Legacy 시스템의 컴포넌트화를 위한 재공학 방법론

김철홍*, 차정은**, 양영종***

한국전자통신연구원 SW,컨텐츠기술연구부

A Re-engineering Methodology for Componentization of Legacy System

Kim, Cheol-Hong, Cha, Jung-Eun, Yang, Young-Jong

ETRI SW · Contents Technology Department

E-mail : kch@etri.re.kr, mary@etri.re.kr, yangyj@etri.re.kr

요 약

본 논문은 레거시 시스템을 새로운 시스템 환경으로의 변환 및 통합을 위한 재공학 방법론을 제시하고자 하는 것이다. 기존의 역공학 및 재공학 방법들은 주로 프로그램 소스를 정적 분석하고 유지보수 하는데 중점을 두었으나, 본 방법론은 다양한 추상화 수준에서 역공학 정보를 복구하고, 컴포넌트화 단계를 통해 새로운 시스템으로 진화할 수 있는 절차 및 기법들을 제공한다. 레거시 시스템 컴포넌트화를 위한 방법론은 변환계획 단계, 역공학 단계, 컴포넌트화 단계, 인도 단계의 4 단계로 구성되어 있으며, 본 논문에서는 각 단계에 대한 활동 및 상세 절차에 대하여 기술한다.

1. 서론

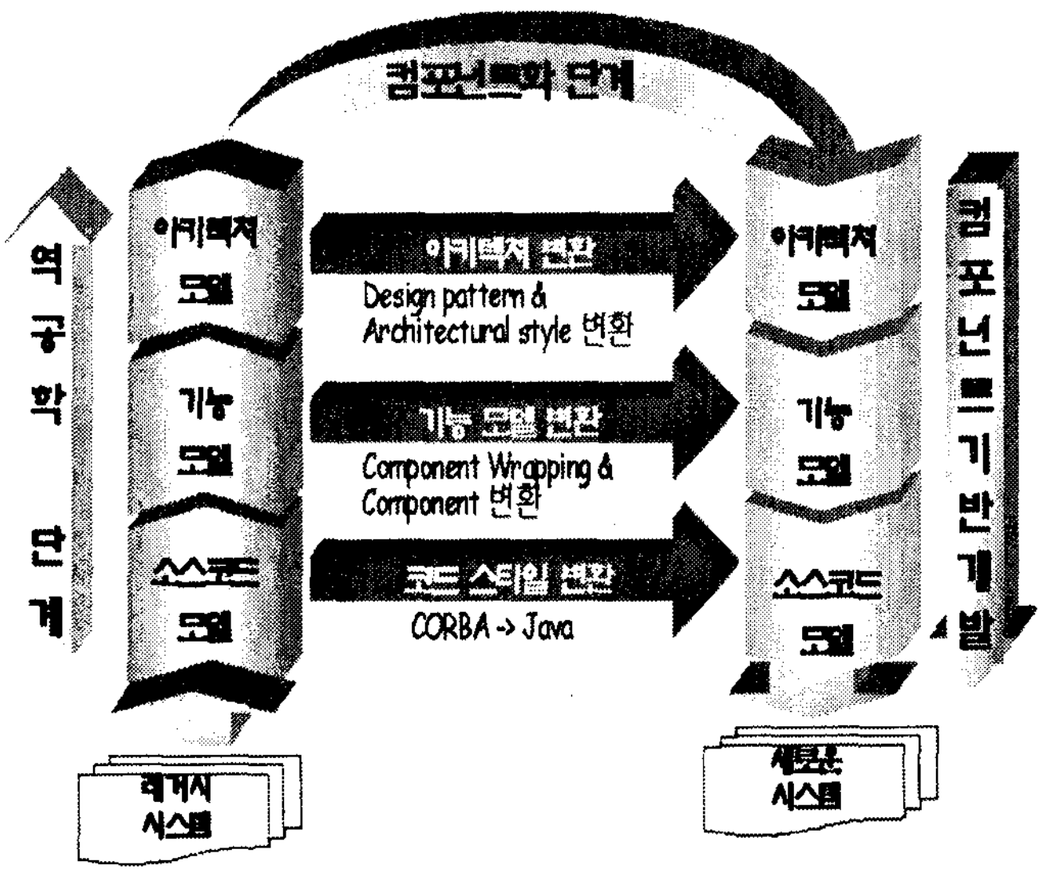
최근 Y2K 문제 해결 이후 기업들은 레거시(Legacy) 시스템을 안정적이고 핵심적인 비즈니스 규칙(Rule)을 포함하고 있는 기업의 소중한 자산으로 인식하기 시작했다. 레거시 시스템은 기업의 핵심 비즈니스 프로세스를 운용하기 위해 설계 및 구현된 중요한 시스템이다. 레거시 시스템의 가장 중요한 가치는 이들 시스템이 가지고 있는 지식(Knowledge)이다. 이러한 지식은 시스템이 다루는 데이터와 비즈니스 프로세스를 포함하고 있다. 현재 전 세계적으로 10,000개 이상의 사이트의 IBM 메인프레임 상에서 운용되는 프로그램 코드가 약 2,000억 라인 정도 되는 것으로 추산되고 있으며, 기업 예산의 85~95%가 레거시 시스템의 유지보수 및 운영에 쓰이고 있는 것으로 보고되고 있다. 그러나 레거시 시스템은 다음과 같은 한계점들은 가지고 있다[1].

