시스템 엔지니어링을 위한 자동화 도구의 변천 및 사례 조사

Survey on System Engineering Tools and Its Evolution

이재철(J. C. Lee) 컴퓨터구조연구실 선임연구원
김용연(Y. Y. Kim) 컴퓨터구조연구실 선임연구원, 실장
임기욱(K. W. Rim) 시스템연구부 책임연구원, 부장

본 고는 시스템 엔지니어링의 유래부터 현재까지의 변천 및 자동화 도구의 발전 과정을 소개하고, 그 사례를 조사 분석하였다. 시스템 엔지니어링의 발전에 대한 조사 연구는 미국을 대상으로 서술하였으며, 적용되었던 자동화 도구(시스템 엔지니어링 소프트웨어)의 사례 조사 연구와 병행하여 발전 동향을 분석하였다.

I. 서 론

미국에서의 시스템 엔지니어링 분야는 1950년대 말부터 1960년대 중반까지 상승 고도를 달려왔으며, 1970년대에는 쇠퇴하였고, 1980년대 말부터 다시 상승 국면을 타고 있다. 시스템 엔지니어링은 시스템 모델을 표현하는 FFBD(Functional Flow Block Diagram)들을 손으로 도면을 제도하고, 타이핑된 간단한 문서로써 규격을 표현하는 일에서 시작하여 수행 모델을 산출하기 위한 자동화 조작에 이르기까지 발전해왔다. 지금의 시스템 엔지니어링은 컨커런트 엔지니어링(Concurrent Engineering)을 위한 도구 세트의 한 부분이 되는 CASE, 타 서브시스템과 컴포넌트(component) 설계 도구들간의 통합(integration) 및 자동화하는 추세에 있다.

본 고에서는 시스템 엔지니어링의 간단한 개념 소개를 하고, 그 변천과정을 살펴보면서 자동화 도구의 사례 조사 연구와 병행하여 발전 동향을 분석하였다[1].

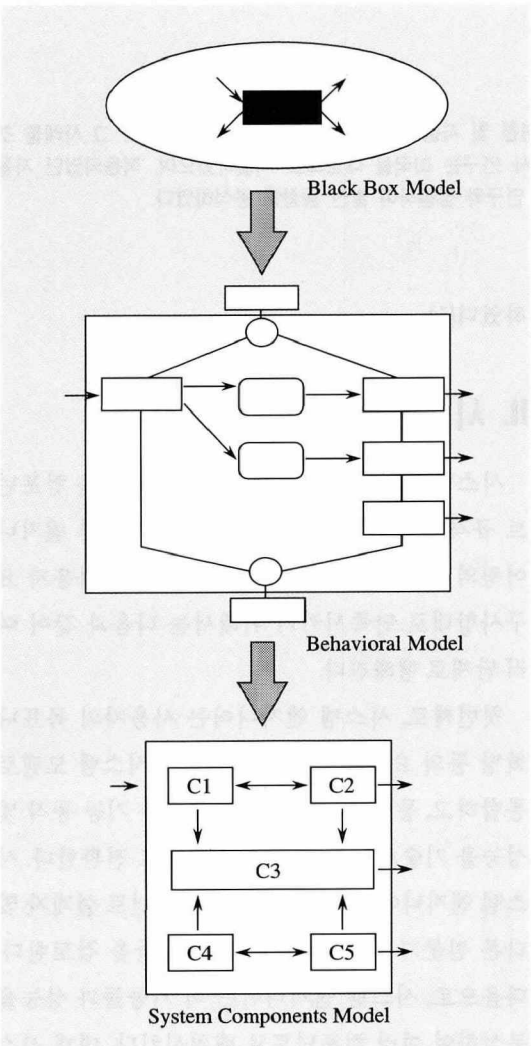
II. 시스템 엔지니어링의 소개

시스템 엔지니어링은 사용자 요구사항을 컴포넌트 규격으로 변환시키는 분야이다. 시스템 엔지니어링의 시스템 컴포넌트 모델을 원래의 사용자 요구사항대로 만족시키기 위해서는 다음과 같이 여러 단계로 행해진다.

