■ 論 文 ■

국가 지능형 교통체계를 위한 아키텍쳐 연구
(모형 및 방법론)

A Study of Architecture for national Intelligent Transportation Systems
(Methodology and Model)

백 인 섭 (아주대학교 정보과학과 교수)
이 승 환 (아주대학교 환경·도시공학부 교수)
이 시 복 (도로교통안전관리공단 연구위원)

목 차

Key Words : ITS(Intelligent Transportation System)-Architecture, System Architecture, Architecting Process, Architecture Model

요 약

우리 나라 환경에 적합한 국가 지능형 교통체계(ITS:Intelligent Transportation Systems)의 구현을 위한 국가 아키텍쳐 수립연구가 국토연구원과 관련 학계와 산업계의 협동연구로 지난 3년간 수행되어 천여 페이지에 달하는 국가 아키텍쳐가 수립된 바 있다. 본 논문에서는 이를 위해서 고안된 도메인 수준과 논리 수준 및 물리 수준의 아키텍쳐 모형과 방법론을 제시한다. 도메인 수준의 아키텍쳐는 ITS를 구현함에 있어 관계되는 영역간의 국가적 협동체계를 이룩하기 위한 국가적 기본 틀이고 논리수준의 아키텍쳐는 서비스의 중복/사각/상충을 방지하기 위한 국가적 기본 틀이며 물리수준의 아키텍쳐는 서비스 시스템을 구축함에 있어 물리적 정보기술 자원을 효율적으로 할당하여 시스템 구축의 경제성과 효율성을 도모하기 위한 시스템의 물리적 구성 틀이다. 이를 위한 모델과 기법은 고전적 컴퓨터 기반 시스템공학에서의 프로세스 지향적 사고와 기법을 수정 보완하고 확장한 것으로 고 수준(High Level)에서의 아키텍쳐 구상에 적절한 것이다. 저 수준(Low Level)의 아키텍쳐는 설계 수준에서 이루어지는 것으로 전제하였다.

Ⅰ. 서론 : 대형 정보체계와 아키텍쳐

정보체계의 합리적 구현을 위한 아키텍쳐 개념이 90년대 초반에 제시되어 기존의 프로세스 위주의 소프트웨어 공학에서 간헐적이고 부분적으로 사용되면서 여러 가지 형태로 발전하다가[3,8,9] 90년대 중반 들어 객체지향 소프트웨어 공학에 도입됨으로서 시스템의 전 수명주기 속에서 명확한 의미를 가지고 일괄성 있게 활용되게 되었다[6,10]. 예를 들어 90년대 중반 미국에서 범국가적으로 시도한 국가지능형 교통체계(ITS:Intelligent Transportation Systems)를 위한 아키텍쳐 구상에는 이러한 기존의 프로세스 지향적 아키텍쳐 개념이 적용되었다[11~13]. 그러나 기존의 프로세스 중심 아키텍쳐 개념은 데이터 모델링과 프로세스 모델링이 전혀 다른 별개의 수단과 방법론에 의해서 구상되었기 때문에 상호 호환성이 없어 이들을 통합하기가 매우 어려워 실제 활용이 크게 제약받을 수밖에 없었다. 반면에 요즘 새롭게 대두되고 있는 객체지향 아키텍쳐 개념은 데이터 모델링과 프로세스 모델링을 한데 묶어 객체화함으로써 이러한 문제를 해소 시켰으며, 또한 전체 수명주기 속에서 아키텍쳐의 의미와 역할을 명확하게 정립하였고 관련되는 모든 단계에서 일괄성을 가지는 모델들을 사용하였다[4,7]. 따라서 아키텍쳐 구상 방식에서도 프로세스 지향 방식에서 객체지향 방식으로의 대 전이가 이루어 질 것으로 전망된다. 일 예로서 90년대 후반에 제정된 ITS를 위한 국제표준으로서의 아키텍쳐 모형은 이미 이러한 객체지향 방식을 채택하고 있다[6].

그러나 객체지향 방식의 아키텍쳐 개념은 객체지향 방식이 기본적으로 추구하는 것이 생산성(Productivity) 제고이기 때문에 그것을 위한 재사용성(Reusability)과 자동화성(Automation) 그리고 확장성(Flexibility)에 초점이 맞추어 질 수밖에 없다. 이는 객체지향 방식에서의 아키텍쳐 개념은 설계이전의 여러 가지 추상적이고 개념적인 정의 단계의 아키텍팅 즉 "거시적 수준 아키텍쳐" 보다는 모든 정의가 명확하게 이루어진 다음의 구체적인 구현을 위한 설계단계의 아키텍팅 즉 "미시적 수준 아키텍쳐"에 보다 적합하다는 의미가 된다. 또한 객체지향 아키텍쳐는 모델과 모델링 수준에서는 이미 성숙단계에 있지만 아키텍팅 방법론 차원에서는 아직 성숙되지 못한 단계에 있다[14]. 바로 이러한 방법론 부재(혼재)의 문제 때문에 객체지향 아키텍쳐 국제표준이 아직 국제표준이 되지 못하고 TR(Technical Report) Type 2(3년간 보류) 상태에 처해 있는 상태이다. 따라서 객체지향 방식은 국가 지능형 교통체계와 같이 초 대규모적이고 기술적으로 복잡 다양하며 운영관리 주체가 매우 다양하고 다 조직적인 정보체계를 초기 단계에서 합리적으로 정의하기 위한 거시 아키텍쳐 모형으로는 적합성이 결여된다고 볼 수 있다. 고로 이러한 다수의 프로젝트로 구성될 국가 프로그램 수준의 거대 정보체계를 합리적이고 효율적으로 구현하기 위한 초기 시작 단계에서의 거시적 아키텍쳐 모델은 기존의 소프트웨어 공학이나 객체지향에서 추구하는 모델보다는 상위 수준으로 추상화된 새로운 모델과 방법론이 필요하다 하겠다. 따라서 본 논문에서는 우리 나라 국가 지능형 교통체계라는 거대 정보화체계의 아키텍쳐 구상을 위해서 고안되고 실제로 건교부/국토연구원의 ITS 국가 아키텍쳐 구상 작업에 적용된 아키텍쳐 모형과 방법론을 제시하고자 한다[1,2]. 다음 논문에서는 이를 적용해서 실제로 구상된 우리 나라 아키텍쳐 사례 중 우리 환경에서 가질 수밖에 없는 특수성에 대해서 논하기로 하겠다.

Ⅱ. 국가 ITS를 위한 아키텍쳐의 필요성 및 요구조건

국가지능형 교통체계란, 보다 안전하고 효율적이고 편리한 국가 교통체계를 달성하기 위해서 국가의 교통자원(도로, 차량, 노변장치, 차량장치, 여행자장치, 등), 교통행위(여행 및 화물 운송, 등) 및 교통관련 운영관리 행위(교통 및 도로 관리, 안전 관리, 사고 관리, 등)를 지능화하고 이들이 합리적이고 효율적으로 상호보완성을 가지도록 구조화한 정보체계이다. 즉, 국가 ITS는 정보화 대상과 필요 기술 차원에서 매우 복잡 다양하고 대규모적인 체계이다. 또한 서비스 차원에서도 교통 및 도로에 대한 실시간 제어 유형의 각종 실시간 제어 서비스에서부터 효율적인 여행 및 운송을 위한 각종 실시간 트랜잭션 처리 서비스는 물론 다양한 온라인 사용자 편의 서비스까지 모두 제공해야하는 초 대규모 정보체계이다.