- 워크플로우와 비즈니스 프로세스의 밀접한 결합
- 하나의 플랫폼에서 운영될 수 있게 설계
- 변경이 어려우며, 우연성 부족
- 개방성 부족

오늘날 레거시 시스템은 e-business라는 새로운 비즈니스 환경을 수용해야 하는 상황에 처해 있다. 이러한 비즈니스 환경들은 시스템 개발의 새로운 페러다임을 요구하고 있는데, 즉, J2EE, .NET 등으로 대표되는 컴포넌트 및 웹 서비스 기술의 적용이다. 레거시 시스템이 이러한 기술들을 수용할 수 있는 방법은 크게 변환(Transformation)과 래핑(Wrapping) 방법으로 나눌 수 있다. 변환은 기존 시스템을 새로운 언어 및 플랫폼 환경으로 이전하는 것이며, 래핑은 기존 코드의 내부를 숨기고 잘 정의된 인터페이스를 통해 외부 시스템과 연계할 수 있는 기술로 기존 시스템을 변경하지 않고 새로운 시스템과 연동 할 수 있는 비용이 저렴하고 안정적인 기술로 기업들이 가장 선호하고 있는 방법이다.

기존의 재공학을 지원하는 도구 및 기법들은 대부분 소스 코드에 기반을 둔 데이터 흐름이나 제어 흐름을 분석하여 메트릭을 제공하는 정적분석 및 재문서화 기법의 연구에 중점을 두었다. 그러나 방법론 차원에서 재공학의 절차와 기법을 체계적으로 정의하고자 하는 노력은 미흡하였다. 따라서 많은 기업들이 재공학 프로젝트를 추진함에 있어서 시행 착오를 겪고 있다.

본 연구는 대규모 레거시 시스템을 컴포넌트 기반 시스템으로의 체계적 변환 및 통합을 위한 프로세스와 기법을 제공하고자 하는 것이다. 현재 방법론 프로토타입이 개발되어 있으며, 본 논문에서는 방법론의 개념과 상세 절차에 대하여 서술한다.



