견실한 시스템 아키텍쳐 개발 지침

A Development Guide for Robust System Architecting

유 일 상* 박 영 원**
Il Sang Yoo, Young Won Park

ABSTRACT

The term system architecting(SA) is widely used in systems engineering area, however the explicit meaning of the term varies considerably from person to person depending on his expertise and viewpoints. In this paper, we surveyed the range of variations in definitions and present the summary. Additionally, the definition of robust system architecting is suggested along with the associated system architecture guides that can be followed during an architecture development. This guide exposes various viewpoints of architecture such as art side, requirement specification, system view, principles, tools and environment.

주요기술용어 : Systems Engineering(시스템공학), Robust System Architecture(견실한 시스템 아키텍쳐), Systems Architecting(시스템 아키텍팅), System View(시스템 관점)

1. 서 론

1760년의 산업혁명 이후 기하급수적인 기술 발전으로 인해 최근에 와서는 기술적으로 복잡한 시스템이 사회적으로 요구되기 시작하고 있다. 예를 들어, 보잉 777의 경우 13만여 개의 부품으로 구성되고 전차의 경우도 5만여 개의 부품으로 구성된다. 또한 반도체 칩의 경우 부피는 작지만 요소간 상호 연결이 수십만 개일 정도로 복잡하다. 기술적 복잡성과 규모 측면 때문에 단순히 일부 학문 분야만으로는 이러한 복잡한 시스템을 개발할 수 없어졌다. 동시에 다양한 고객의 요구, 시스템 수명의 단축, 시장 경쟁의 심화 등으로 연구 개발 환경이 급속히 변화하고 있다.

* 아주대학교 시스템공학과 박사과정
** 아주대학교 시스템공학과 교수

1950년대 말부터 고객의 요구를 반영한 복잡하고 규모가 큰 시스템을 개발하기 위해 대두된 시스템공학은 최근 들어 점점 복잡해지는 모든 시스템을 개발하는 데 필요한 시스템 설계 및 관리의 방법론이 되었다.[16] 국제시스템공학학회(INCOSE)에서는 시스템공학이란 시스템의 성공적인 구현을 위한 다학제간의 접근법이자 방법으로 정의한다. 다시 말하면 시스템공학은 시스템에 관련된 복잡한 문제들을 해결하기 위해 이 문제에 관련된 모든 이해관계자(Stakeholder)의 요건(Requirement)을 만족시키기 위한 다분야간의 종합적 접근법이다. 여기서 다분야간이라는 말은 시스템공학은 한 전문분야에 의해서 달성될 수 없고, 각 공학 분야 및 비공학 분야의 전문가와 기능조직으로부터의 사람들이 모여서 그들의 기술과 지식을 통합하여 시스템 개발 시의 문제해결이 필요한 모든

대상에 효율적이고 효과적으로 그 해법을 찾아냄을 말한다. 이러한 시스템공학은 고객의 수요와 요구되는 기능과 성능을 개발 초기에 정의하고 문서화한 후, 비용과 일정, 생산, 시험, 운영, 훈련, 정비지원, 폐기를 고려한 시스템 합성(Synthesis)과 유효성 검증(Validation)을 수행하는 것에 초점을 두고 있다. 시스템공학은 모든 학문과 특수공학 그룹들을 개념, 생산, 운영으로 이어지는 구조화된 개발 과정을 형성하는 팀의 노력으로 통합하고 고객의 수요를 만족하는 고품질의 제품을 제공하기 위해 업무적 수요와 기술적 수요를 모두 고려한다.[2,3,6] 시스템 공학적 접근법을 적용할 경우, 전통적인 시스템 개발 방식에 비해 제품 개발 기간의 60% 단축, 설계변경 요청 건수의 50% 감축, 재설계와 재작업 업무량의 75% 절감, 제조비용의 40% 절감 등의 구체적인 효과가 사례로 보고되고 있으며 이러한 사례를 통해 시스템공학은 보다 경쟁력있는 제품 개발, 외부 소비자의 기대와 요건에 부응할 수 있는 과정의 정립 등의 일반적 효과가 얻어진다는 견해로 이어진다.[17] 미국의 경우 국방과 항공 분야와 더불어 수송 시스템, 통신 시스템, 에너지 시스템, 정보 기술 시스템 등의 상용 시스템의 개발 사업에 그 적용이 확대되고 있지만[1] 국내에서는 국방 사업과 고속/경전철 사업,[18] 전자교환기[20] 등의 정부 프로젝트와 항공 분야[19]에서만 제한적으로 응용의 중요성이 인식되고 있다.