첫번째로, 시스템 엔지니어는 사용자의 목표나 희망 등의 요구사항들을 블랙박스 시스템 모델로 통합하고, 통합된 블랙박스의 모델은 기능 동작 및 성능을 기술하는 동작 시스템 모델로 전환한다. 시스템 엔지니어, 사용자, 그리고 컴포넌트 설계자 및 다른 전문가 들은 이 동작 및 기능들을 검토한다. 다음으로, 시스템 엔지니어는 이 기능들과 성능을 분석하여 여러 컴포넌트로 배치시킨다. 대개 시스템 엔지니어들은 체계적으로 설계 영역을 조사하

면서 여러 컴포넌트의 분해 및 배치들을 발전시킨다. (그림 1)은 시스템 엔지니어링의 3가지 단계로 처리되는 과정을 나타낸 것으로 블랙박스 시스템 모델에서 동작 시스템 모델로 변환되고, 동작 모델에서 시스템 컴포넌트 모델로의 변환과정을 나타낸 것이다[1, 4].

컴포넌트 개발자들은 실행 가능성, 비용 효과, 개발일정, 위험성을 고려한 설계를 평가한다. 추가적으로, 신뢰성, 가용성, 휴먼 인터페이스, 교육, 생산성 등의 관련 전문가들은 설계 관점에서 설계의 전체 라이프 사이클의 결과를 평가한다. 시스템 엔지니어링의 처리 과정은 최적화된 설계라고 판단될 때까지 반복된다[2, 5].



(그림 1) 시스템 엔지니어링의 처리 과정

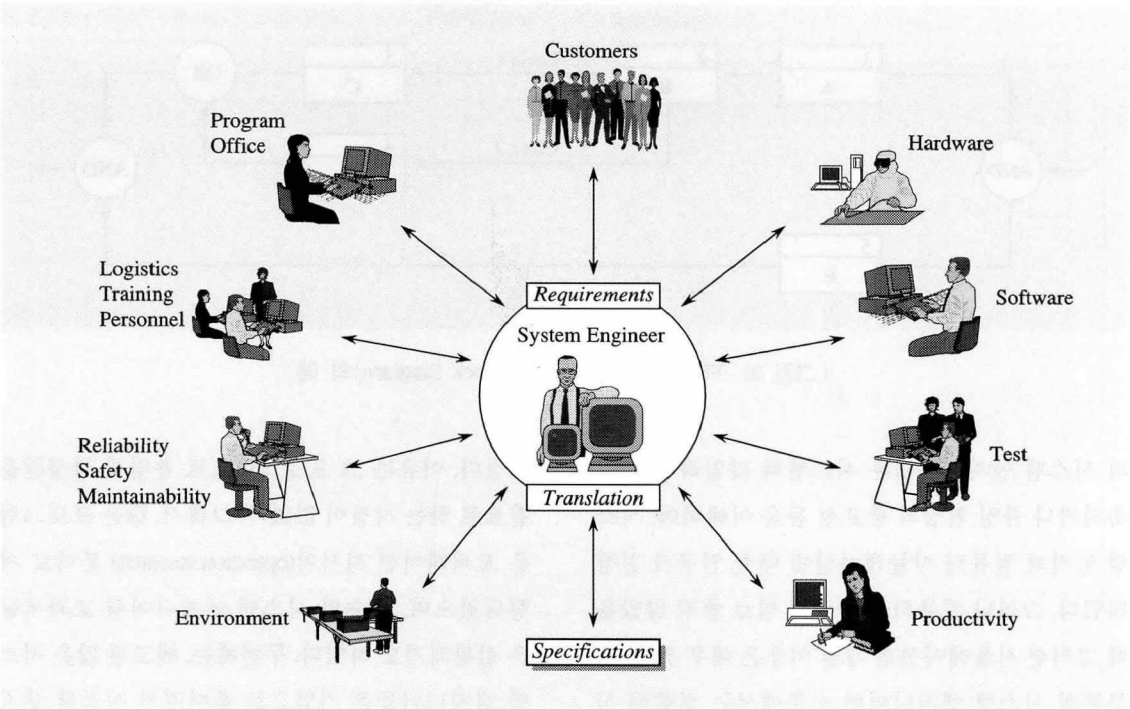
(그림 2)는 시스템 엔지니어링의 개념을 시스템 엔지니어를 중심으로 한 다른 여러 엔지니어들과의 상호 관계로 표현한 것이다. 시스템 엔지니어링의 처리 과정은 시스템 엔지니어, 컴포넌트 개발자 그리고 여러 전문가들이 한 팀으로서 모든 일을 한다면 최상의 모습이 될 것이다.

