 주는 결정 요소(업무처리 형태, 업무 단위별 상호 참조성, CPU 의존도, 데이터 유형, 트랜잭션 수, 사용자 상태, 필드의 수)와 실제 사용시 고려해야 할 구현 요소(현 정보기술의 유용성, 자원 지원 능력, 개발자의 능력, 개발 강도)로 구분하였다. 그리고 7 가지 결정 요소는 업무의 복잡도(업무 처리 형태, 업무 단위별 상호 참조성, CPU 의존도)와 데이터 량(데이터 유형, 트랜잭션 수, 사용자 상태, 필드의 수)이라는 두 가지 기준으로 구분하였다. 한편, 클라이언트-서버 아키텍쳐 선정에 직접 영향을 주는 결정 요소 중 사용자 상태에 대해서 "사용자의 위치나 환경에 대한 요소로서 사용자 수가 많거나 원격 작업일 경우 네트워크에 흐르는 데이터 량에 따라 시스템에 영향을 주게 된다." 와 같이 정의하였다.

그러나 사용자 수와 데이터 량의 관계는 필요충분 조건이 아니다. 사용자 수가 많다면 데이터 량은 많다고 볼 수 있으나, 데이터 량이 많다고 해서 사용자 수가 많다고는 볼 수 없다. 게다가 서버라는 하드웨어와 데이터베이스 관리 시스템의 능력은 거의 절대적으로 접속 사용자 수에 의존한다. 따라서, 혼합된 의미를 내포하고 있는 사용자 수는 별도로 분리해 낸다. 한편, 사용자 수를 제외한 업

우의 복잡도와 데이터 량에 대한 논의는 pp.176~179 에서도 논의된 것처럼 업무 처리가 복잡하면서 데이터량이 적은 경우는 네트워크의 부담이 적을 것이므로 가급적이면 로직을 클라이언트로 내리고, 업무 처리는 간단하지만 데이터량이 많은 경우는 서버의 부담이 적을 것이므로 가능하면 서버에서 처리하라는 것이 타당하므로 그대로 사용하기로 한다. 그리하여, 업무의 복잡도(업무 처리 형태, 업무 단위별 상호 참조성, CPU 의존도), 데이터 량(데이터 유형, 트랜잭션 수, 네트워크 경유 데이터량, 필드의 수), 사용자 수(사용자수)에 의한 3 차원적인 관계를 구성하였다.

또, 김&박(1995) pp.173~174 에서 논의가 된 것 중에서 원가관리의 경우 원격데이터가 적합하다고 했으나 실상 원가관리의 경우 최종 산출물은 간단할 지언정 업무의 특성은 많은 데이터와 복잡한 업무로 구성이 되어 있다. 또, 회계관리의 경우 원격표현을 권장했으나 요즘의 회계관리는 결산수정분개와 타 시스템 자동분개 등의 개념이 도입되어 실상은 많은 데이터와 꽤 복잡한 배치처리성의 업무가 주를 이루고 있다. 김&박(1995) 의 클라이언트/서버 설계 전략 (pp.179~pp.181)대로 한다면 웬만한 기업의 클라이언트-서버 아키텍쳐는 3 단계 구조와 로직 분할 구조가 되어야만 한다. 실무적으로 2tier 와 3tier 를 구분하는 가장 명백한 분류는 사용자 수를 커버(cover)하는 서버와 데이터베이스 관리 시스템의 능력에 거의 절대적으로 달려 있다.