따라서 국가 ITS는 여러 가지 다양한 정보체계들로 구성될 수밖에 없으며, 범 국가적인 조직이 참여할 수밖에 없고 또한 다양한 정보기술이 적용되면서

장기적 안목에서 점진적으로 구현될 수밖에 없다. 그 중 일부는 국가의 기반시설로서의 인프라(Infra-Structure)를 구성하고 이들을 기반으로 해서 여러 가지 상부 구조적(Super-Structure) 서비스들이 가능해지는 것이다. 이러한 국가 프로그램 수준의 초 대규모적이고 복잡 다양한 시스템을 구현함에 있어 가장 중요한 것은 설계단계의 합리성이나 개발 단계의 생산성 등 미시적 관점의 이슈가 아니라 초기 정의 단계에서 중복이나 상충이 배제되면서 사각지대가 발생하지 않도록 합리적으로 전체를 통합적으로 정의하고 법·제도 조직적인 상충의 문제를 사전에 조율함으로써 거시적 관점에서의 국가적 경제성 내지는 효율성이 극대화되도록 구조화하고 또한 한꺼번에 구현이 불가능한 전체를 적당한 규모의 부분으로 구조화함으로써 해당 주체로 하여금 해당 부분을 점진적으로 구현하도록 하는 것이다. 이러한 것을 초기 정의단계에서 거시적 관점으로 구상하는 것이 거시 아키텍쳐이며 본 논문에서는 이러한 거시적 관점의 아키텍쳐 구상을 위해서 적절한 모델 과 관련 표기법 및 방법론을 제시하고자 한다. 따라서 본 논문에서 제시하는 아키텍쳐 모델과 방법론은 다음의 5가지 관점에 초점을 맞추어 구상되었다.

1. 사용자 서비스 지향 아키텍쳐

ITS 서비스는 매우 다양하기 때문에, 수요자 측면에서 요구되는 서비스가 중복되기가 쉽고 서로 상충되는 서비스가 요구될 수도 있으며 보다 고차원적 서비스는 미쳐 요구되지 않아 서비스 측면의 사각지대가 발생할 소지도 크다. 따라서 다양한 사용자(Stake Holders)들로부터 요구되는 각종 서비스가 중복이나 사각을 피하고 상충되는 것들이 적절하게 조정될 수 있도록 아키텍쳐가 구상되어야 하며, 통합 조정된 서비스가 아키텍쳐의 시작점이 되어야 한다. 이를 위해서는 아키텍쳐 구상의 첫 단계에서 요구서비스들이 합리적으로 통합 정의되어야 하고 이것에 의해서 시스템의 기능이 부여되어져야 한다. 서비스가 중복되거나 상충되거나 또는 사각지대가 발생하는지 여부를 파악하기 위해서는 요구 서비스를 수집해서 통합 정의 할 때 각각의 서비스에 대해서 설명적 명칭(Descriptive Name)만이 아니라 세부적 서비스 내용, 서비스 대상(사용자),서비스 목적 및 필요성, 시기, 제공자, 예산출처, 등에 대한 정보까지 수집되어 가시적으로 표현되어야 한다.

2. 합리적 기능성 제고를 위한 아키텍쳐 (for Working System)

필요한 ITS 서비스를 제공하기 위한 공급체계 또한 기술적 복잡 다양성과 체계 구성의 대규모성 및 관련 기관의 다양성으로 인하여 자원이나 기능차원에서의 중복, 사각, 상충이 발생할 소지가 크다. 따라서 다양하고 거대한 공급체계들을 구현함에 있어서도 자원적 또는 기능적 중복은 최소화하고 사각을 피하고 상충되는 것은 적절하게 조정될 수 있도록 아키텍쳐가 구상되어야 한다. 이를 위해서는 공급체계의 모든 기능이 합리적으로 통합된 제공 서비스로부터 도출되도록 함으로서 기능성 차원에서의 합리적 전체 통합성(Working System)이 얻어질 수 있도록 되어야 한다. 이 경우 기능이나 자원의 중복은 재사용(Reuse) 관점에서 해결되어질 수 있도록 하여야 한다.

3. 합리적 품성 제고를 위한 아키텍쳐 (for Workable System)

ITS 공급체계는 제공 기능측면에서는 물론 구성 기술측면에서도 매우 복잡 다양하다. 또한 물리적 설치 공간에서도 일반적 사무공간 뿐만이 아니라 각종 도로 또는 각종 차량 또는 각종 여행자 단말 등 여러 형태의 특수 공간에 설치되어야 하는 특성을 가진다. 따라서 구현될 공급체계가 적절한 작동성(Workable System)을 가지기 위해서는 품성적 조건 또한 복잡 다양할 수밖에 없으며 이는 설계·개발에서 결정적 중요성을 가지기 때문에 초기 아키텍쳐 단계에서 요구되는 모든 품성적 조건들이 명확하게 도출되어 가시화 되어야 한다.

4. 법·제도·조직적 수용성 및 사회·문화적 수용성 제고를 위한 아키텍쳐(for Acceptable System)

사업 영역(Domains)별로 구현 제시되는 각 시스템들이 요구된 기능성이나 품성을 충분히 만족한다해

도 그 시스템을 운영 관리함에 있어 국가의 현행법이나 제도상 또는 조직상의 문제를 가지는 경우 실제 운영상의 심각한 문제가 초래될 수가 있다. 이러한 경우가 발생하면 경제적으로 시간적으로 막대한 국가적 손실을 초래하게 된다. 이것은 정보기술의 문제도 아니고 도로기술이나 교통기술의 문제도 아닌 비 기술적 문제이지만 문제의 치명성으로 인하여 반드시 실제 설계 및 개발 이전의 초기 아키텍쳐 정의 단계에서 반드시 적절하게 조정되어져야 한다.

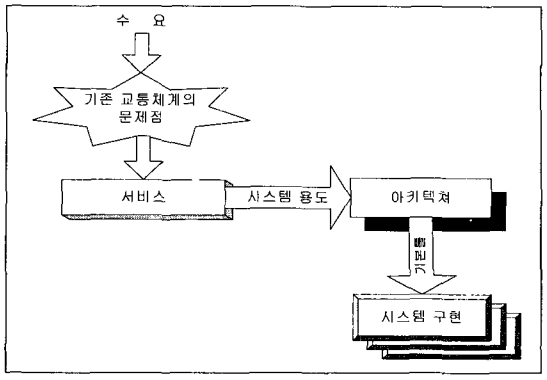
5. 구현의 기술적 시간적 경제적 타당성 제고를 위한 아키텍쳐(for Feasible System)

아키텍쳐 구상단계에서 시스템의 분해 및 정의는 현재 및 장래의 기술적 여건과 경제적 측면 및 시간적 측면에 따른 시스템 구현가능성을 검토하여야 한다. 구상된 시스템의 서비스와 기능 및 품성이 이상적이라 할지라도 기술수준이 뒷받침되지 못하든지, 경제성이 없다든지 또는 시간적으로 적기를 마칠 수가 없다든지 하면 구현이 매우 어렵거나 불가능하게 될 수도 있기 때문이다. 따라서 이러한 기술적, 시간적, 경제적 타당성 문제가 검토될 수 있도록 아키텍쳐가 구상되어져야 한다.

Ⅲ. 국가 ITS를 위한 아키텍쳐 모형

1. ITS 사용자서비스와 아키텍쳐의 관계

ITS가 제공하는 사용자서비스는 기존 교통체계가 안고있는 문제점으로 인해 발생되는 ITS분야의 수요라 할 수 있다. 다시 말해, 사용자서비스란 ITS를 통해 달성할 수 있는 문제점의 해결이나 개선 목표인 것이다. 또한 ITS 시스템의 구축은 이러한 수요에 대응한 공급이라 할 수 있는데 이를 위하여 아키텍쳐는 시스템 구축을 위한 기본 틀(Framework)로서 사용자서비스를 반영하는 역할을 한다. 따라서 ITS 분야의 수요인 사용자서비스와 공급의 틀인 아키텍쳐는 불가분의 상호관계가 성립되므로 아키텍쳐는 사용자서비스 제공에 가장 유리한 형태로 구상되어야 한다. ITS 서비스와 아키텍쳐의 관계는 〈그림 1〉에 도시한 바와 같다. 국가 ITS 서비스란 매우 다양하기 때문에 요구과정에서 서비스가 중복 될 수 있고 서로 상충되는



〈그림 1〉 서비스와 아키텍쳐 관계

경우 그리고 어디에서도 요구되지 않는 서비스의 사각지대도 발생할 소지가 많다. 이러한 중복·상충·사각의 문제는 요구 서비스를 수집 통합하는 단계에서 적절하게 분석 정리되어져야 한다.

