시스템 통합을 위한 소프트웨어 과제 관리 및 개발 모형

Management and Development Models of Software Projects for System Integration

한영근*, 이규봉**, 권순기***

* 명지대학교 산업공학과
** 생산기술연구원 생산시스템개발센터
*** 펜실베니아주립대학교 산업공학과

ABSTRACT

In order to accomplish large-size information systems successfully, one of the most important factor is the system integration(SI) of individual software projects which consist of the systems. A developer of each project should consider aspects of SI through the development cycle, and a manager of the entire system should manage, control, and evaluate each from a SI point of view. In this research, management models for the system managers, development models for the developers of unit projects, and standardized output documents for the management and evaluation purpose are presented based on the SI concepts.

1. 서론

정보기술이 더욱 복잡화, 다양화, 고도화되며 발전되어 가는 추세에서 종래와 같은 방식으로의 정보시스템 구현은 점점 어려워 지고 있다. 많은 투자비용과 기술인력이 소요되며, 정보시스템도 전략적 차원과 결합됨으로써 시스템 개발이 복잡해지게 되었다. 효율적이고 경쟁력을 갖춘 시스템을 개발하기 위해서는 정보시스템 어플리케이션의 통합연계, 데이타베이스의 공유 뿐만 아니라 정보통신망 같은 네트워크의 통합과 여러 관리제도나 업무처리의 통합 등 총체적인 시스템통합(SI; System Integration)의 개념이 필요하다.

시스템통합을 이루기 위해서는 먼저 기반구조(Infrastructure)가 구축되어야 한다. 그 중의 하나로 공통의 소프트웨어 개발 방법론의 구축을 들 수 있다. 현재까지 사용되고 있는 방법론은 주로 물리적인 시스템 설계와 프로그래밍을 위한 것이었다. 따라서 그러한 방법론에 의한 정보시스템개발은 단지 소프트웨어 개발자만을 위한 것이었다. 그러나 정보시스템의 최종 사용자들이 필요로 하는 소프트웨어를 개발할 수 있도록 개발과정을 체계적으로 관리할 필요가 있다. 즉, 모든 개발과정을 단계별로 나누어 진도를 점검하고, 각 단계의 관리를 위해 최소한의 표준문서를 규정하는 것이 바람직 하다.

정부 주도형과 같은 대형 시스템 개발 과제가 성공적으로 수행되기 위해서는 현재 개발이 진행되고 있는 단위 소프트웨어 과제들의 통합화가 매우 중요하다. 따라서 모든 소과제들이 개발되는 과정에서 시스템통합 측면이 고려되어야 하고, 주관자는 각 과제들을 시스템통합 관점에서 관리, 조정, 평가하여야 한다.

본 연구에서는 SI 개념에 기초하여 각 소프트웨어 개발과제를 위한 관리자 측면에서의 관리 모형과 개발자 측면에서의 개발 모형과, 관리와 평가를 위해 각 단계별로 요구되는 표준화된 산출물 종류를 제시한다.

2. 소프트웨어 과제 관리 모형

2.1 개요

소프트웨어 개발 과제 관리는 성공적인 과제를 보장하기 위해 요구되는 모든 사항을 포함하며, 여기서 성공적인 과제란 주어진 납기 내, 예산 범위 내에서 완성된 사용자와 개발자의 품질 요구조건을 모두 만족하는 것을 의미한다. 소프트웨어 개발 과제의 관리는 수행할 작업내용이 무엇인가를 결정하고, 과제 수행의 절차 및 일정을 계획하고, 그에 따라 필요한 자원을 균형적으로 할당하는 작업(Balancing) 등을 포함한다. 좀더 세부적으로 그림 1은 소프트웨어 개발 과제 관리의 활동 내용을 보여 주고 있다.

2.2 주요 관리 활동

과제관리의 주요 활동들은 다음과 같다.