(그림 1) 레거시 시스템 컴포넌트화 방법론 개념 모델

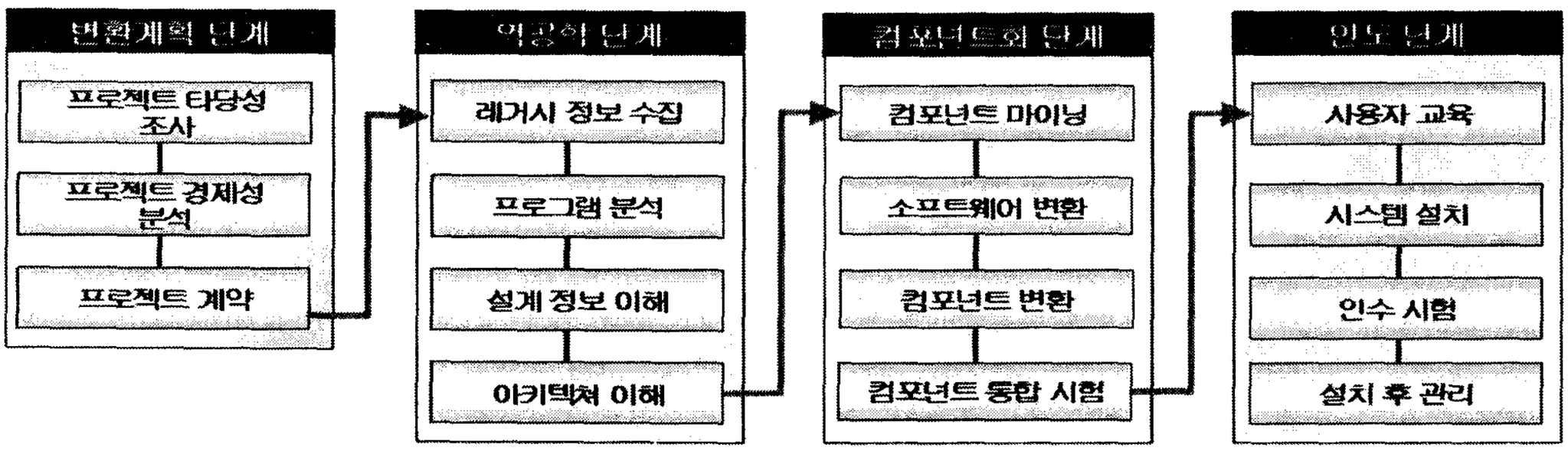
2. 방법론의 개념

본 방법론은 COBOL, RPG, PL/1, C 등으로 작성된 레거시 시스템을 J2EE, .NET 등 컴포넌트 및 웹 서비스 시스템으로 변환하거나 래핑하기 위한 절차와 기법을 제공한다. (그림 1)은 본 방법론의 개념 모델을 보여준다. 역공학 단계는 최하위 수준의 소스 모델로부터 기능 모델, 아키텍처 모델로 추상화된 정보를 복구한다. 소스 모델은 프로그램 소스를 분석하여 텍스트 및 AST 정보를 생성한다. 기능 모델은 소스 모델보다는 추상화된 설계 정보인 구조도 및 데이터 흐름도 등으로 표현하며, 아키텍처 모델은 기능 모델을 그룹핑 및 필터링하는 과정을 거쳐 좀 더 추상화된 정보인 컴포넌트(서브시스템)와 그들간의 인터페이스 관계를 표현한다.

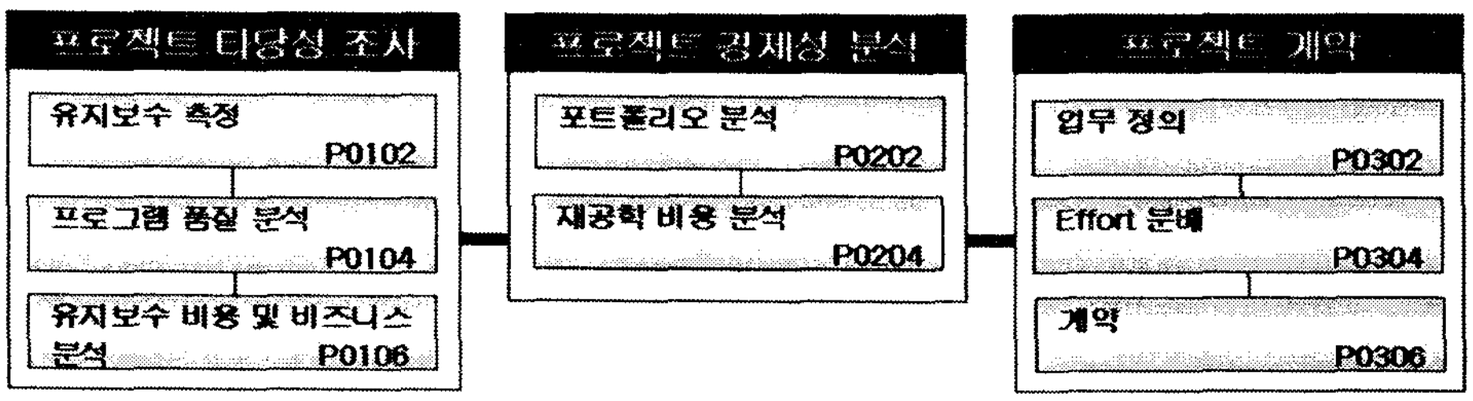
컴포넌트화 단계는 역공학의 각 수준에서 조직내의 역량에 따른 변환 프로세스를 지원한다. 가장 낮은 단계의 변환인 코드 스타일 변환은 단순히 프로그램 언어간의 변환을 지원하며, 기능 모델 변환은 래핑과 DB 스키마 변환이 전형적인 사례이며, 아키텍처 변환은 복구된 레거시 시스템 아키텍처를 새로운 아키텍처로 변환하는 프로세스이다. 컴포넌트 기반 개발은 마르미-III와 같은 순공학 방법론으로 각 변환 단계에서 마르미-III와 연계될 수 있는 방안을 제시한다.

3. 방법론의 상세 절차 및 주요 산출물

본 방법론은 (그림 2)에서 보는 것과 같이 변환계획 단계, 역공학 단계, 컴포넌트화 단계, 인도 단계 등 4 단계로 구성되어 있다. 각 단계는 활동으로 구성되어 있으며, 활동은 하위 업무(Task)들로 구성되고, 업무는 세부 절차 및 기법으로 정의되어 있다.



(그림 2) 방법론 전체 단계



(그림 3) 변환 계획 단계의 활동 및 작업

3.1 계획 단계

대규모 재공학 프로젝트의 성공은 계획이 적절히 수립되었느냐에 따라 좌우된다. 경영층은 비용의 투자를 최소화하고 최대의 부가가치를 얻기를 원한다. 따라서 기대되는 품질과 생산성 향상 여부에 대한 심도 있는 분석과 예측이 필요하다. 이러한 관점에서 본 단계는 프로젝트 타당성과 경제성 분석 등에 대한 절차 및 기법을 제시한다. 계획단계는 (그림 3)과 같이 3개의 활동과 8개의 업무로 이루어져 있다.