III. 시스템 엔지니어링의 변천 및 도구의 발전

1. 1940년대에서 1960년대: 연필과 종이 시대에서의 시스템 엔지니어링의 성장

시스템 엔지니어링 분야는 1940년대 말에서 1950년대 초에 세계 2차대전에서의 경험을 토대로 해서 시스템 과학 기술의 이론적 기반으로부터 시작하였다. 소련의 인공위성 및 전략 탄도탄을 전개한 것과 관련하여 미 공군은 유아기의 시스템 엔지니어링 기술에서 완전 탈피하기 위한 정책을 수립하였으며, 이를 위해 Cal Tech의 Simon Ramo 교수와 Hughes Aircraft사의 Deal Wooleredge에 의해 Ramo-Wooleredge Laboratories가 설립되었다. 그리고, 탄도 미사일 Atlas, Titan 그리고 Minteman 개발을 위한 시스템 엔지니어링 서비스를 제공하게 되었다.

1963-4년경, 핵탄도 미사일 개발에서의 성공으로 인해 시스템 엔지니어링은 초보 단계에서 벗어나



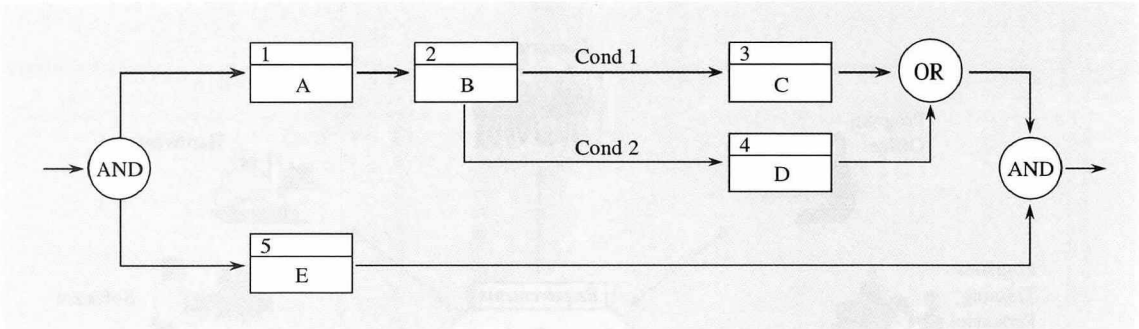
(그림 2) 시스템 엔지니어링의 개념

더욱 성숙한 단계로 진보하게 되었다. 결과적으로 미 국방성에서는 모든 DoD 프로젝트들의 개발에서는 시스템 엔지니어링의 사용을 채택, 지정하였다.

그리고 시스템 엔지니어링의 개념은 미국 국방성이 제정하는 국방 표준 규격인 MIL-STD499로 발표하게 되었다. 이로써 시스템 엔지니어링의 관심도는 더욱 높아져서 미국의 대학 교과 과정에 포함되어졌으며, 약 24권의 시스템 엔지니어링 교과서가 나오게 되었다. 많은 미국의 대학들에서 시스템 엔지니어링 학과(Department of System Engineering) 혹은 시스템 과학과(Department of System Science)가 출범하게 되었다.

1960년대 초의 시스템 엔지니어링 기술의 공헌 중 하나는 요구되는 시스템 동작(behavior)을 제공하기 위한 시스템 기능의 상호작용 전달기능을 인식시킨 것이다. 기능적 흐름을 나타내는 FFBD는 순차(sequence), 선택(selections), 그리고 기능의 병행성으로 시스템 동작을 그래픽으로 표현해준다. FFBD를 이용한 접근의 장점은 시스템 동작의 타임라인(time-line) 분석을 직접 지원한다는 것이다. 시간적 표현과 분석이 이루어져야 하는 아주 복잡한 시스템에 있어서 구조적인 메커니즘을 제공한다. 그러나 기능의 입출력 에러가 존재할 수 있다는 결점을 가지고 있다. (그림 3)은 FFBD의 예를 나타낸 것이다[7].