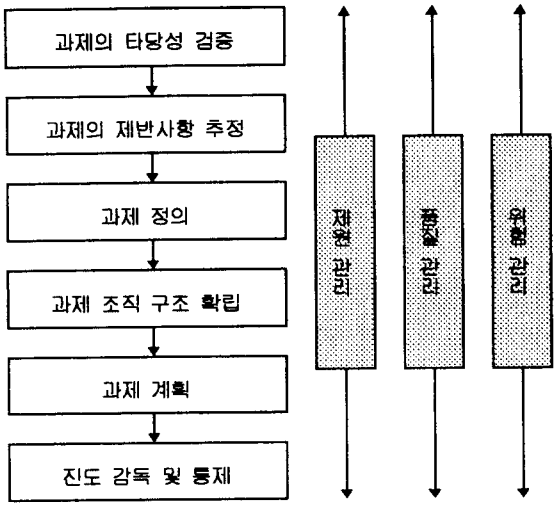

그림 1. 소프트웨어 개발과제 관리의 주요 활동

- 과제 타당성 검증
  최종 시스템으로부터 얻어질 목표와 이익을 달성하기 위해 필요한 투자안들 간에 비교가 이루어짐으로써 과제 착수 이전에 타당성을 검증하는 활동을 의미한다.
- 과제 제반 사항 추정
  기간, 비용 및 소요자원의 관점에서 과제를 분석하는 활동을 의미한다.
- 과제 정의
  과제가 추진되기 전에 그 자체의 정확한 범위가 정의되어야 하는데, 여기에는 과제의 목표, 제출물의 목록, 제안된 소요기간, 소요자원, 고객의 책임, 품질계획, 비용 요소, 타 과제와의 통합 사항 등이 포함된다.
- 과제 조직 구조 확립
  과제 수행을 위한 조직구조를 확립해 나가는 활동으로 책임의 명백한 할당, 분명한 위임구조와 권한구조, 적절한 통제 체계 따위가 이루어져야 한다.
- 과제 계획
  과제가 분명하게 정의되고 조직이 구성되게 된 후 필요한 일은 수행할 업무에 대한 계획을 세우는 활동이다.
- 진도 감독 및 통제
  감독은 과제 진도에 대한 정보의 수집 과정을 말하며, 통제는 얻어진 정보의 관점에서 취해지는 행동을 말한다.

2.3 기타 관리 활동

그림 1에서 우측의 관리 활동은 주요 활동 과정과 병행하여 운영, 협조되어야 할 활동들임을 나타내는데, 그 내용은 다음과 같다.

- 재원 관리
  과제 수행 과정 중에 최종 시스템을 구성하는 여러 재원들이 어떤 일관된 통제 하에서 유지되는 것이 중요하다. 따라서 공식적으로 과제의 라이프사이클 중에 모든 항목들을 식별하고 유지보수될 방법을 확립하여 주며, 출하 및 변경을 통제하고, 상태 변경 시에는 절차를 확립시켜주는 것이 필요하다.
- 품질 관리
  소프트웨어 시스템의 여러 재원 항목은 제품이 사용자의 요구 사항을 따르고 있다는 것을 보장해 주기 위해 개발 과정에서 품질적인 측면에서 통제되어야 한다.
- 위험 관리
  위험요소는 과제의 목표를 충족시키지 못하게 하는 원인이 될 수 있는 것으로서 과제의 여러 부문 중에 이러한 것을 포함하고 있는 곳을 위험영역이라 한다. 위험영역은 작업성격에 따라 혹은 과제를 수행해 감에 따라 변경될 수 있으며, 따라서 과제의 정의 과정에서 식별되고 개발 전기간에 걸쳐 특별하게 감독, 통제되는 것이 필요하다.

이밖에 효과적인 소프트웨어 과제 수행을 위해 고려해야 될 중요한 것 중의 하나는 과제 진척 성과의 가시성(Visibility)에 관한 것이다. 가시성이란 과제가 진행됨에 따라 이루어진 성과를 직접 눈으로 확인할 수 있어야 한다는 것인데, 이를 위해서 여러가지 유형의 중간 성과물이 요구된다. 즉, 과제의 진행과정 중 일정 시점들을 평가 시점으로 설정하고 각 시점에서 평가되어야 할 중간 성과물을 사전에 명확히 정의해 놓음으로써 보다 가시적인 통제를 가능케 해야 한다. 중간 성과물은 과제 수행자들 사이의 의사소통을 원활히 하는 데도 크게 도움이 되어 과제 생산성의 향상과 소프트웨어의 품질 향상에도 크게 기여할 수 있다.