3.1.1 프로젝트 타당성 조사

프로젝트 추진의 타당성을 위해서 기존 SW 프로덕트, 유지보수 프로세스, 어플리케이션의 비즈니스 가치를 분석한다. SW 품질이 어느정도까지 향상될 수 있는지, 유지보수 프로세스의 효율성 및 비즈니스 가치가 어느 수준까지 개선될 것인지 등 재공학에 대한 투자 회수율을 평가한다. <표 1>은 프로젝트 타당성 조사 활동을 구성하고 있는 작업, 절차, 주요 산출물들을 요약한 것이다.

<표 1> 프로젝트 타당성 조사 활동의 작업 및 절차들

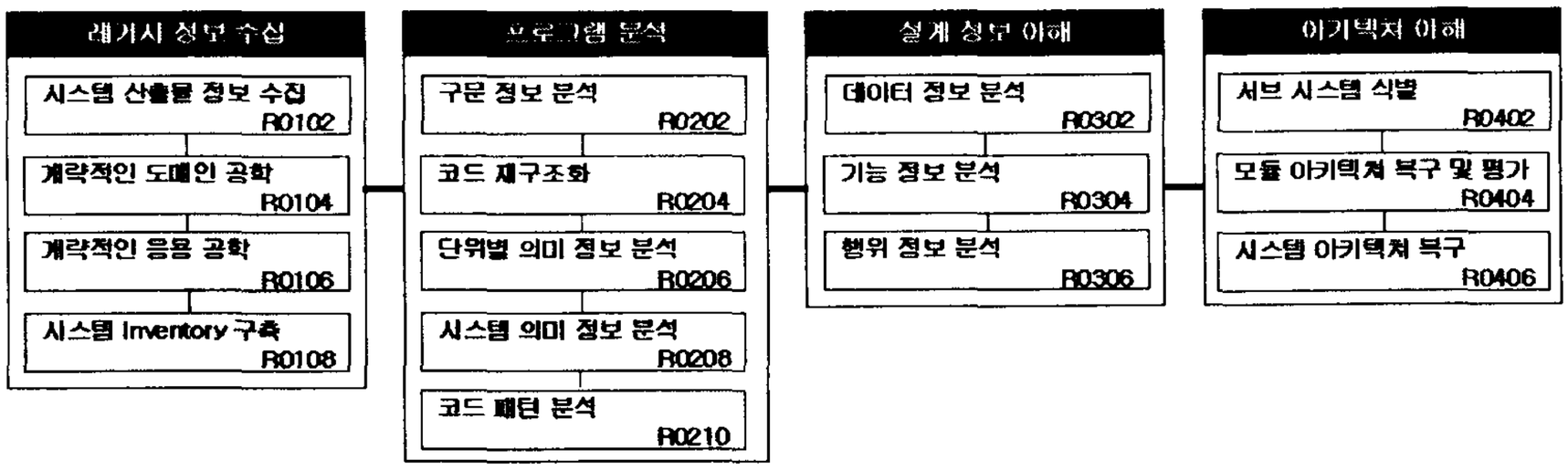
작업	작업 개요	절차	주요 산출물
유지보수 측정	유지보수하고 있는 시스템의 생선성, 신뢰성 및 유지보수성을 측정한다. 측정 메트릭을 정의하여 재공학을 통해서 얻을 수 있는 시스템 향상을 양적으로 측정한다.	① 유지보수 측정 메트릭 정의 ② 측정 메트릭 프로그램 선택/실행 ③ 유지보수 측정 메트릭 분석	**유지보수 메트릭 분석서** 현재 시스템의 데이터 처리 오퍼레이션 상태에 대한 양적 측정 메트릭 정보 제공
프로그램 품질 분석	소스 프로그램, DB, 작업제어 프로시저, 통신 인터페이스 등을 분석하여 기본적인 특정요소들을 수집하여, 크기, 복잡도, 품질 메트릭을 분석한다.	① 품질 메트릭 정의 ② 품질 분석 프로그램 선택/실행 ③ 품질 분석 데이터 분석	**프로그램 품질 메트릭 분석서**: 소스 프로그램, DB 구조, 작업제어 프로시저, 통신 인터페이스, 데이터 사전 등에 대한 질적, 양적, 복잡도 등에 대한 정보 제공
유지보수 비용 및 비즈니스 분석	기존 유지보수 프로세스와 새로운 유지보수 환경의 효과를 평가한다..	① 유지보수 업무 유형 정의 ② 유지보수 프로세스 측정 메트릭 정의	**유지보수 비용 분서서** 기존 유지보수 비용과 새로운 환경의 유지보수 비용 비교평가 정보 제공