운&정(1997) 은 pp.257~258 에서 업무 처리의 복잡도와 기존 정보시스템 규모에 따라 2 가지 축의 클라이언트-서버 아키텍쳐를 선택하는 상황 모형을 제시하였다. 그리고, 기존 정보시스템의 규모를 "특정 기업이 산업 전반에 걸쳐 정보시스템이 구축되어 있는 상대적인 크기"로 정의하였다. 상대적인 크기에 의한 상황 모형의 제시는 다음과 같은 의문을 지울 수가 없다. 즉, 산업 내에서 상대적으로 정보시스템 규모가 적은 기업이 업무에 있어서 복잡도가 얼마나 클 수 있을 것인가? 반대로, 상대적인 정보시스템 규모가 큰 기업의 업무처리의 복잡성이 그렇게 작을 수 있을 것인가? 또한 데이터량과 사용자수라는 변수가 고려되지 않은 채 업무처리의 복잡도만으로 클라이언트-서버 아키텍쳐의 제시가 가능한 것일까? 결국 상황 요인과 관련된 클라이언트-서버 아키텍쳐의 제시(pp.258 그림 2)는 기업이라는 조직을 너무 획일적인 기능을 갖는 시스템처럼 취급하고 있다. 실제로 기업의 업무 시스템은 기능별로 상당히 다른 특성을 보이고 있다(예를 들면, 데이터량은 많으나 비교적 간단한 업무와 데이터량은 적지만 복잡도는 높은 업무, 또 사용자 수가 많은 업무와 적은 업무 등의 경우이다.). 즉, 상황 모형의 차원에서 본다면 상당히 다른 아키텍쳐를 가질 수 있다(김&박(1995)). 따라서 pp.257 그림 1 의 상황 모형에 대한 의문이 제기되는 것이다.

그리고 앞에서도 살펴본 바와 같이 상황 모형은 3 차원으로 표시하며 이것은 기업내의 기능이 어떠한가에 따라 적절한 시스템의 선택(예를 들면 2tier 가 나을 것인가? 또는 3tier 가 나을 것인가? 와 같은 것이다.)을 위한 가이드 역할을 한다. 표 1 에 3 차원 상황 모형을 나타 내었다.

모형에도 제시되어 있는 바와 같이 사용자수가 많은 경우는 복잡도나 데이터량과 상관없이 3 단계 구조 또는 로직분할구조를 선택하는 것이 바람직하다. 특히 로직분할구조의 경우는 융통성이 있는 구조이긴 하나 본인의 주장대로 한다면 반드시 소프트웨어적인 3 단계구조(3-tier)가 되어야 한다. 그리고, 사용자수가 적은 경우는 선행 연구자의 모형과 비슷하나 로직분할구조가 2tier 라는 점이 다르다.

이상에서 제시된 모형은 기업의 상황에 따라 폭 넓은 선택이 가능할 것이다. 그러나, 상위 구조로의 선택 (예를 들면, 원격데이터로 충분한 경우에 로직분할구조를 선택하는 것과 같은) 은 선택자의 마음이라고 볼 수 있으나, 그에 따른 비용과 기간 및 관리상의 어려움은 고려해야 할 것이다.

3-3. CAC 모델

이상의 논의에서 살펴본 바에 의해 구조의 수정이라는 종속변수에 영향을 주는 2가지 독립변수(stored concept, virtual presentation-이것은 저자가 명명한 것이다.)와 상황모델의 수정이라는 종속변수에 영향을 주는 3가지 독립변수(데이터의 량, 업무의 복잡성, 사용자 수)와의 관계를 그림 2에 나타내었다. 이것은 체계적인 클라이언트/서버 아키텍쳐와 상황 모델을 위한 변수와 그것의 상관 관계를 나타내는 것으로서 CAC Model (Comprehensive client/server Architecture and Contingency Model) 이라고 명명하기로 한다. 앞에서 언급이 된 버츄얼 프리젠테이션(virtual presentation)의 개념은 컴퓨팅 파워(computing power)의 분산에 따른 클라이언트 관리의 어려움으로 인해 나타난 개념이며, 사용자수는 혼합된 의미를 내포하고 있기에 분리된 변수이다. 그리고, 상황모델은 클라이언트/서버 구조에 그 근본을 두고 있음은 자명하다.

제 4 장. 결 론

만약 데이터량에서 사용자수를 분리하지 않았다고 가정해 보면 사용자수가 이미 전산 자원의 한계를 넘어서는 경우에도 업무의 복잡성이 낮은 업무들은 원격프리젠테이션이라는 구조를 가져 갔을 것이다. 또한, 앞에서도 언급이 되었지만 중대형 기업의 시스템들은 어느 정도는 데이터량이 많고 업무가 복잡하다고 언급하였다. 그럴 경우는 거의 모두가 3단계 구조나 로직분할구조로 가져 가야 했었다. 그러나, 사용자수를 분리하고 나니 위의 모형과 같이 정리가 되어 기업의 입장에서는 상황에 따라 보다 폭 넓은 선택이 가능하게 되었다. 선행 연구자들의 부족한 점을 들추어 내기는 했으나, 이 연구에 지대한 영향과 도움을 주었음은 물론이다. 실로 시스템 통합 업계에서 13년이란 세월을 개발에 전념했던 본인 조차도 미처 생각할 수 없었던 것을 선행 연구자들의 도움으로 본인의 CAC Model이 탄생한 것을 생각하면 고마울 따름이다. 앞으로도 정보기술은 우리가 상상할 수 없을 만큼 급속하게 변화해 갈 것이다. 아무쪼록 이 연구의 결과가 후행 연구자들에게 조금이라도 도움이 되었으면 하는 마음 간절하다.