시스템 아키텍팅이라는 용어는 30여 년 동안 시스템공학 분야에서 널리 사용되어 왔다. 이것의 결과물인 시스템 아키텍처는 시스템 개발을 위한 중요한 틀을 제공하며 시스템 특성들의 70% 이상을 결정한다. 하지만 많은 문헌의 저자들은 개개인의 경험, 관점과 학문 분야에 따라 시스템 아키텍팅을 구현하려는 사람들에게 여러 가지 정의와 범위를 제시하기 때문에 이 용어에 대한 이해보다는 혼란을 가중시키고 있다.[15] 따라서 시스템 아키텍팅에 관한 기존 문헌을 고찰하고 명확한 정의를 도출하는 것이 필요하다. 또한 시스템의 기술적 복잡성과 불확실성이 커짐에 따라 고객이 기대하는 시스템과 실제 개발된 시스템사이에 큰 차이가 생기고 이 차이를 줄이기 위한 단계적 접근 방법으로 시스템 아키텍팅을 수행한다. 따라서 시스템 아키텍팅을 잘 하기 위해 시스템 설계자가 고려해야 할 사항들을 파악할 필요가 있다

따라서 본 연구에서는 먼저, 시스템 아키텍팅에 대한 정의, 역할, 중요성과 시스템공학과의 관계에 대한 이해를 제공하고자 한다. 두 번째로 고객이 기대하는 시스템과 개발된 시스템사이의 차이를 최소화하기 위한 견실한 시스템 아키텍팅의 개념을 제시하고 이를 위해 시스템 수준 설계자가 고려해야할 지침들을 제시하고자 한다.

2. 시스템 아키텍팅에 관한 고찰

많은 기존 문헌들을 고찰한 결과 저자의 개인적 경험, 관점과 학문 분야에 따라 시스템 아키텍팅의 정의, 역할, 중요성과 시스템 공학과의 관계 측면에서 다양하게 진술하고 있다. 따라서 본 장에서는 기존의 문헌을 고찰하여 시스템 아키텍팅의 제반 사항을 정리하여 이해를 제공한다.

2.1 용어 정의

먼저 시스템 아키텍팅을 이해하기 위해 관련 용어들에 대해 INCOSE(International Council on System Engineering) SAWG(System Architecting Working Group)의 정의를 아래와 같이 정리하였다.[13]

- 시스템 아키텍쳐(System Architecture): 시스템 구성요소, 인터페이스, 프로세스, 제약사항, 및 거동에 대하여 정의한 기본적이고 단일화된 시스템

구조(골격)로 시스템 개념 설계의 산출물

- 시스템 아키텍팅(System Architecting): 시스템 아키텍쳐의 적절한 구현을 정의, 유지보수, 개선, 및 보증하는 활동
- 시스템 설계자(System Architect): 시스템 아키텍팅을 책임지는 사람, 팀 또는 조직
- 시스템 설계(System Design): 실현 가능한 시스템 요소, 하부 요소, 및 인터페이스들의 조합으로 상세 설계 활동의 요건 시방서

2.2 시스템 아키텍쳐의 역할

시스템 아키텍쳐는 이해관계자의 요건으로부터 개발될 시스템의 모든 관점들을 보여주는 틀로서 아래와 같이 중요한 역할을 한다.

① 시스템 개발의 초기 단계에서 이해관계자의 요건 분석, 기능 분석, 합성, 시스템 해석 및 최적화를 통해 생성되어서 시스템 개발의 후속 단계인 설계 단계와 구현 단계에 필요한 포괄적인 시방서로서 사용된다.

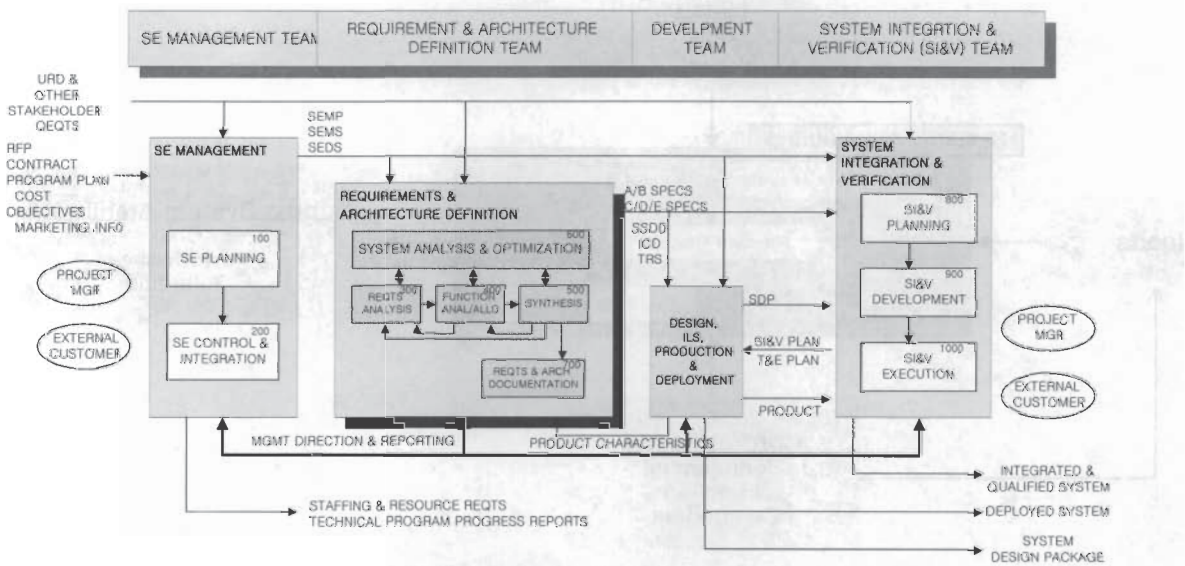
② 이해관계자와 개발자 사이의 의사소통 매개로서 사용된다.

③ 시스템 개발의 초기에 문제점들을 파악하고 여러 대안들을 절충하는 방법을 제공함으로써 발생될 위험을 감소시키는 데 사용된다.