2. 아키텍쳐 구상 프로세스

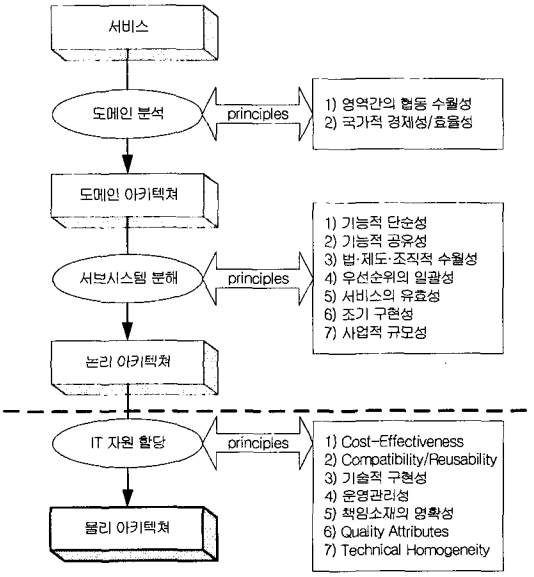
아키텍팅 방법론으로서의 구상 프로세스는 기존의 시스템 공학 분야에서의 정보공학적(Information Engineering) 접근 방식과 제품공학적(Product Engineering) 접근 방식이 있다[8]. 그러나 기존의 정보공학적 접근 방식은 사업 환경의 정보화를 위한 거시적 관점의 아키텍팅에 초점을 맞추어 물리적 구성 틀을 구상함에는 취약성을 가질 수밖에 없으며, 제품공학적 접근 방식은 제품(Product)위주의 관점에 초점을 맞추어 상위 수준의 거시적 아키텍쳐를 구상함에는 취약성을 가질 수밖에 없다. 또한 새롭게 대두되고 있는 객체지향 방식에서의 방법론은 매우 다양하게 발전되어 오다가 최근에 이른바 RUP(Rational Unified Process)모델[13]로 통합되어 Defacto 수준의 표준으로서 자리를 굳히고 있지만 이는 아키텍팅 프로세스가 아닌 시스템의 전체 수명주기에 대한 프로세스 모델로서 아키텍쳐를 포함하고 있지만 그 개념이 시스템의 기본 틀(Framework)이 아닌 기본 골격(Skeleton)으로서 미시적 아키텍쳐에 초점이 맞춰 있는 상황이다.[14] 일례로 객체지향방식의 아키텍쳐 국제표준안인 ISO TR 14813은 적절한 방법론의 부재로 인해서 현재 시한부 보류상태에 처해 있는 상황이다[6].

따라서, 이러한 취약성 문제를 해결하기 위해서는

이들을 상호 보완적으로 적절하게 결합시킨 새로운 방법론이 필요하다 하겠다.

본 논문에서 제시하는 새로운 아키텍팅 방법론으로서의 아키텍쳐 구상 프로세스는 우선 거시적 관점에 초점을 맞추었고(거시 아키텍쳐) 따라서 객체지향 방식이 아닌 기존의 구조적(프로세스 지향) 방식을 취하며 이를 도시하면 〈그림 2〉와 같다. 여기서 육면체로 표시된 것은 구상되는 아키텍쳐를 의미하며 원으로 표시된 것은 프로세스를 의미한다. 각 프로세스는 해당되는 구상원칙에 따라 수행되어야 한다. 아키텍쳐는 3개의 다른 수준에서 구상된다. 이 중 상위 두 개의 수준은 정보 공학적 접근 방식을 지향하며 하위 1개 수준의 제품 공학적 접근방식을 지향한다.

최상위 수준에서 서비스 분석을 통한 도메인 아키텍쳐가 구상된다. 도메인 아키텍쳐란 도출 정의된 서비스들을 국가환경에서 최선의 합리성을 가지도록 사업 영역별로 할당함으로서 고 수준의 사업영역 즉 도메인을 정의하고 이들간의 상호 협동적 관계를 정의하는 도메인 수준의 기본 틀이다. 이 수준에서의 아키텍쳐 구상은 거시적 관점에서의 국가적 경제성과 효율성(중복/사각/상충을 방지하는)을 극대화하도록 이루어져야 한다. 도메인 아키텍쳐의 구조는 일반적으로 공공영역의 국가 인프라를 구성하는 하부구조(Infra-Structure)와 민간영역의 다양한 서비스를 구성하는 상부구조(Supra Structure)로 구성되지만



〈그림 2〉 아키텍팅 프로세스 및 설계 지침

(미국, 유럽연합, 일본 등 모두 2계층 구조를 가짐), 우리 나라의 경우는 현행 법 제도적 복잡성에서 유발되는 여러 가지 비합리성 문제를 기술 수준에서 해결하기 위해서 중간구조(Middle Structure)를 설정함으로서 3계층 구조를 가지도록 하였으며 도출 정의된 시스템들은 각각의 특성에 따라 이 3계층 구조 속으로 적절하게 설정되어진다. 중간구조는 공공영역으로서 기술적으로나 법·제도적인 복잡 다양성과 운영의 계속성 유지 때문에 폐쇄성이 강할 수밖에 없는 하부구조와 민간영역으로서 개방성이 강할 수밖에 없는 상부구조간의 중간 매개체로서 상호 협동이 원활하게 이루어 질 수 있도록 하기 위하여 준공공의 성격을 가지도록 정의하였고 이는 전 세계적으로 유일한 한국적 해결방안이라 볼 수 있다.

다음 두 번 째 수준에서는 이들 도메인 수준의 시스템들을 적절한 분해 과정을 통해서 구현 수준의 서브시스템들로 도출 정의하고 이들간의 관계를 구조적으로 정의함으로써 논리 아키텍쳐가 구상된다. 논리 아키텍쳐를 구상하기 위한 모형과 방법론은 3장에서 설명된다.

마지막 세번째 수준에서는 논리 아키텍쳐에서 정의된 각각의 서브시스템에 대하여 최적의 물리적 자원을 적절하게 할당함으로서 물리적 아키텍쳐가 구상된다. 이를 위한 적절한 모형과 방법론은 4장에 기술하였다.

3. 논리 아키텍쳐 모형 및 구상 원칙

논리 아키텍쳐란 각각의 사용자서비스를 구현하기 위해서 필요한 ITS의 기능단위를 설정하고, 이를 기반으로 기본 요소시스템으로서의 서브시스템을 도출·정의하고 이들간의 관계를 정보흐름(Data Flow)수준으로 연계 정의하는 시스템의 논리 수준의 기본 틀이다. 서브시스템이란 ITS 사용자서비스를 실제로 제공하기 위한 서비스구현단위로써 실제 ITS구축사업의 최소 구축단위라고도 할 수 있다. 서브시스템을 설정하는 목적은 다양한 ITS기능을 모듈(module)화 하고 패키지화(Package)하여 ITS서비스를 선택적, 점진적으로 구현할 수 있도록 하기 위한 것이다. 논리 아키텍쳐 구상은 크게 두 가지로 대별된다. 하나는 ITS 사용자 및 운영관리자를 포함한 ITS와 이해관계를 가지는 모든 외적인 요소들과 서브시스템간의 관계를 ACD(Architecture Context Diagram) 모델[8]을

통해서 모형화 함으로써 시스템의 외부 환경을 정의하는 것이고 다른 하나는 서브시스템간의 정보흐름 관계를 AFD(Architecture Flow Diagram)[8] 모델을 통해서 모형화 함으로써 시스템의 내부 구성을 정의하는 것이다. 서브시스템간의 관계가 합리적으로 정의되기 위해서는 각 서브시스템 내부에서 수행되어야 할 세부기능이 정의되어야하며 동시에 이러한 기능과 흐름정보간의 관계가 특정 형식을 사용하여 모델화 되어야 한다. ITS 사용자서비스는 특성상 복수 조직적이며 기능적으로 복합적이어서 일차적으로 도출되는 각 사용자서비스 자체를 시스템화하게 될 경우 중복은 물론이고 현행 법·제도와 상충이 벌어지는 등 여러 가지 풀기 어려운 기술외적 문제들이 야기 될 소지가 있다. 따라서 각 서비스가 이러한 비기술적이지만 풀기 어려운 고유특성을 가지지 않도록 서브시스템이 단순 조직적이고 단순 기능성을 가지도록 분해 정의되어야 한다. 서브시스템의 분해 설정과정을 위한 설계지침으로 본 논문에서 제시하는 원칙은 다음과 같다: 상위 두 개의 원칙은 일반적으로 논리 아키텍팅 단계에서 사용되는 원칙이지만 하위 4개의 원칙은 본 논문에서 새롭게 제시되는 중요한 원칙들로 특히 우리 나라와 같이 법제도 조직적 경직성을 가지는 환경에서는 결정적 중요성을 가진다 하겠다.