<표 2> 프로젝트 경제성 분석 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
프트폴리오 분석	프로그램의 기술적 품질과 비즈니스 가치에 따라 재공학 대상이 되는 어플리케이션 후보들의 우선 순위를 결정한다.	① 기술적 품질 분석 ② 비즈니스 가치 분석 ③ 재공학 대상 우선 순위 결정	**포트폴리오 차트** 기술적 품질과 비즈니스 가치에 대한 가중치에 따른 재공학 업무 우선 순위에 대한 정보 제공
재공학 비용 분석	재공학을 위한 비용을 산정하고, 비용 대 효과를 분석한다..	① 비용 산정 ② 비용 대 효과 분석	**재공학 비용 산정 보고서**: 재공학 대상이 되는 각 프로그램의 코드에 대한 비용 산정에 대한 정보 제공

<표 3> 프로젝트 계약 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
업무 정의	재공학 대상이 되는 객체들로부터 수행되어야 할 업무 목록들을 작성하고, 업무 수행 방법들을 정의한다..	① 업무 목록 작성 ② 업무 수행 방법 정의	**업무 목록** 재공학 대상이 되는 업무 목록 정보 제공
Effort 분배	재공학 프로젝트 수행 및 완료 후 시스템 시험을 위한 인원 및 작업을 할당하기 위한 전략을 수립한다..	① 수행 전략 수립 ② 프로그램 규칙 및 시험 전략 수립	**프로젝트 수행 전략 계획서**: 프로그래밍 규칙, 시스템 시험 계획, 인원 분배 등에 대한 정보 제공
계약	재공학 업무정의와 Effort 분배결과를 토대로 계약조건을 협의하고, 계약을 체결한다..	① 계약 조건 협의 ② 계약 체결	**계약 합의서** 프로젝트 계약 방식, 비용, Effort 분배 등에 대한 조건들을 기술한 문서



(그림 4) 역공학 단계의 활동 및 작업

3.1.2 프로젝트 경제성 분석

기술적 측면에서 시스템 품질과 비즈니스 가치에 따라 재공학 대상이 되는 어플리케이션 후보에 대한 우선 순위를 결정하고, 재공학 대상이 되는 모든 SW 구성요소를 인식하고 중요도에 따라 재공학 프로젝트 비용을 산정하고 비용 대 효과를 분석한다. <표 2>는 프로젝트 경제성 분석 활동을 구성하고 있는 작업, 절차, 주요 산출물들을 요약한 것이다.

3.1.3 프로젝트 계약

프로젝트 타당성 조사와 비용 효과 분석이 완료되면 최종적으로 프로젝트 계약을 해야 한다. 이를 위해 수행 할 업무를 상세히 정의하고 이에 따른 비용을 정확하게 산정하고, 최종 계약을 체결한다. <표 3>은 프로젝트 계약 활동을 구성하고 있는 작업, 절차, 주요 산출물들을 요약한 것이다.

3.2. 역공학 단계

본 단계는 레거시 시스템에 대한 산출물들을 수집하고, 원시 코드 중심의 분석 작업을 수행함으로써, 레거시 시스템에 대한 이해를 높이는 것이다. 따라서 레거시 시스템 요소들 간의 관계성을 인식하여 기능 및 행위 정보를 식별하는 코드의 의미 분석과 분석 결과들을 설계 정보로 추상화하는 모델링 작업이 수행된다[2, 3]. 역공학 단계는 (그림 4)에서 보는 것과 같이 4개의 활동과 15개 작업으로 구성된다.

3.2.1 레거시 정보 수집

본 활동에서는 레거시 시스템의 신출물 및 참조할 수 있는 도메인 정보를 수집하여 레거시 시스템에 대한 Inventory를 구축함으로써, 재공학을 위한 준비 작업들이 수행된다. 또한 도메인 전문가 및 시스템 전문가의 도움을 받아 레거시 시스템과 관련된 도메인의 일반적인 지식을 수집하고, 이를 레거시 시스템의 분석 과정에 반영한다. 각 작업에 대한 개략적인 설명이 <표 4>에 제시된다[4].

3.2.2 프로그램 분석 활동

원시 코드의 분석과 재구조화를 수행하는 전형적인 역공학 프로세스이다. 따라서, 레거시 프로그램을 시스템 수준과 단위 프로그램 수준에서 분석하고, 코드 패턴을 식별하며, 비구조적 코드들을 재구조화 한다.