참고문헌

1.김영걸,박영면,"기업의 정보처리특성과 클라이언트/서버 아키텍쳐 구현전략에 관한 연구",한국경영정보학회 춘계학술대회 논문집,1995.6., pp.159-188

2.문태수,정윤,"클라이언트-서버 아키텍쳐 구축을 위한 상황모형의 개발에 관한 탐색적 연구",한국경영정보학회 춘계학술대회 논문집,1997.6., pp.251-259

3.엄기현(역), 클라이언트/서버 구조, 이한출판사, 1995

4.Berson, A., Client/Server Architecture, Singapore:McGraw-Hill Inc., 1994.

5.Citrix Systems, Inc. , Server-based Computing White Paper, 1999.

6.Edelstein, H., "Unravelling Client-Server Architecture",DBMS, May 1994, pp. 34-42

7.http://www.kds.co.kr/kdsns/sbc.htm

8.http://www.kds.co.kr/kdsns/th_cli_1.htm

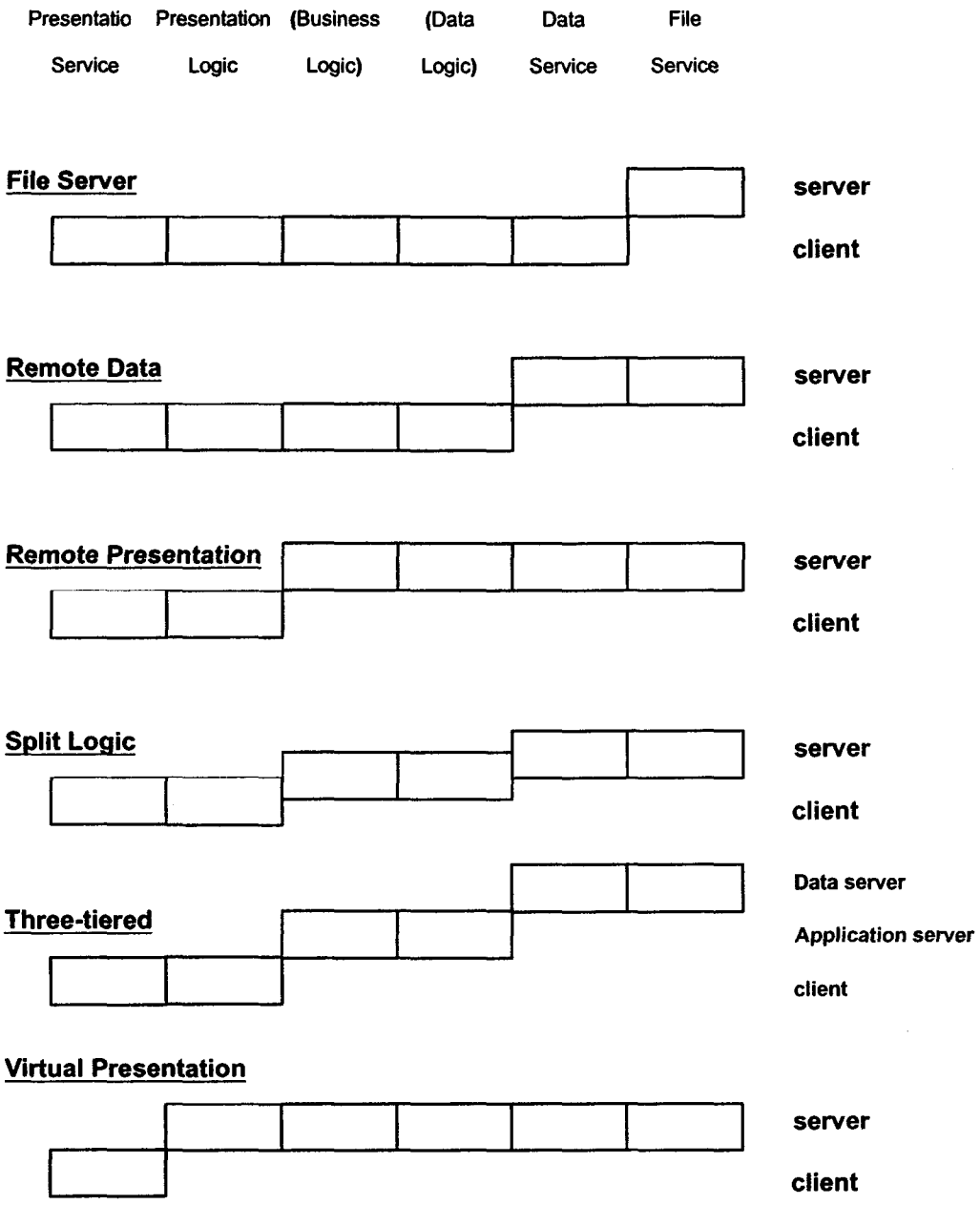

그림 1. 체계적인 클라이언트/서버 구조

		데이터량이 많은 경우	데이터량이 적은 경우
사용자수가 많은 경우	복잡도가 큰 경우	3 단계구조, 로직분할구조(3-tier)	
	복잡도가 작은 경우		
사용자수가 적은 경우	복잡도가 큰 경우	로직분할구조(2-tier)	원격데이터
	복잡도가 작은 경우	원격프리젠테이션	파일서버

표 1. 체계적인 상황모델

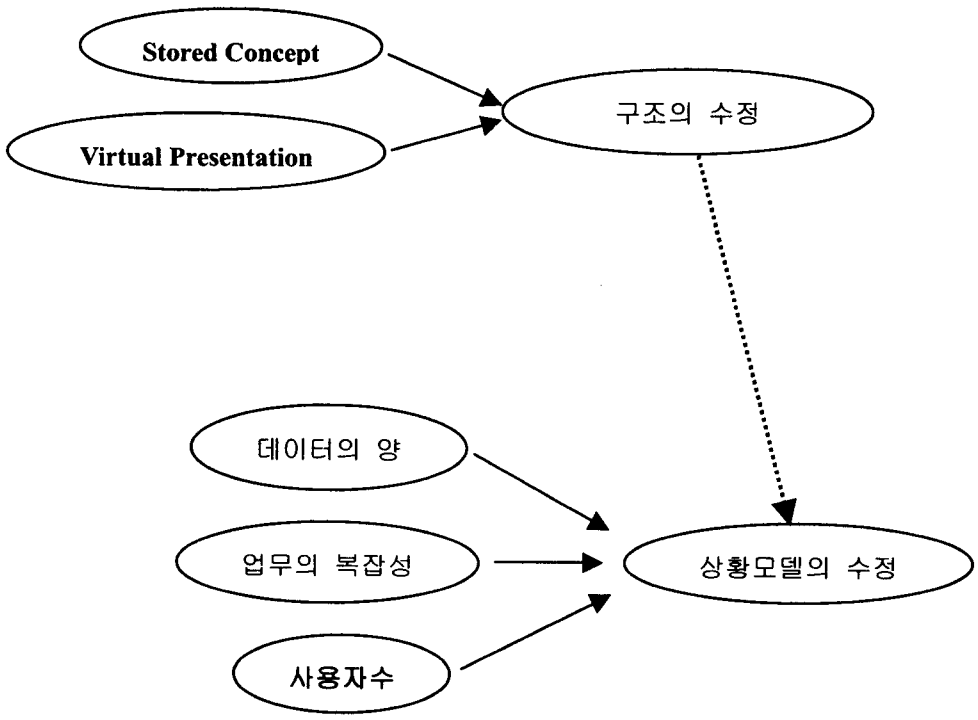
Stored Concept
Virtual Presentation
구조의 수정
데이터의 양
업무의 복잡성
사용자수
상황모델의 수정

그림 2. CAC 모델