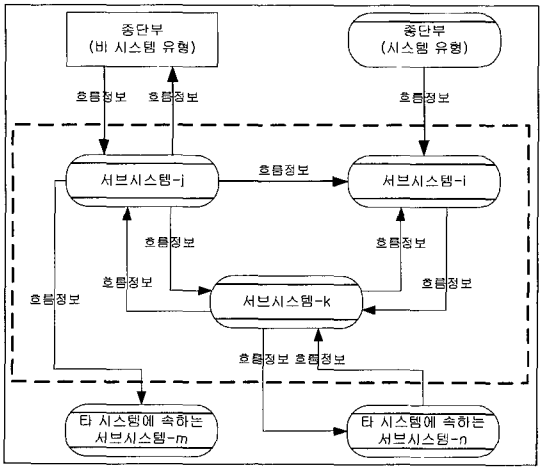
■ 논리 아키텍처 구상 원칙

- 사용자서비스와 서브시스템의 동급화 : 사용자서비스와 서브시스템은 여러 형태의 대응관계(일대일, 다대일 또는 일대다)를 가질 수가 있지만 서브시스템 수준의 재사용성(Reuse in Macro-Level) 제고를 위해서 가급적 일대일 관계를 가지도록 한다.
- 서비스의 기능적 세분화/단순화 : 각 사용자서비스는 여러 개의 복합적인 비지니스 수준의 기능을 포함하고 있는 경우가 많다. 이러한 경우, 복합적인 비지니스 기능을 분리시켜 세부서비스화해서 독자적 서브시스템으로 구성함으로써 실제 구현 시에 서브시스템 수준의 재사용성이 제고되도록 하여야 한다.
- 기술적 구현난이도의 동질화 : 특정 서브시스템이 복합적 사용자서비스 기능을 포함해야 하고 그들 기능간에 기술난이도가 확연히 다른 경우, 현재와 장래의 기술수준을 고려하여 기술의 난이도가 동질화될 수 있도록 서브시스템을 분해 설정함으로써 구현성을 제고한다.
- 법·제도·조직적 수월성 제고 : 임의 서브시스템의 운영주체 또는 관리주체가 다양한 경우 또는 관련 법 체계 등이 상이한 경우, 실제 구축된 서브시스템의 운영 및 관리가 매우 어려워 질 수도 있다. 따라서, 서브시스템 설정 시에 가급적 기존의 법·제도·조직적인 틀을 충분히 반영하여 실제 구현 및 운영관리 차원의 수월성을 제고한다.
- 조기 구현성 : 서비스구현의 시급성에 따라 특정 서브시스템의 경우는 조기 구현성이 필수적으로 요구될 수도 있다. 그러한 서브시스템의 경우는 그것이 조기 구현성을 가질 수 있도록 가능한 한 독립된 서브시스템으로 분해 정의되어야 하며 이를 위해 필요한 기능들이 부가적(중복적)으로 부여되어져야 한다.
- 사업단위로의 적정성 : 서브시스템은 최종사용자(End User)를 대상으로 하는 서비스를 제공함과 동시에 사업단위로서 적정한 규모와 수준이 되어야 실제 서비스제공자(특히 공공부문)가 이를 사업화하기에 용이하다. 따라서 필요한 경우 서브시스템 정의가 사업단위로서의 적정한 규모를 가지도록 이루어져야 한다.

이상의 다양한 원칙 가운데 가장 기본적인 것은 각각의 서브시스템은 한 개의 사용자 서비스에 해당하도록 설정하는 것이다. 따라서 먼저 통합적으로 정의된 각각의 사용자서비스는 앞에서 언급된 여러 가지 원칙이 만족될 때까지 세분화되고 구체화되어야 한다. 이렇게 적정 수준의 서브시스템이 도출 정의되면, 해당되는 서비스를 구현함에 있어 서브시스템 수준의 재사용성이 극대화 될 수가 있다. 또한 서브시스템을 합리적으로 설계 개발하기 위해서는 그것이 제공해야 하는 정보 기술적 기능이 도출되어야 한다. 서브시스템의 설계 개발 과정에서 타 서브시스템과의 기능 중복이 발생하는 경우는 기능 수준의 재사용(Reuse in Micro-Level)이 이루어지도록 기능 또한 적절한 단위로 구조화되는 것이 바람직하다.

1) 서브시스템간 정보흐름 연계 정의 모형

앞에서 제시한 원칙에 따라 분해 설정되는 각 서브시스템은 적절한 수준 즉 응용 서비스시스템 단위 수준으로 상세화 된다. 이들 각각의 서브시스템은 상호 밀접한 연관관계를 가지게 된다. 따라서, 이러한 서브시스템간의 상호연계에 따른 정보흐름관계를 구체적



〈그림 3〉 논리 아키텍쳐 모형

으로 명시할 필요가 있으며, 이를 위해 시스템 환경도(ACD)와 정보흐름연계도(AFD)를 구상한다. 〈그림 3〉은 ACD와 AFD모형 형식을 나타낸 것으로 Hatley의 시스템 아키텍쳐 모형을 수정 확장한 것이다[5]. 점선은 임의의 서비스 영역(또는 사업 영역)의 한계를 의미한다. 따라서 점선 내부의 서브시스템은 해당 서비스 영역내의 세부서비스시스템에 해당하며, 점선 밖의 서브시스템은 다른 서비스 영역의 세부서비스시스템에 해당한다. 외부의 종단부는 본래 모형을 확장한 것으로 두 가지 유형으로 구분할 수 있는데, 유형 1은 사람이나 현상 같은 비 시스템적 유형의 종단부를 의미하고, 유형 2는 정보시스템 유형의 종단부를 나타낸다. 전자는 ITS 관련 서브시스템이 구현될 때 직접적으로 적절한 인터페이스를 통해서 연결될 수 있지만, 후자의 경우는 ITS와 기존 체계간의 적절한 프로토콜 변환을 통해서 연결이 가능하게 된다. 논리 아키텍쳐 구상에서 중요한 것은 구성요소들간의 관계가 제어흐름(Control-Flow)은 배제하고 정보흐름(Data-Flow)만으로 정의된다는 것이다. 이는 구성요소들간의 결합(Coupling)을 약하게 하여 임의 구성요소가 변경될 때, 그것의 파급효과를 최소화하고 또한 각 요소가 구현될 때 타 요소에 대한 의존성의 형태를 단순화하여 구현의 유연성(Flexibility)을 도모하기 위함이다. 또한 ITS를 구현함에 있어서 예산상의 문제 기술환경의 문제 또는 법 제도적 환경의 문제 등으로 해서 점진적 구현은 불가피하며 이러한 점진적 구현이 가능하기 위해서는 구현 대상 서브시스템의 타 서브시스템에 대한 의존성을 알아야 한다. 이러한 의존성은 정보흐름 연계도를 통해서 알 수 있으며 따라서 합리적이고 효율적인 서브시스템의 구현 순서를 설정할 수 있는 것이다. 특별한 경우는 이러한 구현 순서를 무시하고 최우선적으로 시급하게 구현되어야 하는 서브시스템들이다. 이러한 경우는 타 서브시스템으로부터 그 것으로 공급되어야 하는 정보를 자체에서 수집할 수 있도록 정보 수집기능을 중복적으로 허용하는 기능정의가 이루어져야 한다.