<표 4> 레거시 정보 수집 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
시스템 산출물 정보 수집	레거시 시스템의 개발 계획서, 요구 사항 정의서, 설계 문서, 원시 코드, 실행 코드, 매뉴얼 등 시스템 개발 과정에서 생산된 산출물 문서 및 시스템 운영과 유지보수에 관련된 모든 문서를 수집, 분류하는 작업이다	① 레거시 시스템 산출물 수집 준비 ② 산출물 정보의 수집 및 분류 ③ "산출물 리스트" 작성 ④ 산출물 정보 분석서 작성	**산출물 리스트** : 수집해야할 레거시 시스템의 산출물 리스트이며, 실제 수집, 획득한 산출물의 양과 품질에 대한 상태를 분석한 결과 그 상태를 기록
개략적인 도메인 공학	레거시 시스템 개발시, 개발자들이 부여한 지식들을 파악하고, 재공학 단계에서 사용할 수 있도록 추상화하기 위해, 레거시 시스템에 관련된 개발자, 운영자의 노하우와 그들의 경험 및 지식을 수집하고, 해당 도메인에 대한 지식들을 수집한다.	① 참조 도메인 선정 ② 도메인 및 시스템 지식의 수집 ③ 산출물 정보 분석서 검토 ④ "레거시 시스템 프로화일" 작성	**레거시 시스템 프로화일** : 레거시 시스템의 핵심 이슈 및 주요 기능성, 특별한 제한 사항들을 파악하고, 도메인에 대한 일반적인 특성 및 요구사항, 제약 조건들을 기록
개략적인 응용 공학	레거시 시스템을 구성하고 있는 프로그램들의 유형을 파악하는 업무로서, 레거시 시스템을 보다 효율적으로 분석하기 위한 기법을 적용하기 위한 업무이다.	① 도메인 및 시스템 정보 분석 ② 시스템 시범 운영 ③ 시스템 유형 판정	**레거시 도메인 정보 분석서** : 레거시 시스템의 핵심 기능 블록과 이벤트 흐름, 제약조건 등을 파악하여 문서화.
시스템 Inventory 구축	전 단계를 거치는 과정에서 파악한 정보들을 분류하고, 정제하는 추상화 과정을 거쳐 의미 있는 재공학 정보 구조로 저장한다. 이를 위해 레거시 산출물 및 참조 정보들의 inventory를 구축한다.	① 수집, 파악된 정보들을 그룹핑 ② 산출물 Inventory 작성	**산출물 Inventory** : 산출물 리스트를 참조하여 산출물 목록들을 "개발 단계에 따른 추상화 정도"와 "참조 정보", "도메인 정보"의 3가지로 분류하여 정리

<표 5> 프로그램 분석 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
구문 정보 분석	레거시 프로그램을 파싱하고, 분석 정보 테이블을 생성하며, Abstract Syntax Tree (AST)를 구성함으로써, 레거시 시스템의 이해 정보 추출을 위한 기초를 마련한다	① 레거시 시스템의 어휘 분석 ② Abstract Syntax Tree (AST) 생성	구문 분석 결과물 · 파서 트리 · 심볼 테이블 · AST
코드 재구조화	프로그램의 로직들을 구조화시키는 업무를 수행함으로써, 레거시 시스템의 이해를 향상시키고, 재공학의 생산성 향상을 도모한다.	① 비구조적인 부분을 찾아 대체할 다른 제어 구조 선택 ② 선택된 제어문으로 대체 ③ 중복된 기능의 모듈의 제거 ④ 레거시 프로그램을 재순서화	구조화된 코드
단위별 의미 정보 분석	레거시 시스템의 개별 단위 모듈을 대상으로 단위 모듈간의 호출관계, 데이터와 제어흐름 등의 정보가 분석된다.	① 데이터 정보 파악 ② 모듈간의 호출관계 파악 ③ 단위별 의미 정보 파악	단위 의미 정보 분석서 프로그램 내의 서브루틴(파라그라프)간의 호출관계 및 파라그라프 내부의 제어흐름 관계 표현
시스템 의미 정보 분석	시스템 전체의 의미정보를 파악한다. 시스템을 구성하는 모듈간의 제어 흐름 및 계층 구조가 파악된다.	① 시스템 자원 정보 파악 ② 프로그램 호출관계 파악 ③ 프로그램과 파일(테이블)간의 참조 테이블 작성 ④ 시스템 전체의 계층구조 파악	시스템 의미 정보 분석서 프로그램과 파일간의 참조 테이블, 시스템 계층구조, 프로그램간의 호출 구조
코드 패턴 분석	레거시 시스템의 도메인에서 공통적으로 적용될 수 있는 재사용 개념들을 바탕으로 레거시 시스템에서 UI 부분, 비즈니스 로직 부분, DB 부분들을 분리한다.	② 코드 패턴 정의 ② 레거시 프로그램을 탐색 ③ 코드 패턴 분석서 작성	코드 패턴 분석서