④ 시스템의 비용, 신뢰성, 가용성과 유지보수성을 추정하고 시스템 개발의 초기에 WBS(Work Breakdown Structure)를 개발하는 데 사용된다.

2.3 시스템 아키텍팅의 중요성

시스템 아키텍팅을 통해 요건들을 분석한 후, 요건 시방서가 작성되면 이후 시스템 개발과정에서 기능 속성의 90%, 일정관련 정보의 80%가 고정되고 품질 속성의 70%와 생산비용의 60%가 결정된다. 따라서 개발된 시스템 아키텍쳐는 시스템의 기능과 성능 및 시스템 개발의 일정과 비용의 대부분을



〔그림 1〕 Systems Engineering Process

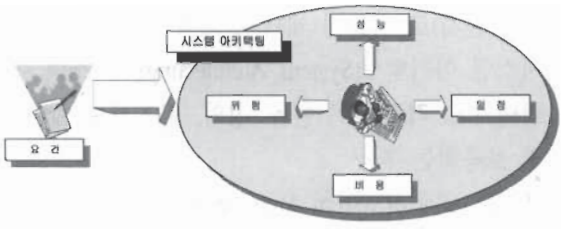
결정하므로 개발 초기에 시스템 아키텍팅을 잘해야 시스템 개발 프로젝트를 성공으로 이끌 수 있다.

2.4 시스템공학과 시스템 아키텍팅

일반적으로 시스템공학 프로세스 상에서 시스템 아키텍팅은 개념 설계 단계에서 고객의 요건을 분석하는 과정에서 시작된다. James N. Martin이 제시한 시스템공학 프로세스[10]는 그림 1과 같고 여기서 시스템 아키텍팅은 요건 및 아키텍쳐 정의 하부과정에서 수행된다. 이 요건과 아키텍쳐 정의 하부과정을 통해 질서 정연하고 반복적으로 문제를 정의하고 해법을 개발할 수가 있다. 또한 요건 및 아키텍쳐 정의 하부과정은 고객의 요건을 기술요건과 시스템 아키텍쳐로 변환하고 문서화하는데 관련된 개념 설계 활동을 포함한다.