2) 서브시스템 정의 모형

도출된 서브시스템들을 적절한 수준으로 정의하기 위해서 〈표 1〉과 같은 정의 모형을 사용한다. 내용으로는 대상 서브시스템에서 제공되는 사용자 서비스와 이것에 대응하여 제공되어야 할 기술적 기능에 대한 정의는 물론 관련되는 추진 및 협조주체, 관리영역, 서브시스템으로의 입·출력 정보, 물리적 체계 구성요소에 대한 요구사항, 품성조건 등의 설계 요구사항에 대한 정의가 포함된다. 여기서 추진주체라 함은 서브시스템의 계획에서 설계 개발 그리고 구축 및 운영·관리상에서 모든 책임과 권한을 가지는 주체적 조직을 의미하고, 협조주체라 함은 서브시스템에 대한 직접적 책임과 권한은 없지만 이해관계 또는 간접적인 관계가 있는 조직을 의미한다. 또한, 관리영역이라 함은 제공하는 서비스의 공간적 범위(행정구역 등) 및 도로의 유형을 의미하고, 구성체계라 함은 서브시스템을 개발 구현할 경우 그것의 물리적이고 정보기

〈표 1〉 서브시스템 정의 모형

<table>
<tr><td rowspan="4">서브시스템</td><td>한글명칭</td><td></td><td rowspan="2">관리영역</td><td>공간</td><td></td></tr>
<tr><td>영어명칭</td><td></td><td>도로유형</td><td></td></tr>
<tr><td>소속시스템
(서비스영역)</td><td></td><td rowspan="2">관련주체</td><td>추진</td><td></td></tr>
<tr><td></td><td></td><td>협조</td><td></td></tr>
<tr><td>설 명</td><td colspan="5"></td></tr>
<tr><td colspan="3">제 공 기 능</td><td colspan="3">제공 서비스</td></tr>
<tr><td colspan="3"></td><td colspan="3"></td></tr>
<tr><td rowspan="2">정보흐름명</td><td>입 력</td><td colspan="4"></td></tr>
<tr><td>출 력</td><td colspan="4"></td></tr>
<tr><td>서브시스템
구성체계</td><td colspan="5">센터형, 도로 장치형, 차량 장치형, 여행자 장치형</td></tr>
<tr><td colspan="2">품 성 조 건</td><td colspan="4">성능성 조건, 안전·신뢰성조건,
유지·관리성 등</td></tr>
</table>

술적인 구성 요소를 의미하며 크게 센터 유형과 노변 장치 유형과 차량장치 유형과 여행자 단말장치 유형으로 대별하여 필요한 요소를 일차적으로 정의하는 것으로 이는 물리 아키텍쳐 구상단계에서 명확하게 상세화되어 정의되어야 한다. 품성조건이라 함은 그 서브시스템 구현시 특별히 요구되는 성능성, 안전성, 신뢰성, 편의성 등 품성에 대한 요구조건의 제시를 의미하는데, 이는 물리 아키텍쳐 구상 단계와 시스템 설계 단계에서 적절한 수준으로 구체화되어 반영되어야 한다. 이러한 속성 가운데 특히 중요한 것은 기능정의와 추진 주체이다. 전자는 시스템을 기능적으로 단순화하기 위한 분해의 근거가 되고 후자는 법제도 조직적 수월성을 제고하기 위한 분해의 근거가 되기 때문이다.

흐름정보를 적절한 수준으로 정의하기 위해서는 〈표 2〉와 같은 정의 모형을 사용한다. 해당 흐름정보의 기점과 종점을 서브시스템 수준에서 파악해서 명기하는 것 외에는 일반적인 데이터 사전의 내용과 형식을 따른다. 용도에 대한 명기는 추후 해당정보가 원래의 용도가 아닌 다른 용도로 사용됨으로써 법·제도적·조직적 또는 사회적 문제를 야기하는 것을 예방하기 위한 것이다. 흐름·갱신의 빈도는 후에 정보의 정확성에 대한 문제가 야기되는 것을 예방하기 위한 것이다. 정보 전달 방식 또한 앉아서 전달받는 방식이냐 또는 스스로 가서 가져오는 방식이냐를 명확화 함으로써 후에 조직간의 협력 방식상의 혼돈이나 갈등을 예방하기 위함이다. 흐름정보의 세부항목은 해당 시스템의 분해에 중요한 근거가 된다. 세부항목이 복잡 다양한 경우는 시스템이 바람직한 수준으로 분해되지 않았음을 의미하고 따라서 이 경우는 해당 AFD를 보다 상세화함으로써 흐름정보를 적절한 단위정보로 분해하고 동시에 해당 시스템을 적절한 서브시스템으로 분해 정의하여야 한다.

〈표 2〉 흐름정보 정의 모형

ID :			
정보흐름명			
소속서브시스템(기점기준)			
정보흐름		기점 (→)	(→) 종점
	서브시스템		
	구성요소		
	관련구축단위 (EP)		
설 명			
세 부 항 목			
용 도			
수집 / 생성원			
수집 /생성방법			
흐름(또는 갱신)빈도			
정보전달 방식			

4. 물리 아키텍쳐 모형 및 구상 방법론

물리 아키텍쳐는 논리 아키텍쳐에서 추출된 각 서브시스템을 물리적으로 구현함에 있어 필요한 센터 또는 장치 수준의 물리적 요소시스템을 도출 정의하고 이들간의 작동 관계를 정의하는 시스템의 물리적 구현을 위한 기본 틀이다. 또한 각각의 물리적 요소시스템에 대한 세부 기능을 정의하고 이를 적절하게 패키지 화함으로써 장치 수준에서의 재사용성 제고를 도모한다. 물리 아키텍쳐의 구성요소는 ITS의 경우 크게 센터 유형, 도로장치 유형, 차량장치 유형, 여행자단말장치 유형 등의 4개 유형으로 구분할 수 있는데, 실제 물리 아키텍쳐 구상 시에는 서브시스템의 특성에 따라 이들 유형에 속한 세부 센터 또는 세부 장치들이 설정되게 된다. 각 서브시스템 운영 시 이들 구성요소들은 주어진 세부기능을 수행하고 상호간에 정보를 교환함으로써 소기의 사용자서비스를 제공하게 된다.

1) 물리 아키텍쳐 구상원칙

물리 아키텍쳐 구상은 논리 아키텍쳐에서 도출 정의된 각 서브시스템별 논리수준의 기능을 물리수준의 기능으로 변환하고 이를 적절한 물리적 구성요소(물리적 서브시스템)로 분산·할당하는 작업으로, 우선 서브시스템 기능 구현에 필요한 물리적 구성 요소 즉 각종 센터 및 장치를 규명하고, 각각에 해당하는 수행기능 및 정보흐름 또는 제어흐름을 부여함으로서 이루어진다.

물리적 아키텍팅 방법론은 논리 아키텍쳐와는 달리 제품 공학적(PE:Product Engineering) 접근방식에 근거하면서 아키텍쳐 수준에 적합한 부분만으로 보다 간략하게 수정 보완한 것이다. 즉, 원형에서 제시되는 4가지 수준 중 상위 두 개의 수준만을 선택하여 어플리케이션 수준과 관련 장치 수준으로 수정 보완하였

고 나머지 하위 두 개의 수준은 사업 개발구현시 설계단계에서 이루어지도록 하였다.