시스템 엔지니어링은 컴퓨터 기술이 발전되면서 개발단계의 한 시점에서 새롭게 이용되었다. 초기



(그림 3) FFBD(Functional Flow Block Diagram)의 예

의 시스템 엔지니어링은 시스템의 타임라인을 예측하거나 큐잉 현상의 중요성 등을 이해하여, 시스템 동작의 컴퓨터 시뮬레이션에 대한 연구가 진행되었다. 그러나 컴퓨터의 능력이 별로 좋지 않았을 때 그러한 시뮬레이션의 개발 비용은 매우 컸다. 대부분의 시스템 엔지니어의 실무에서는 컴퓨터 사용료가 비싼 이유로 이 기술에 대해서는 매우 비효과적이었다. 시스템 엔지니어는 FFBD 다이어그램을 만들어 내기 위하여 제도공(도안가)을 사용하였다. 그리고 규격서와 관련된 분석 보고서들을 만들기 위해서 타이프라이터가 이용되었다.

2. 1970년대와 1980년대: 시스템 엔지니어링의 쇠퇴기; 자동화 도구의 상승기

1968-69년, 미국의 항공우주산업이 불황국면을 맞이하게 되면서, 많은 젊은 시스템 엔지니어들이 회사로부터 해고되는 사태가 발생하였다. 이러한 이유로, 이 시기의 미국의 시스템 엔지니어링 분야에 대한 흥미는 쇠퇴하기 시작하였다. 항공우주 산업체의 많은 시스템 엔지니어들이 해고되었다는 사실은 다음의 두가지 주요 결과를 초래하였다. 하나는 학교에서의 시스템 엔지니어링의 인기가 없어졌다. 이유는 그 프로그램으로 졸업한 학생들을 필요로 하는 시장이 없었다. 그래서, 많은 프로그램은 오퍼레이션 리서치(operation research) 분야로 지향되었으며, 다수의 시스템 엔지니어링 교과서들은 절판되기도 하였다. 두번째는, 해고된 많은 시스템 엔지니어들은 기업으로 흩어져서 시스템 엔지니어링의 개념이 다른 여러 학문 분야와 통합되었으나, 그들의 시스템 엔지니어링에 대한 본질은 잃게 되었다. 시스템 엔지니어링의 지식 또한 학교에서의 인기가 감퇴됨에 따라 분산되어졌다. 이로 인해 시스템 엔지니어링의 저널이 없었던 것은 물론 시스템 엔지니어링을 위한 학회 조직도 없었으므로 시스템 엔지니어링의 연구, 경험으로 축적된 기술을 교환하기 위한 학술회의 등은 이루어지지 못하였다. 이 시기에는 시스템 엔지니어들의 세력이 없어지고, 시스템 엔지니어링 분야는 거의 사장되었다.

1970년대에는 우주항공산업의 재원이 늘어나면서 시스템 엔지니어링이 다시 일어나기 시작하였다. 그러나 과거에 시스템 엔지니어링의 적용이 실무에 표준, 제도화되어 왔던 이래로 시스템 엔지니어링 개념의 개발을 위한 더 이상의 재시도는 하지

않았다. 교육은 우주 항공 회사들의 내부 프로그램에 의해 실무와 표준을 가르치는데 초점을 맞춘 기초적인 사항으로 다룬 정도였다. 결과적으로 시스템 엔지니어링 기법 적용이 제대로 안되었던 납품 시스템의 고장률은 증가하게 되었으며, 시스템 개발은 그 요구사항에 부합되지 않았고, 그의 목표에도 훨씬 미달되었다.

1970년에서 1980년경의 여러 주요 개발들은 강력한 새로운 표기법, 방법론, 자동화 도구에 의한 지원, 컨커런트 엔지니어링의 개념으로 대치되는 환경으로 변화되었다. 1970년대에서는 컴퓨터 소프트웨어 개발자들이 복잡한 컴퓨터의 소프트웨어 시스템 혹은 광대한 소프트웨어 컴포넌트를 가진 시스템의 고장률에 대해 관심을 가지게 되었다.