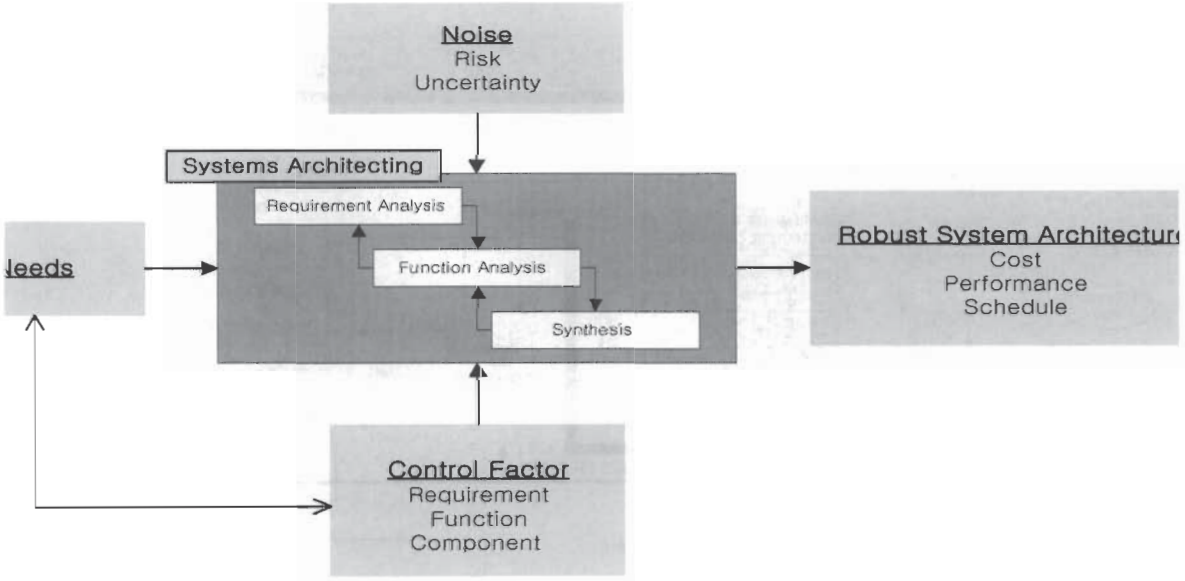
3. 견실한 시스템 아키텍쳐 개발 지침

시스템 아키텍팅은 시스템의 성능, 비용, 일정과



〔그림 2〕 시스템 아키텍팅의 경쟁요소

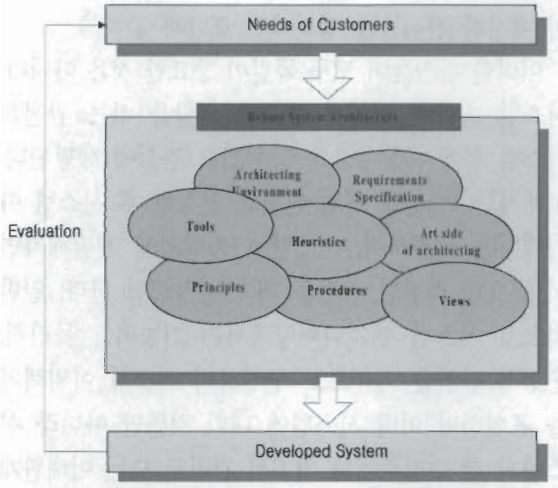
위험 부담을 추정 및 분석하고 시스템 연구, 개발, 생산, 관리를 위한 지침을 제공하기 위해 고객의 요건을 반영한 시스템을 표현하거나 추상화하는 것이다. 특히 시스템 아키텍팅 동안에 시스템 아키텍쳐가 고객의 요건을 정확히 반영해야 한다. 만약 개발 초기에 잘못된 요건이 반영되면 그 개발 프로젝트는 성공할 수 없다. 따라서 시스템 개발 프로젝트의 성공 여부는 얼마나 견실한 시스템 아키텍쳐를 개발하느냐에 달려 있다고 해도 과언이 아닐 것이다. 따라서 본 장에서는 견실한 시스템 아키텍쳐 개념과 견실한 시스템 아키텍쳐 개발을 위해 시스템 설계자가 고려해야 할 사항을 제시한다.



〔그림 3〕 Robust Systems Architecting

3.1 견실한 시스템 아키텍쳐

단순한 시스템을 개발하는 경우 시작품(Prototype)을 만들어서 이해관계자와 개발자 사이에 의사소통의 매개로서 이들간의 의견을 조율하고 위험부담을 경감할 수 있었으나 기술적으로 복잡한 시스템을 개발하는 경우에는 서로의 생각하는 바를 잘못 의사소통하여서 이해관계자가 기대하는 시스템과 개발자가 실제로 개발된 시스템사이에 큰 차이가 생긴다. 통계학적 의미에서 견실성(Robustness)은 설계 대안들에 영향을 주는 불확실한 여러 요소들 속에서 파라미터 설계를 통해 최적의 조건을 찾고 그 최적조건에서 산포에 많이 영향을 주는 인자들을 조절하여 허용차가 최소가 되는 조건을 견실하다고 정의한다.[19] 시스템엔지니어링 측면에서 견실한 시스템 아키텍쳐는 시스템 개발 과정에서 발생하는 기술, 비용, 일정 및 관리적 위험부담과 불확실성에 둔감하고 이해관계자가 원하는 시스템과의 차이가 최소화된 대안의 시스템 골격으로 정의한다. 이렇게 하기 위해 요건, 기능, 컴포넌트와 같은 제어 인자들을 요건 분석, 기능 분석과 합성을 통해 조절해야 한다. 이때 항상



〔그림 4〕 견실한 시스템 아키텍팅 지침

이해관계자와의 의사소통을 가져야 한다. 이러한 시스템 아키텍쳐를 개발하기 위해 시스템 설계자는 고려해야할 많은 사항들이 있으며 본 장에서 이를 위한 지침을 제공한다.

3.2 예술적 측면 이해

시스템 아키텍팅의 주요 문제점은 복잡한 시스템 요건들을 모두 식별해내고 경험, 통찰력, 창의성에 의해 이것들을 정확히 표현하는 데에 있다. 시스템 아키텍팅 시에는 과학적 측면보다 예술적인 측면이 더 중요함을 인식해야 한다. 실제로 엔지니어링 문제에서 예술적 측면이 종종 가장 중요하다. 복잡한 시스템을 개발하거나 개선할 경우에 발생하는 가장 심각한 문제점은 예술적 측면에서 초래된다.[11,12]

특히 전례가 없는 시스템을 개발하는 데, 예술적 측면은 더욱 중요한 역할을 한다. 또한 시스템 아키텍팅의 예술적 측면은 과학적 측면이 부족한 경우에서 보완하는 역할을 한다. 따라서 시스템 아키텍팅 시의 과학적인 측면뿐만 아니라 예술적인 면을 이해해야지만 견실한 시스템을 개발할 수 있다. 그러기 위해 다음의 사항들을 고려해야 한다.

① 시스템 개발이 성공하기 위해서는 최상위 시스템 개념 모델에서 점차 하부로 내려가면서 가치, 안전성, 가격성, 정치성, 환경 영향, 공중 위생과 심지어는 국가 안보와 같이 측정할 수 없는 요소에 대한 인식들이 포함되어야 한다.

② 시스템 요건들 중 무엇이 중요한 지를 판단하는 데에는 가정이 항상 존재하고 또한 대부분 각 시스템 설계자는 시스템 요건을 분해하는 과정에서, 자신의 경험, 편견과 관점을 기초로 의사 결정을 하여 시스템을 표현한다. 따라서 시스템 설계자는 시스템 아키텍팅 초기에 가정에 대한 이

유와 분해의 근거를 이해해야 한다.

③ 자동화된 좋은 도구를 사용해서 인간의 제한된 이해력을 극복해야 한다.

④ 복잡한 시스템을 아키텍팅하는 데에는 많은 개념 개발, 문제 해결과 의사 결정 활동들이 수행된다. 이 경우 경험적 사고(Heuristics)를 이용해야 한다. 특히, 명확히 정의되지 않는 요건이나 정량화가 어려운 요건들을 다룰 때는 경험적 사고가 아주 유용하다.