적합한 물리 아키텍쳐가 구상되기 위해서는 우선 그것을 구성하는 물리적 구성요소가 적절하게 도출 정의되어야 하고 또한 그것들이 가져야 될 물리 수준의 기능 정의가 적절하게 이루어져야 한다. 이를 위해 본 논문에서 제시하는 물리아키텍쳐 구상 원칙은 다음과 같다:

■ **물리아키텍쳐 구상 원칙**

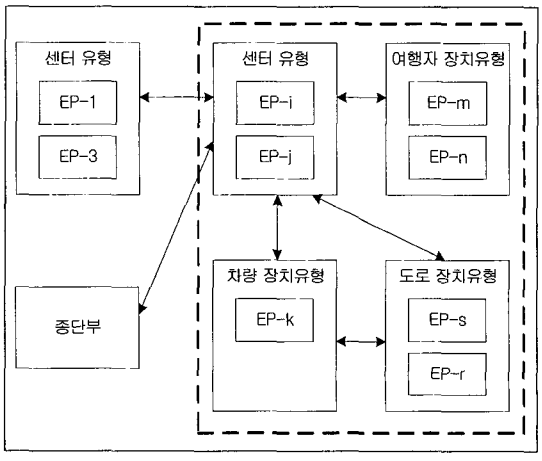
- 물리적 구성요소의 공유성(Sharability) 극대화 : 다수의 서브시스템 구현에 필요한 물리적 구성요소가 동일하거나 유사한 경우, 이들 시설들이 중복이 극소화되면서 효율적으로 공유되도록 역할·기능을 설정하도록 한다.
- 책임성 소재의 합리화 : 물리적 구성요소들에 대한 기능을 설정함에 있어 운영단계에서의 발생 가능한 사고에 대한 책임성 문제가 관련 주체간에 합리적이고 명확하게 배분되도록 기능을 배분 설정하여야 한다.
- 구현성(Implementability)의 극대화 : 시스템 구축 시 필요한 기술이 현실적으로 활용 가능하고 적절한 비용으로 유용한가를 고려해서 실제 구현성을 가지도록 기능 설정을 해야 한다.
- 운영 용이성(Operability) 극대화 : 특히 노변장치 유형의 경우 구축 후에 시스템의 운영이 어려워 서비스의 질이 저하되지 않도록 운영 용이성이 극대화 되도록 기능 설정이 이루어져야 한다.
- 유지 관리의 용이성(Maintainability) 극대화 : 특히 노변장치 유형인 경우 그것의 물리적 위치로 해서 유지관리가 어려워져 결과적으로 서비스의 질이 저하되지 않도록 적절한 유지관리 기능이 설정되어야 한다.
- 안전신뢰성(Trustability)의 극대화 : 관련 장치들의 안전성과 신뢰성이 극대화되도록 기능이 적절하게 분리 설정되어야 한다.
- 유효성(Availability)의 극대화 : 한 장치에 여러 가지 기능이 있는 경우 경제적 득이 있을 수는 있지만 어느 한 기능의 고장이 장치 전체의 고장을 유발한다면 다른 모든 기능의 유효성까지 마비시키는 결과를 초래하게 되므로 이러한 일이 발생하지 않도록 적절하게 기능이 설정되어야 한다.

이러한 원칙 하에 물리적 요소들에 대한 기능 및 품성 정의가 이루어지면 해당 서브시스템의 실제 구축 및 운영관리상의 비용대비 효과가 극대화될 수 있을 것이다.

2) 물리 아키텍쳐 구상 모형 및 구축단위 정의

물리 아키텍쳐 모형이란 서브시스템별로 물리적 구성체계에 대한 기본 틀을 정의하는 것으로 〈그림 4〉과 같이 수정 보완된 AFD모형을 사용한다. 이것은 논리 아키텍쳐 모형에서 사용된 것이나 이 경우는 상호연계가 정보흐름 뿐만이 아니라 제어흐름까지 포함하며 또한 구성요소들이 논리수준의 서브시스템이 아니라 그것을 구성하는 물리적 장치 수준의 요소들을 의미한다. 각 장치 수준에서의 물리적 요소체계는 사용기술에 따라 또는 제공기능에 따라 여러 가지 형태로 제품화될 수 있으며, 이러한 제품 수준에서의 차별화는 구축단위(EP:Equipment Package)를 도출 정의함으로써 실현될 수 있다. 〈그림 4〉에서 점선 박스는 서브시스템의 경계선으로 서브시스템의 내부와 외부를 구별한다. 외부로 연결된 구성요소는 논리 아키텍쳐에서 구상된 타 서브시스템과의 연계로서 물리 아키텍쳐 수준에서는 그 서브시스템의 내부에서 해당되는 물리적 구성요소로 표현된다.

물리 아키텍쳐 구성요소는 논리 아키텍쳐에서 논리적으로 정의된 서브시스템을 실제 물리적 시스템으로 구현하기 위한 물리적 수준에서의 구성요소(Building Block)라고 할 수 있다. 구성요소는 각각의 위치와



〈그림 4〉 물리적 아키텍쳐 모형

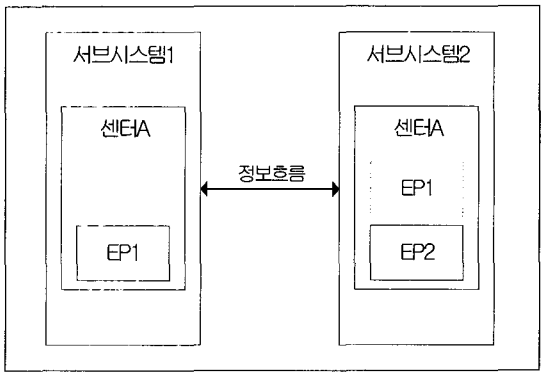
기능 및 역할에 따라 센터형, 도로장치형, 차량장치형, 여행자장치형 등 4가지 유형으로 분류할 수 있다. 센터형 구성요소(Center Type Components)란 각 서브시스템의 핵심요소로서 도로장치형 구성요소 등 타 구성요소를 제어하고 그것을 통해 필요한 정보를 수집하고 이를 분석·처리한다. 도로 장치형 구성요소(Roadside Equipment Type Components)란 신호기, 표시기, 검지기, 톨(Toll), 등 도로상에 분산되어 있는 인프라구조로서 차량 및 센터 구성요소와 연계되어 도로 및 교통에 대한 정보를 수집·제공하고 도로 및 교통을 제어하는 등의 기능을 수행한다. 차량 장치형 구성요소(Vehicle Equipment Type Components)란 GPS수신장치, 태그(Tag), 통신장치 등과 같이 각종 차량(승용차량, 대중차량, 화물차량, 긴급차량, 등)에 장착되는 구성요소로 노변 또는 센터와 연결된다. 여행자 장치형 구성요소(Traveller-Terminal Type Components)란 차량장치형 구성요소와 달리 PC, 전화, 휴대폰과 같은 개인 단말기나, 노변 단말장치(kiosk)와 같이 출발 전 혹은 운행 중 이용되는 사용자 단말장치를 의미한다.

구축단위(EP:Equipment Packages)란, 서브시스템의 각 물리적 구성요소들이 갖추어야 하는 제품 수준의 기능적 구현단위로 해당 구성요소의 기능적 부분으로 볼 수 있고 이는 기능의 패키지화를 의미한다. 이러한 기능의 패키지화를 통행서 관련 장치들이 새로운 기술로 업그레이드(Upgrade)되는 것을 쉽게 함으로써 새로운 기술에 대한 수용성을 제고하고(Open for New Technology) 또한 제품 수준에서의 재사용성(Reusability in Product Level)을 극대화할 수 있게 된다.