3. 소프트웨어 과제 개발 모형

소프트웨어 프로세스 모형이란 그것을 통해 소프트웨어 제품이 완성되어 나가지는 일련의 단계를 뜻한다. 여러가지의 모형이 개발되어 있는데, 이 중에서 Waterfall 모형과 Rapid Prototyping 모형이 가장 널리 쓰이고 있으며 Spiral 모형이 차츰 각광을 받고 있다.

실제 프로세스 모형과는 관계없이 일반적으로 모든 모형은 요구분석, 명세화, 설계, 구현, 통합 등의 단계를 거치게 된다.

3.1 요구 분석 단계

고객의 요구를 파악, 결정하는 단계이다. 가장 주로 쓰이는 방법은 원형제작(Rapid Prototyping)이다. 요구파악의 어려움 때문에 개발자와 사용자가 계속 함께 과제의 개발 과정에 관여하는 JAD(Joint Application Design)가 제시되고 있다.

3.2 명세화 단계

전단계에서 결정된 요구를 명세화하는 단계로, 명세화 문서는 납기, 호환성, 편의성, 신뢰성, 빠른 응답시간 등과 같이 시스템이 만족해야 하는 제한 사항과 각각에 대한 허용 기준을 담게 된다. 바람직한 명세화 문서 혹은 도구는 사용자가 충분히 이해할 수 있도록 비형식적이면서도, 모호하지 않고 오류가 발생하지 않도록 정확해야 한다.

명세화의 방법은 크게 비형식적(Informal), 반형식적(Semi-formal), 그리고 형식적(Formal)인 3가지로 구분할 수 있다. 비형식적인 명세화는 자연언어로 명세서를 작성하는 것으로서, 사용하기 쉬우나 불가피하게 고객의 요구를 무시할 수 있고 모호한 결과를 산출하게 된다. 반대로 형식적인 방법은 강력하고, 정확한 표현을 할 수 있으나 사용하기가 그다지 쉽지 않아 교육시간이 많이 드는 단점이 있는데, Finite State Machines, Petri Nets 및 Z 등의 모델링 도구를 예로 들 수 있다. 그 중간격인 반형식적인 방법의 대표적인 것은 구조적 분석(Structured Systems Analysis)을 들 수 있다. 다이어그램 등과 같은 그래픽적 기법을 이용하여 명세화를 하는 것으로 DeMarco, Gane and Sarsen, 그리고 Yourdon의 기법들과 PSL/PSA, SADT, SREM 등을 예로 들 수 있다.

이 단계에서 CASE 도구가 도와주는 것은 여러가지 그래픽 도구를 제공하여 주는 것과 자료사전(Data Dictionary)을 만들어 주거나 작성을 도와주는 것이다. ADW, Analyst/ Designer, Excelerator, Software through Pictures 및 Teamwork와 같은 CASE 도구들은 자료사전과 그래픽 도구를 통합시켜, 한쪽의 변경이 다른 쪽에서 자동적으로 반영될 수 있게 되어 있다.

3.3 계획 단계

실제적으로 과제를 수행하기에 앞서 세부적으로 전체의 과제 수행 과정에 대한 계획을 세우는 단계이다. 비용이나 소요기간을 추정하기 위해 여러가지 메트릭스가 주의 깊게 고려 및 분석되어야 한다. 세부적인 소프트웨어과제 관리계획(SPMP; Software Project Management Plan)이 산출된다. 이 단계에서 제시된 계획에 의거하여 관리자는 과제의 수행과정을 감독하고 계획에서 벗어날 경우 적절한 행동을 필요할 때마다 취하게 된다. 이 단계에서 연두에 둘 사항은 초기계획은 충분할 수 없으며, 따라서 계획은 전 소프트웨어 개발 과정 및 유지보수 과정에 걸쳐 수정 및 보완되어야 한다는 것이다.