3.3 정확한 요건 시방서

개발 문제의 명료한 정의를 담당하는 좋은 요건 시방서에서 견실한 시스템 아키텍처가 나올 수 있음은 주지의 사실이다. 따라서 견실한 시스템 아키텍처를 개발을 위해 요건 시방서는 다음의 성질을 가져야 한다.[14]

① 일관성(consistency): 시스템의 부분적인 지식으로 나머지 부분을 예측할 수 있도록 일관성이 있어야 한다.

② 직교성(orthogonality): 각각의 독립적인 기능들은 시방서에서 따로 분리되어 기술되어야 한다.

③ 타당성(propriety): 시스템의 필수적인 요건에 적당한 기능만 채택되어야 한다.

④ 절약성(parsimony): 구조적 설명에서 어떠한 기능들도 다른 형태로 반복되지 말아야 한다.

⑤ 투명성(Transparency): 구현과정에서 도입된 기능들을 사용자가 명확히 알 수 있어야 한다.

⑥ 일반성(Generality): 기능이 도입될 때, 가능한 많은 목적에 사용될 수 있게 도입되어야 한다.

⑦ 개방성(open-endedness): 다른 방법으로 기능을 사용할 수 있는 자유가 주어져야 한다.

⑧ 완전성(completeness): 도입된 기능들은 기술적이고 경제적인 제한 범위 안에서 사용자의 요건과 욕구를 가능한 한 완전하게 만족시켜야 한다.

시방서에 기술된 각각의 요건들은 명확해야 하고, 이해할 수 있어야 하고, 정확해야 하고, 간결해야 하고, 관련된 다른 요건이나 문서로 추적 가능해야 하고, 구체적인 설계와는 독립되어야 하고, 검증 가능해야 한다. 이러한 요건을 기술할 때 용어의 정의가 먼저 개발되어야 하고 이를 복합문이나 부정문으로나 모호한 동사나 형용사를 사용해서는 안된다. 요건은 개발하려는 시스템을 주어로 입력과 출력 등을 목적어로, 실제 조건을 부사로, 마지막으로 동사를 기술한다. 동사의 경우 요건, 사실과 목적을 진술할 때 각각 분별되게 표현해야 한다.[3]

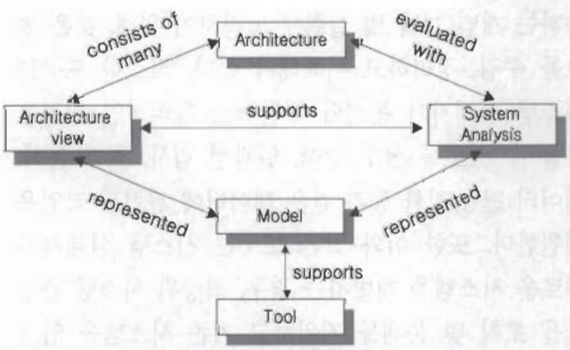
3.4 적절한 시스템 관점(System View)

시스템 관점이란 시스템의 어떤 하나의 특별한 면을 묘사하는 것이다. 각 관점들은 각각의 관심 사항에 초점을 맞추고 명쾌하고 능률적으로 표현하기 위해서 범위가 제한되어야 한다. 지금까지 여러 학자들이 제시한 관점들을 정리하면 표 1과 같다.[8]

이러한 관점들이 모두 똑같이 중요한 것은 아니다. 시스템 개발에 있어서 중요한 역할을 하는 관점은 시스템 설계자가 중요하게 여기는 관점에 해당된다. 설계자는 어느 한 개의 관점을 선정한 후 시스템 아키텍팅을 시작하며, 시스템 아키텍처를 개발해가면서 필요한 적절한 관점을 계속 선정해야 한다. 일반적으로 시스템 엔지니어링에서는 기능적 , 물리적, 운영적 관점을 기본으로 구성하여 시스템 아키텍처를 표현하며 이때 시스템에 대한 것뿐만 아니라 시스템의 각 수명주기와 거기서 관련된 모든 이해관계자를 반드시 고려해야 한다.

〔표 1〕 주요한 시스템 관점

관 점	관심 사항 및 범위
Form	무슨 시스템인가? 시스템의 실현
Objective/Purpose	고객과 사용자가 바라는 시스템의 역할
Function	시스템이 무엇을 하고 어떤 기능을 하는가?
Data	시스템 안에 있는 정보와 그 상호 관계
Managerial	시스템이 만들어지고 관리되는 과정
Application	시스템 안에 있는 구성 요소들의 실행
Scenario	시스템에 의해서 실행되는 것에 대한 거동적인 설명
Hardware	시스템의 물리적 구성 요소
User	시스템을 운영하고 사용하는 사람들과 관계된 모든 것
State/Mode	시스템의 운영 상태와 양상
Infrastructure	시스템의 하드웨어와 적용측면을 지원하는 하부구조
Facility	시스템 배치 시에 필요하는 공간



〔그림 5〕 아키텍쳐 관계도

시스템 관점과 시스템 분석, 모델, 아키텍쳐, 그리고 도구와의 관계는 그림 2와 같다.[4]

시스템의 모든 관점을 통해 시스템 분석을 수행할 수 있다. 시스템 분석은 시스템이 얼마나 시스템 목표나 요건을 만족시키는 가를 해석하고 판단하는 것이다. 각 시스템 관점과 시스템 분석은 한 개 또는 여러 개의 모델로 표현할 수 있다. 모델은 물리적 모델부터 데이터 모델, 문서, 수학적 모델 등 다양한 방법으로 여러 가지의 관점을 이용하여 시스템을 표현한다. 예를 들어, 4+1 모델, OMT모델, ADARTS 모델은 정보시스템을 표현하고, ARIS모델은 업무시스템을, Hatley Pirbhai(H/P)모델은 항공전자시스템을, MSA모델은 생산시스템을 표현한다.[9] 시스템 설계자는 많은 관점으로 표현된 통합된 모델을 가지고 보다 견실한 시스템을 설계할 수가 있게 된다.