<표 6> 설계 정보 이해 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
데이터 정보 분석	레거시 시스템을 구성하고 있는 주요 데이터 구조에 대한 정보를 추출하여 정적인 구조를 이해하기 위한 업무이다.	① 관계 클러스터링 파악 ② 추상 요소들의 정제 ③ 최종 스키마 작성	E-R 다이어그램 : Relation과 엔티티를 간의 계층 구조를 파악하고, 엔티티 타입을 분류하고, 관계성을 식별
기능 정보 분석	프로그램 분석 단계에서 추출된 정보를 기반으로 레거시 시스템들의 기능성을 그룹화된 단위로 추상화시킨다.	① 상위 Use Case 추출 ② 하위 Use Case 추출 ③ Actor 추출 ④ Association 추출	Use Case 다이어그램 : 프로그램 및 하위 서브루틴을 중심으로 Use Case를 식별하고 관련된 Actor와 그들 간의 관련성을 식별
행위 정보 분석	시스템 외부 혹은 내부로부터의 사건, 시간의 경과 등으로 시스템의 주요 프로세스가 어떻게 호출되어졌는지를 이해하기 위한 작업이다.	① 시스템 전체를 레벨 0으로 표현 ② 기능 정보의 상위 Use Case를 레벨1 프로세스로 표현 ③ 레벨 1과 2의 제어 흐름 표현 ④ 프로세스에 대한 Data Store를 정의하고 이들간의 데이터흐름 정의 ⑤ 하위 레벨을 대상으로 ①~④를 반복	Data Flow 다이어그램 : 단위 기능 모듈들을 프로세스로 정의하고 테이블로의 접근 관계를 고려하여 Data Store를 정의하여, 프로세스간의 제어 흐름과 데이블에 대한 데이터 흐름을 DFD 형식으로 표현

<표 5>는 프로그램 분석 활동을 구성하는 작업들과 절차, 그리고 주요 산출물들을 나타낸 것이다.

3.2.3 설계 정보 이해 활동

본 활동에서는 앞 단계의 프로그램 분석 정보들을 토대로 레거시 시스템의 정적 구조 및 주요 기능과 행위들을 식별하고 그룹핑 한다. 이는 레거시 시스템에 대한 설계 정보를 모델링 하고 구조적인 다이어그램으로 추상화시킴으로써 더 높은 이해를 획득하기 위한 것이다. 전체 3개의 작업과 12개의 절차로 구성되며, 개요적 특징들은 <표 6>에 설명된다. 설계 정보 이해 활동에서는 기능적인 단위 프로세스들을 식별하고 이들간의 제어 흐름과, 관련된 테이블과의 데이터 흐름을 명확히 하고, 다음의 아키텍처 이해 활동을 위한 시스템 설계 정보를 제공한다.

3.2.4 아키텍처 이해 활동

아키텍처 이해 활동은 레거시 시스템으로부터 독립적인 프로세스인 서브 시스템들을 식별하고, 이들간의 인터페이스 및 호출 관계를 계층 구조로 표현한 모듈 아키텍처를 복구한다. 레거시 시스템의 하드웨어 및 소프트웨어 환경을 파악함으로써, 물리적인 시스템 아키텍처를 유도해 내는 작업들을 포함한다.

<표 7>는 전체 3개의 작업과 7개의 절차로 구성되는 아키텍처 이해 활동의 특징들을 요약한 것이다.

<표 7> 아키텍쳐 이해 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
서브 시스템 식별	시스템을 구성하는 프로그램 및 서브루틴들 간의 호출 및 사용 관계를 분석한 정보들을 그룹핑하고 필터링하여 서브시스템으로 패키징 하는 작업이다.	① 프로그램간의 호출 간계 참조 ② 프로그램과 파일 간의 CRUD 메트릭스 분석 ③ 서브 시스템 정제	**서브 시스템** : 프로그램 호출 관계와 CRUD 메트릭스를 통해 추출된 후보 서브시스템 중 시스템 전문가의 판단으로 최종적인 서브 시스템을 제시
모듈 아키텍쳐 복구 및 평가	식별된 서브 시스템들 간의 호출 및 참조 관계를 분석하여 서브 시스템의 계층 구조를 표현하는 모듈 아키텍쳐를 복구한다.	① 모듈 아키텍쳐 복구 ② 모듈 아키텍쳐 검증 및 평가	**모듈 아키텍쳐 다이어그램** : 서브 시스템들 간의 호출관계 및 코드의 디렉토리 구성 정보로부터 트리 구조의 계층적 모듈 구조를 제공
시스템 아키텍쳐 복구	레거시 시스템을 받쳐주고 있는 하드웨어 시스템 구성도를 이해하고, 요구되는 S/W 환경을 파악한다.	① 전체 시스템 구성도 작성 ② S/W 환경 이해	**시스템 아키텍쳐 다이어그램** : 프로세스를 나타내는 노드 사이에 프로토콜과 연결 정보가 명시된 컨넥션을 표현하며, 개별 노드에서 실행되는 모듈들을 컴포넌트로 표현