3.4 설계 단계

적절하고 올바른 설계를 위한 여러가지 지침에 따라 실제적인 구현시스템을 설계하는 단계이다. 설계 단계의 입력될 가장 중요한 것은 명세화 문서(시스템이 무엇을 해야 하는지에 대한 서술)이고, 출력될 것은 설계 문서로 시스템이 어떻게 명세화된 것들을 달성하는지에 대한 설명을 담고 있는 것이다.

소프트웨어 설계 단계는 기본설계 혹은 구조설계, 상세설계 혹은 모듈설계 그리고 설계검사의 세부 단계로 구분될 수 있다 [7]. 기본설계에서는 일련의 모듈들이 존재한다는 전제하에 그 모듈들의 전체적인 구조에 대한 논리적 설계, 즉 상위설계가 이루어진다. 상세설계에서는 각 모듈들이 서로 연관되어 있다는 사실은 보류한 채 각 모듈에 대한 물리적인 설계, 즉 하위설계가 이루어진다. 그리고 전 설계과정은 설계검사에 의해 검증 및 보완된다. 이들의 관계는 그림 2에 표현되어 있다.

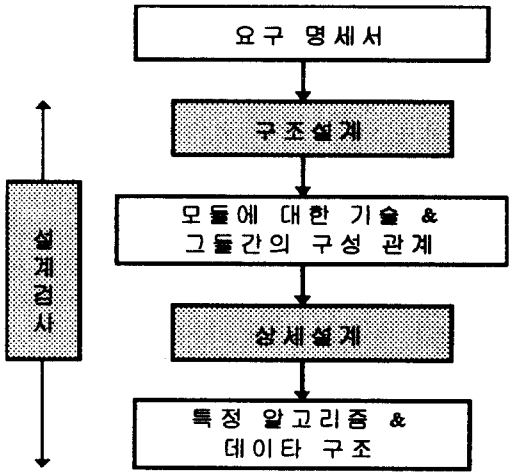

그림 2. 설계 단계의 세가지 활동

3.5 구현 단계

설계단계에서 명세된 내용을 바탕으로, 즉 설계과정의 알고리즘을 중심으로 구체적으로 설계하고 적합한 프로그래밍 언어를 사용하여 실제적인 구현을 하는 단계이다. 또한, 단위 모듈의 처리 논리가 정확한지, 그 결과가 요구된 기능을 만족하게 수행하고 있는지를 단위 시험을 통해 검사한다. 단위시험 과정에서 변경사항이 발생했을 경우 운영지침서를 수정하는 등의 추가조치가 필요하다.

3.6 통합 및 통합 테스트

일반적으로 통합 방법은 통합 과정에 따라 비점진적인 접근 방법과 점진적접근 방법으로 구분된다. 전자는 여러사람이 참여하는 소프트웨어 개발에서는 잘 활용되지 않으며 모든 단위모듈이 한번에 통합되는 Big-Bang 통합이 여기에 속한다. 후자의 방법은 단위모듈의 통합과 동시에 통합된 구성요소에 대하여 시험과 병행하여, 계속적으로 새로운 구성요소를 통합해 나가는 방법이다.

설계명세서에서 작성된 소프트웨어 구성요소의 기능이 제대로 수행되는지를 평가하기 위하여 통합 시험의 항목 설정, 사례 설정 및 절차 설계

등의 활동이 수행되게 된다. 통합시험은 개별 모듈이 통합된 소프트웨어 구조 그리고 모듈간 인터페이스가 타당한 것인가를 시험하기 위한 것으로 통합 접근방법에 맞추어 통합과 함께 수행되는 것이 일반적이다.