3.5 원칙(Principles) 준수

시스템 설계자는 시스템 아키텍팅 시에 안정된 형식, 우선순위, 인터페이스, 협동에 중점을 두어야 한다.[8] 견실한 시스템 아키텍팅의 토대가 되는 원칙들은 아래와 같으며 복잡한 시스템 개발에서는 반드시 준수되어야 한다.[7]

① 시스템 접근법. 시스템은 어느 정도 상호 독립적인 구성요소와 이들간의 상호작용으로 이루어진 복잡한 집합체로서 시스템의 가치는 이러한 상호관계가 창출하는 시너지 효과에 있다. 시스템 거동은 구성요소가 가지고 있는 특성이 아니라 시스템 전체가 가지고 있는 시너지 효과의 특성을 말한다. 시스템 설계자는 시스템의 물리적 구조와 시스템 거동에 관심을 갖고 이해관계자와 개발자 사이의 의사소통에 초점을 두어서 가능하고 만족스런 시스템 개념 설계를 해야 한다.

② 목적 지향. 이해관계자와 빈번한 유효성 검증을 통해 시스템의 비전과 임무를 명확히 해야 한다. 시스템 아키텍팅은 기술혁신 또는 개발자의 우선순위보다는 이해관계자의 목적하는 바에 따라 좌우되기 때문이다. 시스템 설계자는 이해관계자와 개발자의 중재자로서 역할을 하며 이해관계자의

목적에 대해 기술적인 선택에 대한 책임과 가능성에 대한 결정을 통해 이해관계자가 만족할 만하고 가능한 시스템을 개발하는 데 목표가 있다.

③ 충실도 높은 모델링. 충실도가 높다는 것은 시스템 전체 수명주기 상에서 고려하지 않은 부분이 없도록 하는 것을 의미한다. 시스템 설계자는 시스템 전체에 대한 책임을 가지고 있기 때문에 여러 가지 관점들을 통합해야만 한다. 통합된 모델을 충실도 높게 개발하기 어려우며 점진적 개선을 통해 추상적인 부분을 감소시켜야 한다.

④ 경험적 사고. 경험적 사고란 문제를 푸는데 도움이 되는 정제된 교훈, 지침, 경고를 제시하는 간결한 문장을 말하며 아래 절에서 자세히 설명하였다. 시스템 설계자는 통찰력이 없이 의사결정을 해야 할 때 경험적 사고를 이용해야 하며 특히, 요건이 미비한 부분이나 없는 부분에는 더욱 그 이용이 필요하다.

⑤ 인증(Certification). 시스템 설계자는 이해관계자와 유효성 검증을 통해 자주 인증을 받아야 한다. 인증은 이해관계자가 개발자로부터 시스템을 인수받는 마지막 시험 단계에 해당된다.

3.6 경험적 사고(Heuristics) 이용

복잡한 시스템 아키텍팅을 설명하고자 하는 최근의 노력을 통해 경험적 사고들이 만들어졌으며 특히 개념설계 단계에서 시스템 설계자에게 많은 도움이 된다. 경험적 사고란 시스템 설계자들이 교육, 경험, 예제 등을 통해 얻은 교훈과 지침을 축적한 간결한 문장을 의미한다. 이러한 축적된 사고를 이용하는 법을 아는 것은 시스템 설계자에게 무엇보다도 중요하다. 1980년대 말 USC 대학원 과정에서 처음100개 정도의 경험적 사고를 만들었으며, 6년 동안에 1000개정도의 경험적 사고를 만들어서 널리 사용하고 있다.[12]

이러한 경험적 사고는 이해하기 쉽도록 여러 가지 방법으로 표현되며 간결하고 익살스러운 것일수록 기억하기 쉽다. 이를 효과적으로 표현하는 방법에는 일반적으로 서술적 표현과 규범적 표현, 두 가지가 있다. 서술적인 표현은, 예를 들어, 무슨 일이 잘되지 않을 때, 머피의 법칙이라고 하는 것이다. 규범적인 표현은, 예를 들어, 모든 것을 간결하게 처리하라는 KISS(Keep It Simple, Stupid) 법칙이 있다. 이러한 경험적 사고를 통해서 시스템 아키텍팅 시에 발생하는 상황을 파악하고 확인할 수 있으며 다음 단계를 위한 지침을 얻을 수 있다. 그러므로 시스템 아키텍팅 시에 이러한 경험적 사고를 이용하면 매우 효과적이고 능률적으로 시스템을 개발할 수 있다.