소프트웨어 엔지니어들은 소프트웨어에서의 제어나 기능 흐름의 세밀한 표현을 위한 필요성을 인식하였다. 그리고 데이터의 흐름을 표현하는 DFD (Data Flow Diagram)들을 보완하기 위하여 상태 머신(state machine)이나 페트리 네트(Petri Net)를 사용하기 시작했다[5]. 이 기간 동안에 타 엔지니어링 분야를 위한 표기법들 예를 들면, 칩 설계를 위한 VHDL (VHSIC Hardware Description Language), 시스템 기술 언어(System Description Language; SDL) 그리고 CAD(Computer Aided Design), CAE(Computer Aided Engineering)와 같은 자동화 도구들이 개발되었다. 시스템 엔지니어들은 대개 시스템 엔지니어링에서 요구되는 규격, 모델링, 시뮬레이션 등의 요구사항들을 처리하기 위한 도구들을 개발하였다. CASE(Computer Aided Software Engineering), CAD 그리고 CAE는 서브시스템과 컴포넌트들의 설계에 있어서 효과와 능률을 높이는 우수한 장점을 가지고 있다. CASE는 우선적으로, 소프트웨어 요구사항들의 표현을 위한 훌륭한 도구로 평가되어왔다. 그리고 소프트웨어 설계 즉, C 및 ADA 프로그래밍 언어 그리고 타 프로그래밍 언어의 소프트웨어 제작에 자동화를 지원한다. CASE는 소프트웨어 시스템의 시스템 레벨 요구사항들을 표현하는데 성공적으로 사용되어 왔다.

CASE 및 다른 자동화 도구는 엔지니어링 분야의 업무를 능률적으로 향상시켰고, 복합 시스템의 설계 및 구축 과정에 혼합되었다. 시스템 엔지니어들은 워드프로세서 시스템을 사용하여 완전한 정도는 아니지만 톱레벨(top-level)의 규격을 만들었다[3]. 컴포넌트 설계자들은 이러한 규격을 읽고 새로운 방법론이나 표기법 즉, 데이터 흐름도(DFD)[7] 및 보충 문서를 사용하여 설계를 하였다. 구현자들이나 전문 기술자들은 문서를 읽고 컴포넌트(예를 들면, 코드의 모듈)들을 산출하거나, 신뢰성, 안전성 등을 결정하기 위해 그 자동화 도구들을 사용하였다.

영문 표기에서 엔지니어링 표기법으로, 그리고 한 엔지니어링 표기법에서 또 다른 것으로의 표기법으로 변환한다는 것은 에러의 유발이나 일거리가 부과되었고, 이해하지 못하는 경향도 있었다. 더욱이 매번 설계를 변경해야만 했었다.

시스템 엔지니어링의 과정이 세밀하게 진행됨에 따라, 시스템 엔지니어들은 설계 문서를 검토하였고, 설계 및 전문 엔지니어들은 구현을 검토하였으며, 모든 엔지니어들은 일을 수정, 재수정하였다. 그렇게 하지 않으면 납품되는 시스템은 요구사항을 만족하지 못한 상태가 되었다. 이러한 자동화 도구들은 컨커런트 엔지니어링에 적용하였지만, 엔

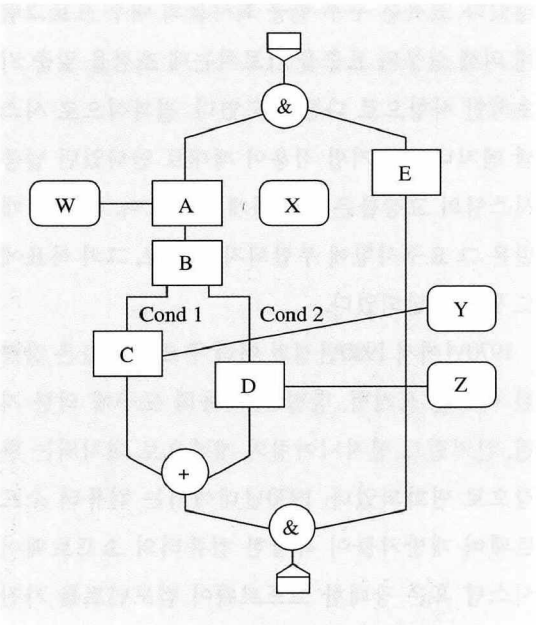
지니어 입장에서는 컨커런트 엔지니어링 팀의 다른 멤버들과 도구 사이에서의 정보 교환에 따른 어려움이 남아 있었다.