4. 대형 시스템 개발 사업의 소프트웨어 과제 관리를 위한 산출물

4.1 대형 시스템 개발 사업의 특성 분석

2절에서 서술된 일반적인 과제 관리 방법론은 관리하고자 하는 실제 과제의 성격에 맞추어 변형되는 것이 필요하다. 정부 주도형의 대형 시스템 개발 사업에 속한 단위 소프트웨어 과제들은 첫째 성격이 서로 다른 소프트웨어를 개발하고 있기 때문에 획일적인 방법론의 제시에는 무리가 따른다는 특성을 가지고 있다. 둘째로, 많은 기관이 참여하고 있으며 각 과제의 수행을 주관하는 기관이 나누어져 있는데 이들은 과제개발 방법론 체계에 관해서 차이가 있다. 셋째로 현실적으로 시간적, 공간적, 경제적 어려움으로 모니터링이나 통제를 하는 것이 용이하지 않다. 따라서 대형 사업의 효과적인 과제 관리를 위해 요구할 산출물들은 다음과 같은 사항을 만족하는 성격의 것이어야 한다.

- 각 개발 기관에 시간적, 경제적 측면에서 과다한 부담이 되지 않도록 산출물의 양이 되도록 적어야 한다.
- 각 개발 기관이 하는 일의 성격에 덜 영향을 받도록 획일적인 개발 방법론 체계와 그에 수반되는 산출물을 제시하기 보다는 관리자의 입장에서 공통적으로 필요한 보고서나 작성 도구를 제시하는 것이 효과적일 수 있다.
- 요구되는 문서나 작성 도구는 작성하기가 용이하여야 한다.
- 산출물 제출의 목적은 감독 및 통제 뿐만이 아니라 개발자에게 도움을 줄 수 있는 것이어야 한다. 그러나 산출물의 형식이나 내용이 개발적 측면을 과도하게 강조하기 보다는 관리적 측면이 보다 강조되야 한다.
- 시스템통합 측면에서 각 소프트웨어 과제의 통합 후의 역할과 기능을 충분히 보여줄 수 있어야 한다.

이와같은 사항을 고려하여, 본 절에서는 작성하기 용이한 시스템 분석 및 설계 도구들을 도출하고 이에 알맞은 산출물 형식을 각 개발 단계 별로 제시한다.

4.2 분석 및 설계 용 산출물 작성 도구

개발자와 관리자의 효과적인 의사 소통을 위해서 유용한 도구 중의 하나는 다이어그램이다. 수많은 분석 및 설계 다이어그램이 소개되고 있으나 그 중 많은 것들이 관리자를 위한 측면보다는 개발자를 위한 측면이 강조되고 있다. 아울러 요구 분석과 명세화에서 쓰일 수 있는 것들을 제외하고는 대부분의 도구들이 너무 상세한 측면을 담고 있어 관리자에게는 큰 도움을 주지 못하는 것이다. 이러한 측면으로부터 과제 관리를 위한 산출물 작성 도구는 다음과 같이 한정하는 것이 바람직 하다.

- 자료흐름도 (Data Flow Diagram): 자료 변환 처리를 흐름으로 연결
- 기능차트 (Function Chart): 계층화된 프로세스간 데이터의 흐름과 제어흐름
- 배경도 (Context Diagram): 개발시스템의 타시스템과의 관계
- 구조도 (Structure Chart): 소프트웨어의 계층 표시
- 실체관계도 (Entity-Relation Chart): 시스템의 총체적인 데이터의 상호관계
- 로드맵 (Road Map): 각 개발 요소에 대한 상세한 일정

위에서 설명된 분석 및 설계 도구들의 소프트웨어 개발 중 적용될 단계와 세부 활용 형태를 표 1 에서 보여주고 있다.

표 1. 분석 및 설계 도구의 단계별 활용

도구	사용 단계	사용 세부 형태
자료흐름도 Data Flow Diagram	요구분석 및 명세화 설계	논리적 자료흐름도 물리적 자료흐름도
기능차트 Function Chart	요구분석 및 명세화	
배경도 Context Diagram	요구분석 및 명세화	
구조도 Structure Chart	계획 설계(구조설계)	조직 구조도 프로그램 구조도
실체관계도 Entity-Relation Chart	요구분석 및 명세화	실체관계도
로드맵 Road Map	개발 전단계	

4.3 개발 단계에서 요구되는 산출물의 내용

앞 절에서 밝혔던 도구들 만으로는 효과적인 관리를 기대하기 어렵다. 개발적 측면과 관리적 측면을 모두 고려할 때 그래픽적인 도구만으로 모든 필요한 내용을 담기는 쉽지 않다. 따라서 이러한 도구들과 함께 기타 서술적인 내용을 담고 있는 보고서가 필요하다.