3.7 적절한 도구 선정

시스템 설계자는 시스템의 성능, 비용, 일정, 위험, 시장성, 규제, 조직 등을 고려해서 고객의 요건을 만족하는 개념 개발 및 실현에 노력하기 위해, 많은 정보를 수집, 정리하고 이해해야 한다. 적절한 도구는 시스템 설계자가 문제와 대안들에 창의적인 열정을 쏟을 수 있도록 정보 탐색, 일관성 검사, 효과 분석, 데이터 전송/전환 등과 같은 데이터에 관련된 작업을 지원한다. 또한 이와 같은 도구는 시스템 설계자가 새로운 시스템을 개발하는 경우, 최상위 시스템 관점들을 포착 및 분해를 지원하고 기존 시스템을 업그레이드할 경우, 많은 다른 관점에서 시스템 아키텍처를 빨리 이해하게 한다.[5]

시스템 아키텍팅 도구는 기본적으로 시스템 설계자에게 다음의 기능을 제공해야 한다.

① 시스템 관점 포착 및 문서화

② Data Flow, Mode/State, 기능성, HMI(Man-Machine Interface) 등의 시스템 특성을 검증할 수 있는 시

뮬레이션
③ 통합된 제품 개발 환경을 가능하게 하기 위한 요건 관리 도구와 설계 도구와의 인터페이스

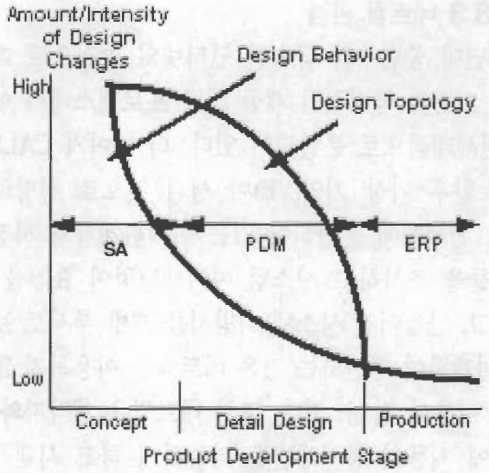
시스템 아키텍팅하는 데 적합한 도구를 선정하는 절차는 아래와 같다. 이 절차는 먼저 도구의 요구사항과 제약사항 목록을 개발하고 나서 최종적으로 선정하기 위한 대안의 도구들을 이 목록에 대해 평가하는 것이다. 좋은 도구란 절대적이지 않고 기능, 비용 등을 고려할 때 자신이나 조직에 적합한 것이다.

① 프로젝트 사명(Mission)과 목적 정의
② 필요한 도구의 기능 결정
③ 실제 고려사항과 제약사항 정의
④ 필수적인 요건 목록 개발(CRL)
⑤ 기존 도구와 벤더에 관한 정보 수집
⑥ CRL과 도구의 일치성 평가
⑦ 추천

3.8 시스템 아키텍팅 환경 구축

3.8.1 지식 DB 구축

시스템 아키텍팅의 예술적 측면을 앞 절에서 설명하였다. 즉 오랜 기간동안 시스템 아키텍팅에 참여한 시스템 설계자 개인의 경험과 지식으로만 남는 특징이 있다. 이러한 경험과 지식을 향후 새로운 시스템을 아키텍팅하는 데에 활용할 수 있어야 한다. 따라서 과학적 측면뿐만 아니라 예술적 측면을 가지는 시스템 아키텍팅 분야에서도 지식 경영을 구현해야 할 필요가 있다. 이를 위해 무엇보다 먼저 지식 DB를 구축해야 한다. 지식 DB 구축을 위해 묵시적 지식을 명시적 지식으로 변환해야 하며, 일단 지식 DB가 구축되고 나면, 이를 통해 시스템 아키텍팅 조직에서 지식이 재창조되는 이점이 생긴다.



〔그림 6〕 PDM/ERP와의 통합

3.8.2 PDM/ERP 시스템과의 통합

기존의 제품 개발 프로세스에서 부분적인 자동화의 결과, 정보의 중복과 부정확한 통제로 인해 부분적인 정보의 홍수가 발생하고 있다. 즉, 많은 시스템에서 발생한 데이터의 폭발적 증가, 변경이 용이한 설계도구의 사용으로 인한 잦은 설계 변경, 새로운 도구의 속도와 기능이 관리통제의 한계 초과 등의 결과를 초래했다. 따라서 부분적인 자동화의 비효율성을 극복하고 통합되고 일관성 있는 제품 개발을 위해, 개념 설계 단계에서 이해관계자의 요건 분석과 시스템 아키텍팅을 수행하는 도구들은 반드시 PDM 시스템과 ERP 시스템에 통합되어야 한다. 시스템 아키텍팅 지원 시스템은 제품의 아이디어 생성단계부터 기본 설계까지를 지원하고, PDM 시스템은 기본 설계에서 상세 설계(Bill of Materials 포함)까지를 지원하며, ERP 시스템은 상세 설계에서부터 제품생산 단계까지를 지원하여 향후 통합 제품 개발(IPD) 프로세스를 이룰 수 있게된다. 현재 개발 프로세스 전체를 포함하는 시스템은 존재하지 않으므로 이들과 연계할 수 있는 시스템의 개발이 필요하다.