이러한 모형으로 물리 아키텍쳐가 구상되면 그것을 통해서 논리 아키텍쳐에서 도출 정의된 서브시스템들을 실제 점진적으로 구현하는 시나리오가 얻어진다. 즉 논리 아키텍쳐에서 도출된 서브시스템들에 대한 점진 구현을 위한 구현순서 및 임의 서브시스템을 구현함에 있어 함께 구현되어져야 하는 타 서브시스템들 즉 구현 범위를 알 수가 있게 된다. 구현순서와 범위는 논리 아키텍쳐에서 정의되는 정보흐름과 물리 아키텍쳐에서 구성요소들에 설정되는 구현단위(EP)에 대한 분석을 통해서 얻어진다. 논리 아키텍쳐에서 두 서브시스템이 흐름정보로 연계된 경우는 그 흐름정보의 용도 및 그것의 중요성 내지는 필수성에 대한 판단에 의해서 순차적 구현 또는 그것에 의존하지 않는 독립적 구현을 할 것인가가 정해지고, 두 서브시스템이 같은 센터/장치 유형을 구성요소로 포함하면서 그 속에 동일한 구현단위(EP)를 내포하는 경우는 그 중 어느 하나가 다른 하나의 구현단위를 공유형태로 재사용할 것인가 아니면 중복형태로 재사용할 것인가의 결정에 따라 다음과 같이 4가지 구현 시나리오가 가능해 진다:

[경우-1] 순차적 & 상호의존적 구현관계

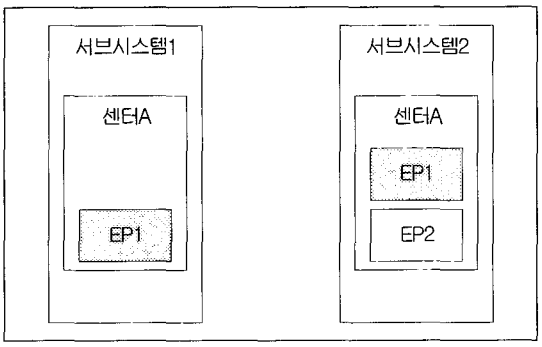
두 개의 서브시스템이 〈그림 5〉와 같이 동일한 물리적 요소(센터 A)로 구성되며 그것의 기능이 한 곳에서는 한 개의 구현단위 EP1만으로 정의되고 다른 한곳에서는 두 개의 구현단위 즉 (EP1+EP2)로 정의되면서 EP1을 공유형태로 재사용하는 경우이다. 이러한 경우는 순차적이고 상호독립적인 구현이 가능하게 된다. 즉, 두 개의 서브시스템 중 우선적으로 서브시스템 1을 구현하고 추후에 동일한 센터 내에 EP2를 새로이 추가 설정함으로써 고급기능 또는 새로운 기능을 가진 서브시스템 2를 구현하게 되는 것이다. 동일한 센터 A를 가지는 서브시스템 1과 서브시스템 2에서 서브시스템 2의 EP2는 먼저 구현된 서브시스템 1에 속한 EP1의 기능을 공유하기 때문에 EP1은 새로운 서브시스템 2에서는 전혀 고려할 필요가 없다. 이러한 관계를 도시하면 〈그림 5〉와 같다. 이러한 두 서브시스템간의 연계 관계는 논리 아키텍쳐 수준에서는 정보흐름으로 물리 아키텍쳐 수준에서는 정보흐름 또는 필요시 제어흐름으로 표현되어야 한다.



〈그림 5〉 순차적 & 의존적 관계

[경우-2] 순차성을 가지나 병행적 & 상호 독립적 구현관계 : 구축단위의 중복 허용

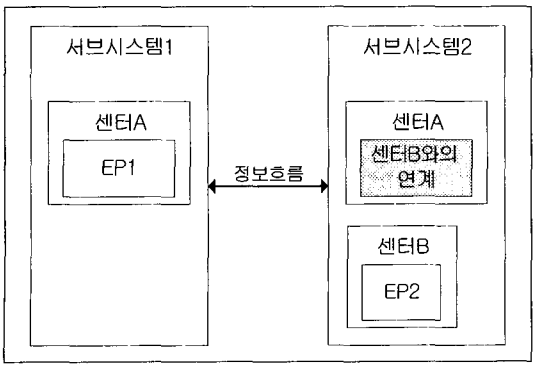
두 서브시스템이 경우-1과 같이 순차성을 가지도록 정의되어 있지만, 〈그림 6〉과 같이 서브시스템 1과 서브시스템 2의 비순차적이고 독립적인 구현도 가능하도록 EP1이 중복 설정됨으로서 EP1이 공유되지 않고 중복적으로 재사용 되는 경우이다. 이러한 결정은 전형적인 시스템 공학적 문제로써 대상 서브시스템의 구현 시급성과 EP1의 중복 비용에 대한 비용대비 효과 분석에 의해서 이루어져야 한다. 이 경우는 물리 아키텍쳐 AFD에서 두 서브시스템간의 정보흐름·제어흐름에 의한 연계가 불필요하게 된다.



〈그림 6〉 순차적 & 독립적 관계

[경우-3] 순차적 & 상호의존적 구현관계

서브시스템 2는 〈그림 7〉과 같이 먼저 구현되는 서브시스템 1의 센터 A에서 흐름정보를 받아서 센터 B에서 분석 처리하는 경우로서 센터 A에서 센터 B로의 연계를 전제로 하는 경우이다. 따라서 서브시스템 2가 구축되는 시점에 서브시스템 1의 센터 A에서도 정보연계 기능이 추가로 이루어져야 한다는 상호의존



〈그림 7〉 순차적 & 의존적 관계

성을 가지게 된다. 이를 도시하면 〈그림 7〉과 같고 여기서 서브시스템 2에 속하는 구현단위(EP) 중 "센터 B와 연계"라는 EP는 서브시스템 2의 구현을 위해 서브시스템 1의 센터 A가 새로이 가져야하는 기능으로서 서브시스템 2의 구현 시점에 구현되어야 한다는 점에서 서브시스템 2에 해당하는 EP로 표현한다. 이러한 연계 관계는 논리 아키텍쳐와 물리 아키텍쳐 모두에서 정보흐름으로 표현된다.

[경우-4] 상호 독립적 구현관계

논리 아키텍쳐에서 정의된 두 서브시스템간에 흐름정보에 의한 연계가 있으나 그것이 중요한(Critical) 연계가 아니라고 판단되거나 전혀 연계가 없는 경우이며, 또는 서로 상대방 서브시스템을 종단부로 연계하는 경우이다. 이러한 경우에는 해당되는 두 서브시스템 중 어느 것이 먼저 구현되어도 관계가 없으며, 만약 나중에 오는 서브시스템의 구현 시에 먼저 구현된 서브시스템과의 연계(연결)가 필요하다고 추후 판단되는 경우에는 양쪽 센터 모두에서 상대방을 연계하는 EP를 갖게 함으로서 가능해 질 수도 있다.

예를 들어 ITS관련 물리 아키텍쳐 수준의 서브시스템 중 다음과 같이 정의되는 두 개의 서브시스템을 생각해보자.

서브시스템1 : 정적주행안내 서브시스템 ::
=(차량단말장치)
where 차량단말장치 ::
= (EP1 : 정적주행안내기능)

서브시스템2 : 동적주행안내 서브시스템 ::
=(차량단말장치+정보센터)
where 차량단말장치 ::
=(EP1+EP2 : 동적정보수집처리기능)

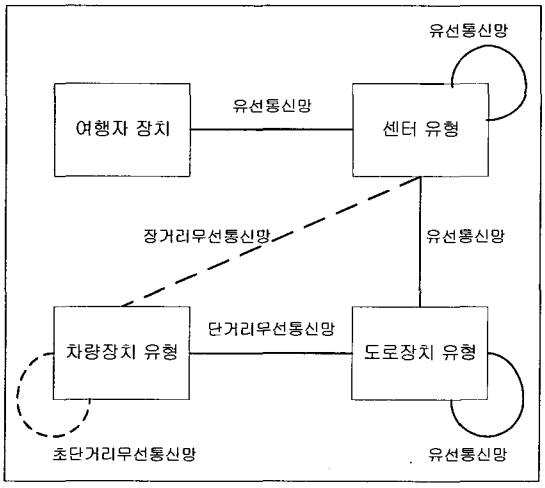
두 개의 서브시스템을 구성하는 물리적 구성요소로서의 차량단말장치에 대한 기능정의에 각각에 EP1이 포함되어 있는 경우로 이를 실제 물리적으로 구현하는 경우에 두 장치를 차별화 할 것인가 동일시 할 것인가의 문제가 생긴다. 차별화 한다는 것은 EP1으로 정의된 단말을 정적주행용 단말장치로 그리고(EP1+EP2)로 정의되는 단말장치를 동적주행용 단말장치로 다르게 제품화함을 의미하며, 이 경우는 EP1이

중복적으로 재사용된다(경우-2). 동일시한다는 것은 같은 제품에서 기능을 추가함을 의미하며, 이 경우는 EP1이 공유적으로 재사용된다(경우-1). 두 경우 모두에서 제품 수준에서 EP1이 재사용(Reuse)되므로 경제적 효과는 물론 조기구현의 효과를 얻을 수 있다.