3. 1980년대말과 1990년대: 시스템 엔지니어링의 부활; 자동화 도구의 통합

1980년대말 시스템 엔지니어링은 NCOSE(National Council on System Engineering)의 창립과 시스템 엔지니어링 자동화 도구의 유용성(availability)이 대두되면서 부활하였다. 1980년대 초기에 개인용 컴퓨터(Personal Computer; PC)의 출현은 시스템 엔지니어링을 지원하기 위한 많은 PC 기반의 자동화 도구들이 개발되어 사용되는 요인이 되었다. 자동화 도구들 중 어떠한 것은 시스템 요구사항 등에 관한 해석 및 시뮬레이션 같은 특정 작업을 자동화하며, 아울러 문서를 생성해 주는 것도 있다.

시스템 엔지니어들은 다른 분야의 엔지니어와 상호 기술 교류를 해야 하기 때문에, 시스템 엔지니어링을 위한 시스템 통합 도구 세트(set)의 필요성에 대한 요구가 증가되어 왔다. 시스템 엔지니어링 도구의 유용성은 시스템의 톱레벨의 요구사항들을 승인하며, 텍스트 형태보다는 하나의 어떠한 모델로 표현하게 되었다.

CASE 도구의 데이터 흐름도와 상태 테이블은 소프트웨어 컴포넌트의 데이터 흐름과 제어 변천의 표현을 하는데, 시스템 모델은 동작 다이어그램(behavioral diagram)의 표현을 사용하여 만들어진다. 시스템 엔지니어링 도구의 한 예로써 Ascent Logic 사의 RDD-100을 들 수 있는데, RDD-100의 동작 다이어그램은 (그림 4)에서 보여지는 것과 같이 정의된다.

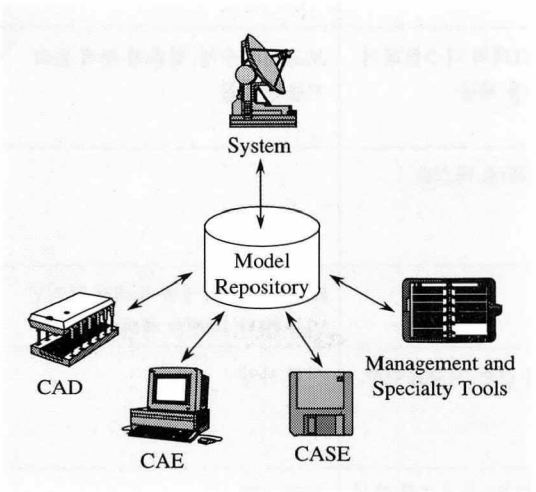


(그림 4) 동작 다이어그램의 예

이 동작이 서브시스템 혹은 컴포넌트에 할당될 때 컴포넌트들 사이에서 이동되는 입출력의 순차(sequence)는 명백하게 정의된다. 이 RDD-100은 CASE, CAE와 같은 타 도구들의 모델 기반 요구사항들에 통용될 수 있는 모델 기반의 시스템 요구사항 세트를 제공한다.

시스템 엔지니어링 도구에서 CAE로의 모델 기반 요구사항들이 정확성을 가지고, 효율적으로 이동하기 위해서는, 도구들간의 조정이 자동적으로 이루어지도록 통합되어져야 한다. 통합된 컨커런트 엔지니어링 도구 세트에서는 모든 도구들의 액세스 및 적당한 제어를 위한 주체(subject)를 받아들이며, 요구사항들을 수정, 확장하는 일을 시스템 엔지니어링 도구에서 자동적으로 시스템 모델을 만들어준다. 여러 엔지니어링 자동화 도구들은 공통의 분산 저장소(repository)를 공유한다. (그림 5)에서

보여지는 것과 같이 시스템 요구사항들은 시스템 엔지니어링 도구에서 CASE, CAD, CAE 그리고 기타 전문성 도구 및 관리 도구까지 플로다운(flow down)된다.