3.8.3 네트워 환경

90년대 중반부터 급속한 인터넷의 발전으로 기업들은 기존의 국지적인 제품 개발 프로세스에서 이제는 전세계적으로 분산되어 있다. 더 나아가 CALS에서는 향후 가상 기업(VE)이 생길 것으로 전망하고 있다. 전세계에 분산되어 있는 이해관계자와 시장의 요건들을 조사하고 시스템 아키텍팅하여 요건을 분석하고, 만들어진 시스템 시방서를 개발 부서로 보내고, 설계하여 생산하는 것은 네트워을 이용하지 않고서는 이루어 질 수 없다. 또한 네트워은 계약자와 정부간에 제품의 동시개발을 가능하게 하고 시행착오를 감소시키며 개발 프로세스를 최적화한다. 따라서 시스템 아키텍팅 환경으로 네트워은 필수적이다. 이의 전제 조건으로 정보 교환의 표준 형식이 제정되고 준수되어야 한다.

4. 결 론

복잡한 시스템 개발 시에 시스템 아키텍팅은 시스템의 성능, 비용, 일정과 위험을 예측 및 분석하고 시스템의 연구, 개발, 생산, 관리를 위한 지침을 제공하기 위해 전체 시스템을 표현하거나 추상화하는 역할을 한다. 고객의 요건을 정확히 반영하는 시스템 아키텍쳐의 개발은 시스템 개발 프로젝트의 성공을 좌우한다. 따라서 견실한 시스템 아키텍쳐 개발이 필요하다. 견실한 시스템 아키텍처는 시스템 개발 과정에서 발생하는 기술, 비용, 일정 및 관리적 위험부담과 불확실성에 둔감하고 이해관계자가 원하는 시스템과의 차이가 최소화된 시스템 구조의 설계대안으로 새롭게 정의하였다. 이를 위해 시스템 설계자는 우선적으로 고객의 요건을 정확히 조사하여 명확한 요건 시방서를 작성하는 것이 무엇보다 중요하다. 명확한 요건 시방서를 토대로 고려되어야 할 시스템 관점을 정하고 체계적인 절차를 따라서 시스템 아키텍팅 한다. 이 과정에서 아키텍팅의 예술적 측면을 이해 보완하고 적절한 경험적 사고를 이용하며 원칙을 따라야 한다. 이와 같은 일련의 반복적인 절차를 지원하는 도구를 미리 선정하고 이를 위한 아키텍팅 환경이 되어야 한다. 따라서 시스템 설계자가 본 논문에서 제시한 측면을 고려해서 시스템 아키텍팅을 하면 보다 견실한 시스템 아키텍쳐를 개발할 수 있을 것이다.

향후 견실한 시스템 아키텍쳐의 개념을 구현하는 구체적인 방법 및 절차에 대한 연구가 수행될 것이다.

참 고 문 헌

1. Ascent Logic co., Introduction to RDD-100 Student Workbook, 1996.
2. Blanchard, B. S. and Fabrycky, W. J., Systems Engineering and Analysis, 3rd Edition, Chapter 1, 2, 3, 4, Prentice Hall, 1997.
3. Buede D. M. , The Engineering Design of Systems, John Wiley & Sons, 2000.
4. Frank, R. F., "System Architecture - A View Perspective", Proceedings of 8th INCOSE Symposium, 1998.
5. Hyer, S. A. and et al, "A Guide to CASE Tools in Support of System Architecting: Background, Capabilities, and Selection", Proceedings of 7th INCOSE Symposium, 1997.
6. INCOSE Web Site, http://www.incose.org/whatis.html
7. Maier, M. W., "Systems Architecting; An Emergent Discipline?", Proceedings of 6th INCOSE Symposium, 1996.
8. Maier, M. W., "Architecting Principles for Systems-of-Systems", The Journal of System Engineering, 1998.

9. Maier, M. W., "Reconciling Systems and Software Architecture", Proceedings of 8th INCOSE Symposium, 1998.
10. Martin, J. N., System Engineering Guidebook: A Process for Developing Systems and Products, CRC Press, U.S.A., 1997
11. Newbern D. and Nolte J., "Engineering of Complex Systems: Understanding the Art Side", Proceedings of 8th INCOSE Symposium, 1998
12. Rechtin, E. and Maier, M. W., The Art of Systems Architecting, CRC Press, U.S.A., 1997
13. SAWG Definitions, Proceedings of 6th INCOSE Symposium, 1996
14. Smith, Timothy B., "Systems Architecting During the Client Interaction Cycle", Proceedings of 5th INCOSE Symposium, 1995
15. Steiner, Rick, "System Structures and Evolvability: Definitions and Perspective", Proceedings of 8th INCOSE Symposium, 1998
16. Yoo, I., Kim, J. and Park, W., "A Development Guide of Robust System Architecture", accepted at INCOSE 10th Annual Symposium, July 16-20, 2000.
17. 고등기술원, 전산보조 시스템 설계 및 개발기술, 과학기술부 보고서, 1998, 1999.
18. 고등기술원, 차세대 고속전철 시스템엔지니어링 체계 구축, 철도연구원 보고서, 1999.
19. 박성현, 품질공학, 민영사, 1993
20. 이재우, "기능분석을 이용한 항공기 설계요구의 할당 및 추적에 관한 연구", 한국군사과학기술학회지, 제2권, 제2호, pp.52-60, 1999.
21. 한국전자통신연구원, TDX-10 개발지침서 I, II, III, 1993.