그러나 관리의 용이성을 보장하기 위해서 상세한 보고서 전부를 제출할 것을 요구하는 것은 무의미하고, 소프트웨어 개발 단계별로 몇가지 보고서만을 요구하는 것이 개발자에게 큰 부담을 주지 않을 것이다. 이러한 고려로부터 개발 단계별 보고서와 그 보고서가 담고 있는 내용들을 보여주는 표 2가 작성되었다. 상세한 산출물의 양식은 참고문헌 [10]에 나타나 있다.

표 2. 개발 모형에 따른 보고서 내용

개발 단계	보고서 명칭	세부 내용	사용 도구
요구 분석 & 명세화	소프트웨어 요구 명세서	기능적 요구사항 명세 자료적 요구사항 명세 인터페이스 요구사항 명세 품질적 요구사항 명세 기타 요구사항	기능차트 논리적 DFD E-R chart 배경도
계획	과제 계획서	과제 제출물 명세 방법, 도구 및 기법 명세 과제의 조직구조 설명 과제관리의 계획 과제일정의 계획	조직구조도 Road Map
설계	소프트웨어 설계 명세서	물리적 구성요소 설계명세 휴먼인터페이스 설계명세 하드웨어와 통신 설계명세 기타 추가사항 설계명세 물리적 데이타베이스 명세 프로그램 구성 명세 프로그램 인터페이스 명세	물리적 DFD DB schema 다이어그램 프로그램구조도
구현	소프트웨어 구현보고서	코딩표준&가이드라인 명세 헤더 형식 명세 프로그램 목록 시험의 목적 및 유형 설명 시험방법과 사례 결정 시험결과 요약	프로그램 일람표
통합 및 통합시험	통합 및 통합시스템 시험보고서	통합방법 선정 통합절차 선정 시험의 목적 및 유형 시험방법과 사례 결정 시험결과 요약 시험의 목적 및 유형 시험방법과 사례 결정 시험결과 요약	

5. 결론

본 연구에서는 시스템통합이라는 명제에 주안점을 두고, 개발자를 위한 소프트웨어 과제의 개발 모형과 전체 시스템의 관리자를 위한 관리 모형을 제시하였다. 아울러 관리와 평가를 위한 목적으로 사용될 수 있는 각 단계에서 요구되어질 산출물의 종류가 제안되었다. 앞으로 더욱 복잡해지고 고도화될 대형의 정보시스템 개발을 위해서는 이와 같은 관리 및 개발 방법에 대한 연구가 더욱 심화되어야 할 것으로 생각된다.

참고 문헌

[1] 시스템공학연구소, *소프트웨어 자동생산기술에 관한 연구*, 과학기술처, 1993.

[2] 이진주 외 4인, *사용자 중심의 경영관리시스템*, 다산출판사, 1992.

[3] IEEE Std 1058.1-1987, "IEEE Standard for Software Project Management Plans," Institute of Electrical and Electronic Engineers, Inc., 1987.

[4] J.P. Martin and C. McClure, *Diagramming Techniques for Analysis and Programmers*, Prentice-Hall, 1985.

[5] P. C. Jorgensen and C. Ericson, "Object-Oriented Integration Testing", *Communcation of the ACM*, Vol. 37, No. 9, Sep. 1994.

[6] R.E. Fairley, *Software Engineering Concepts*, McGraw Hill, 1985.

[7]. S. R. Schach, *Software Engineering*, Irwin Inc. & Asken Associates Inc., 1993.

[8] 이주헌, *전략정보시스템구축론*, 푸른산, 1991.

[9] E. Yourdon, *Modern Structured Analusis*, Yourdon Press, 1988.

[10] 이규봉, 서효원, *시스템통합운용기술 (G7 과제 연구개발 보고서)*, 통상산업부, 1995.

* 이 논문은 통상산업부의 선도기술개발사업 중 첨단생산시스템 개발사업에 의하여 연구되었음.