(그림 5) 시스템 요구사항의 플로다운 모습

이 자동화 도구 세트는 시스템 설계의 상호 정보 전달 및 시스템 엔지니어, 설계자, 프로그램 관리자 등 여러 엔지니어들간의 작업 흐름 정보를 최적화 시킨다.

자동화 도구 세트의 통합을 위해서는 다른 시맨틱(semantic) 구조의 도구들 간의 변환 및 변환된 정보의 이동, 정보의 각 형태에 대한 형상 관리(configuration management) 규정 등을 요구한다[6,7]. 그리고 도구들 간의 정보 일치성을 유지하기 위한 메커니즘, 도구들의 수정을 위한 방법에 대한 규정이 필요하다.

시스템 엔지니어링 도구와 관련하여 하나의 예를 들어 살펴보면, 소프트웨어와 관련된 것들이 시스템 설계에 변경을 요청할 때 CASE 도구는 시스템 도구에게 어떠한 설명이나 사유를 보낸다. 그리고 엔지니어는 새로운 버전을 만들기 위해 시스템 설계를 편집한다. 시스템 도구는 CASE 도구에게 변경된 버전의 한 세트를 제공한다. 이 때, CASE 도구는 변경되지 않은 버전들을 해칠 위험이 없는 소프트웨어 설계와 이 변경된 버전들을 병합하기 위한 메커니즘이 필요하다. 시스템 엔지니어링 도구와 CAD, CAE 도구 사이에서 변환이 일어날 때는 상호간의 정보 교환이 필히 요구된다[8].

IV. 시스템 엔지니어링 도구의 사례: RDD-100

RDD-100®은 미국의 Ascent Logic사에서 개발한 시스템 엔지니어링을 위한 자동화 도구(시스템 엔지니어링 소프트웨어 패키지)이며, 시스템 레벨의 설계 및 CASE, CAD 등의 자동화 도구와의 인터페이스를 지원한다.

RDD-100에는 여러가지의 기능을 가지는 소프트웨어 패키지들로써 구성되어 있으며, 각각의 패키지에 따른 기능에 대한 설명은 <표 1>과 같다.

RDD-100은 프로젝트의 개발, 시스템 분석 및 규격화, 검증 등의 업무에서 사용된다. RDD-100의 기술적인 면을 살펴보면, 프로젝트 관리, 시스템의 개념 설계를 기술한 데이터의 저장, 계층적 설계, 보고서 작성기, 검증을 위한 기능(Dynamic Verification Facility; DVF) 선택, 컴포넌트 도구와의 인터페이스 기능 등과 같은 특징을 가지고 있다. RDD-100에서의 시스템 동작 모델은 그래픽으로 기술하며, 시스템 모델로부터 동작 시나리오가 자동적으로 만들어진다. 그리고 시스템 설계를 검증하기 위한 모델을 실행시킨다. 시스템을 기술하는 방식으로는 톱

〈표 1〉 RDD-100의 패키지 구성 및 기능 비교

패키지 품목	기 능	비 고
RDD-100/SD™ System Designer	그래픽, 보고서 작성 및 자동 생성, 모델 시뮬레이션 등의 기능을 제공	RDD-100 전 특징을 제공
RDD-100/SA™ System Analyst	시스템 동작 모델링, 모델 분석 및 개선시키는 기능을 제공	
RDD-100/RM™ Requirements Manager	전문(full text)편집, 보고서 실행, 형상관리, 그래픽 디스플레이 (FFBD, N-squared Chart, Hierarchy View) 기능을 제공	보고서의 수정, 일관성 검색 등의 기능은 없음
RDD-100/RE™ Requirements Editor	외부의 원천 문서를 시스템 설계 데이터(SDS)로 텍스트 편집하는 기능을 제공	
RDD-100/SB™ System Browser	설계과정을 심의하기 위한 기능을 제공	RDD-100의 특징들에 대해 읽기만 가능함(단, DVF는 제외)
RDD-100/DVF™ Dynamic Verification Facility(DVF) Option	RDD-100/SD, RDD-100/SA와 함께 사용되어 검증, 시뮬레이션, 시스템 모델을 개선시키는 기능을 제공	선택 사양
RDD-100/ILF™ Interleaf Option	보고서 작성기에 의해 만들어지는 보고서를 Interleaf 호환 양식으로 만들어 주는 기능을 제공	선택 사양
RDD-100/TWI™ Cadre Teamwork Interface Bridge Option	CDIF(CASE Data Interchange Format) context와 데이터 플로 다이어그램(DFD)으로 출력될 수 있도록 해주는 기능을 제공	선택 사양이며, CASE 및 EDA 개발 환경에 직접 인터페이스 할 수 있음

다운(top-down) 혹은 보텀업(bottom-up) 방식을 사용할 수 있으며, 상위 레벨의 기술을 상세한 하위 레벨의 기술로 만들어 준다.