5. 통신 아키텍쳐

통신 아키텍쳐란 물리 아키텍쳐에서 구성요소간의 연계를 위한 통신 수준에서의 기본 틀을 의미하며 이는 〈그림 8〉과 같은 시스템 연계도(AID:Architecture Interconnect Diagram)로 모형화 될 수 있다. 이는 각 구성요소가 물리적으로 설치되어 운영될 때, 각 구성요소간의 실제적인 정보흐름 또는 제어흐름을 구현하는 통신방식을 나타낸다. 우리 나라 국가 ITS를 위한 통신 아키텍쳐는 〈그림 8〉과 같이 유선통신망, 무선통신망(장거리/단거리), 차량간 초 단거리 무선통신방식 등으로 구성된다. 예를 들면, 도로장치형 구성요소에서 검지된 교통정보는 유선통신망을 통하여 각종 교통관제센터로 전달되며, 센터에서 분석·처리된 정보는 광역무선통신망을 이용하여 여행자장치형 구성요소와 차량장치형 구성요소로 전달될 수가 있다.

ITS를 위한 물리 아키텍쳐에서의 구성요소간에 사용 가능한 통신방식은 크게 다음과 같이 4가지 유형으로 대별된다.



〈그림 8〉 통신아키텍쳐(AID)

1) 광역 무선 통신(Wide Area Wireless Communication)

넓은 지역에서 정보를 전송하기 위한 셀 전용 무선 전송방식으로 고정된 지점과 움직이는 사용자(자동차) 사이에 사용되는 통신 방식이다. 음성과 데이터의 두 가지 전송이 모두 가능하며, 단방향 방송식(broadcast) 전송방식을 포함한다. 예로는 PCS망, 특수 모빌 라디오 등이 있고 FM 부전송 방식도 가능하다.

2) 단거리 전용 무선 통신(Short Range Wireless Communication)

가까운 거리 상에 있는 움직이는 사용자와 고정된 기지국에서 사용되는 단거리 전용무선전송방식이며 (5~100feet 이하), 자동차가 톨 부스를 통과할 때 사용되는 통신방식이다.

3) 차량간 초단거리 무선통신(Vehicle to Vehicle Communication)

차량간 통신을 위해(AVHS관련 데이터 전송 시) 사용되는 방식으로 높은 데이터 비율, 낮은 에러율의 특성을 지니며, 자동차간 라디오 송수신기 시스템이 사용된다.

4) 유선 통신(Wireline Communication)

동축케이블, 광학섬유 등의 유선을 이용한 두 고정된 지점간 통신방식이다.

Ⅳ. 결론

본 논문에서 제시하는 ITS를 위한 아키텍쳐 모형과 방법론을 적용하여 실제로 구상되는 아키텍쳐는 우선 논리 수준에서 ITS란 거대 정보시스템의 응용대상인 교통·운송 영역의 도메인 공학적 분석 즉, 관련 법·제도·조직에 대한 공학적 분석은 물론 관련 기술 영역으로서 교통공학, 도로공학, 차량공학, 물류공학 등의 차원에서 분석이 이루어져야 한다. 다음은 물리 수준에서 시스템 자체의 효율성과 합리성 등을 위한 정보공학적 분석이 이루어져야 한다. ITS는 구현 범위가 초 대규모적이고 또한 제공 서비스 측면, 관련 법·제도 조직 측면, 필요 기술 측면에서 매우 복잡 다양한 체계이다. 따라서 이를 점진적으로 합리적으로 구현하기 위해서는 구체적인 실제 개발 구축에 앞서 이러한 다차원 공학적 분석을 통한 전체 시

스템에 대한 통합적 아키텍쳐를 구상하는 것이 필수적이다. 만약 구상된 아키텍쳐를 기본 틀로 하지 않는 경우, 막대한 재원의 손실은 물론 우후죽순 격으로 개발 구축됨으로 말미암아 구축된 시스템들이 조화롭게 통합되기는커녕 자칫 서로 상충되어 치명적 혼란이 야기되거나 중복에 의한 막대한 경제적 손실을 초래하며, 서비스의 사각지대로 인한 ITS의 효과 저해가 발생될 소지가 크다 하겠다.

한편, 최근 들어 ITS를 위한 국제 표준화기구(ISO TC 204)에서 이러한 필요성에 따라 국제표준으로서의 ITS 아키텍쳐를 구상·발표하였다. 그런데, 발표된 국제표준은 정보기술의 발전 추세에 따라 재사용성과 유연성과 자동화성 등을 강점으로 가지는 객체지향적 아키텍쳐 모형과 기법을 사용하였다. 그러나 우리의 경우는 미국이나 유럽처럼 기존의 프로세스 지향적 방식으로 아키텍쳐를 구상하였기 때문에 향후 국제적 호환성의 문제는 물론 불가피한 새로운 기술로의 전이 문제 등 여러 가지 문제가 야기될 수 있다 하겠다. 따라서 우리의 경우 이를 어떻게 수용해 갈 것인가에 대한 연구가 시급하게 이루어져야 할 것이다.

참고문헌

1. 국토개발원, "국가 ITS 아키텍쳐 확립을 위한 연구, 개발사업(1)", 1998. 12.
2. 국토개발원, "국가 ITS 아키텍쳐 확립을 위한 연구. 개발사업(2)", 2000. 8.
3. L. Bass, P. Clement, and R. Kazman, "Software Architecture in Practice", Addison Wesley Inc., 1998.
4. H-E. Eriksson and M. Penker, "UML Toolkit", John wiley & Sons Inc., 1998.
5. D. J. Hatley, and I. A. Pirbhai, "Strategy for Real-Time System Specification", Dorset House, 1987.
6. ISO/TC204/WG1/N300, "Transport Information and Control Systems(TICS) : Reference Architecture Tutorial", Oct. 1, 1997.
7. I. Jacobson, "Oject Oriented Software Engineering : A Use Case Driven Approach", Addison-Wesley, 1992.
8. R. S. Pressman, "Software Engineering(4th Edition)", McGROW-Hill, 2000.
9. A. N. Sinha, F. L. Willingham, and M. Hermes, "An Architecture Framework for DSSs in Air Traffic Management", System Engineering, Vol. 4, No. 1, John Wiley & Sons Inc., 2001.
10. J. L. Smith and A. M. Schoka, "The ISO TICS Reference Architecture-An Extendable Specification", ISO TC204 Technical Report, 2000.
11. US DOT-FHA, "ITS Architecture: Physical Architecture, Joint Architecture Team(Loral Federal Systems & Rockwell International)", Oct. 1995.
12. US DOT-FHA, "ITS Architecture: Implementation Plan, Joint Architecture Team (Loral Federal System & Rockwell International)", Oct. 1995.
13. US DOT-FHA, "ITS Architecture: Logical Architecture, Joint Architecture Team(Loral Federal System & Rockwell International)", Oct. 1995.
14. I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Software Development Process", Addison Wesley, 1999.

♧ 주 작 성 자 : 백인섭
♧ 논문투고일 : 2001. 4. 11
논문심사일 : 2001. 5. 23 (1차)
2001. 9. 6 (2차)
2001. 11. 6 (3차)
2001. 11. 19 (4차)
심사판정일 : 2001. 11. 19