RDD-100은 시스템 설계자로 하여금 시스템 성능, 품질, 신뢰도 등을 높일 수 있도록 지원하는 객체 지향적 도구이며, 컴퓨터 기반의 시스템 설계용 자동화 도구로 평가된다.

V. 결 론

시스템 엔지니어링 분야는 사용자 요구사항이 원래대로 구현 및 통합이 안정된 컴포넌트 규격으로 전환될 때의 처리 과정을 정의하는 분야로서 1940년대부터 1960년대에 나타났었다. 1980년대말과 1990년대에서는 자동화 도구의 통합이 요구됨에 따라 시스템 엔지니어링이 부활되었다. 한편, 그러한 통합 기술이 자동화 도구의 설계자에게 많은 도전적인 시도가 있었음을 나타내며, 그 통합 기술은 분산 설계 데이터베이스, 정교하고 간결한 언어로 된 엔지니어링 모델들을 공유할 수 있도록 해준다.

본 고에서는 시스템 엔지니어링의 유래부터 현재까지의 변천 과정 및 자동화 도구의 발전 동향을 살펴 보았고, 시스템 엔지니어링을 위한 도구로서

RDD-100을 대상으로 한 조사 분석을 하였다. 시스템 개발 프로젝트에서 규격화하는 일, 설계하는 일 그리고 복합 시스템을 이루기 위한 과정은 프로젝트 라이프 사이클을 통한 많은 분야의 엔지니어 간의 강한 상호 작용을 요구함을 알 수 있었다[5].

결론적으로, 향후의 복잡한 대규모 시스템의 개발 프로젝트를 위해서는 사용자 요구사항을 비롯하여 컴포넌트 설계까지의 시스템 엔지니어링 자동화 업무를 도입해야 하며, 이를 위해서는 컴퓨터 기반의 시스템 엔지니어링 도구가 CAD 환경과 통합되는 개발환경의 구축이 절실히 요구된다고 본다. 그리고 대규모 시스템 개발 프로젝트에서 시스템 엔지니어링 도구는 시스템 설계자로 하여금 시스템 성능, 품질, 신뢰도 등을 높일 수 있다고 기대한다.

참고문헌

[1] J. Douglas Sailor, "System engineering: An introduction," *Tutorial; System and Software Requirements Engineering*, R. H. Thayer and M. Dorfman, Ed., Washington, D.C.: IEEE Computer Society Press, pp. 35-47, 1990.

[2] H. Eisner, *Computer-Aided Systems Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[3] E. Dale Nelsen, "System engineering and requirement allocation," *Tutorial: System and Software Requirements Engineering*, R. H. Thayer and M. Dorfman, Ed., Washington, D.C.: IEEE Computer Society Press, pp. 60-76, 1990.

[4] P. Zave, "The operational versus the conventional approach to software development," *Communications of the ACM*, vol. 27, no. 2, pp. 104-118, Feb. 1984.

[5] Alan M. Davis, "A comparison of techniques for the specification of external system behavior," *Communications of the ACM*, vol. 31, no. 9, pp. 1098-1115, Sep. 1988.

[6] Edward H. Bersoff, "Elements of software configuration management," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, pp. 79-87, Jan. 1984.

[7] Richard H. Thayer, "Software system engineering," *Tutorial: System and Software Requirements Engineering*, R. H. Thayer and M. Dorfman, Ed., Washington, D.C.: IEEE Computer Society Press, pp. 77-116, 1990.

[8] A. I. Wasserman and P. A. Pircher, "A graphical, extensible integrated environment for software development," *SIGPlan Notices of ACM*, pp. 131-142, Jan. 1987.