3.3 컴포넌트화 단계

컴포넌트화 단계는 역공학 단계에서 복구된 산출물을 토대로 컴포넌트를 식별, 추출하여 목표 시스템의 환경에 맞게 변환하거나 래핑할 수 있는 절차 및 기법을 제공한다. 본 단계의 핵심은 레거시 시스템으로부터 컴포넌트를 식별, 추출해내는 컴포넌트 마이닝과 추출된 컴포넌트들을 새로운 재공학 시스템에 적용하기 위해 변환 전략을 수립한 후, 시스템 아키텍쳐를 정의하여 실제 비즈니스 컴포넌트를 개발하는 것이다. (그림 5)는 컴포넌트 단계의 전체 활동 및 각 활동에서 수행되는 작업들을 나타낸다.

3.3.1 컴포넌트 마이닝

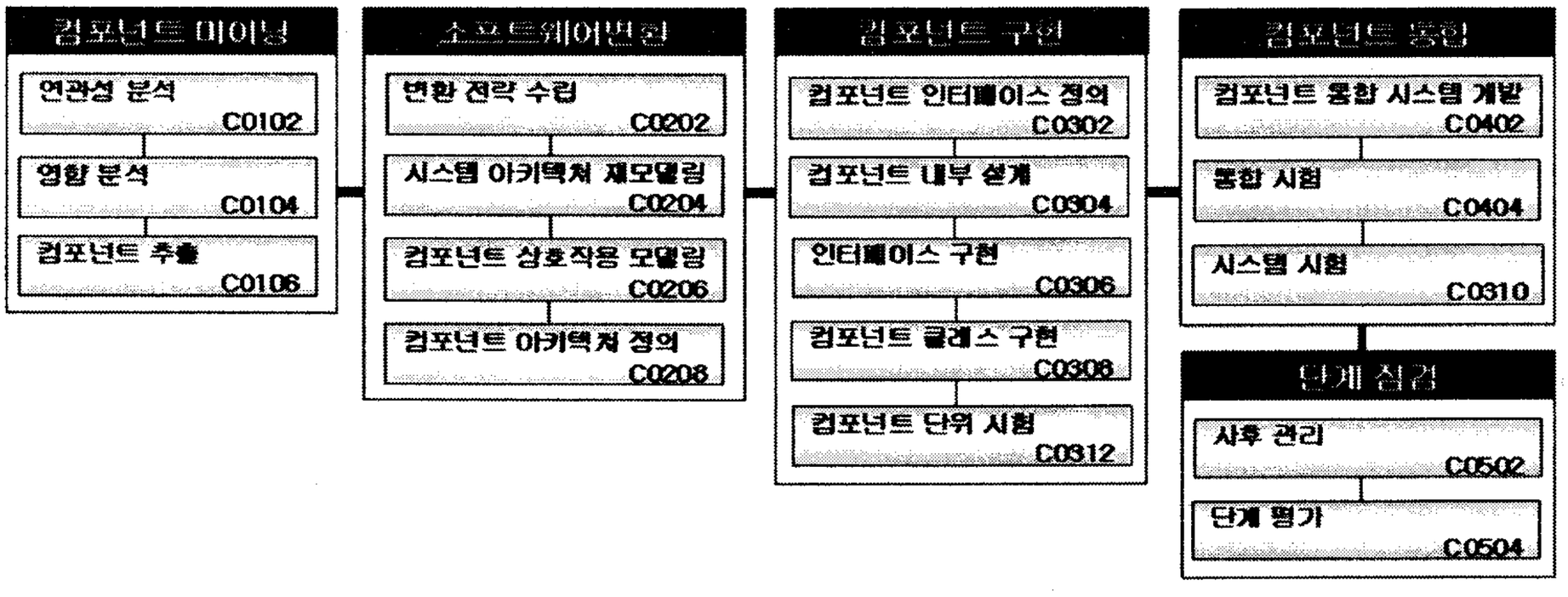
역공학 단계 과정을 통해 추출된 정보를 바탕으로 연관성이 높은 시스템 요소들을 컴포넌트화 하기 위한 것이다. 입력물로는 전 단계에서 산출되어진 프로그램 목록, 파일 목록, 호출 그래프(Call Graph), 제어 흐름 그래프, 시스템 흐름 그래프 등이 사용된다. <표 8>는 컴포넌트 마이닝 활동을 구성하는 작업들과 절차, 주요 산출물들을 나타낸 것이다.

3.3.2 소프트웨어 변환

레거시 시스템을 재공학 하기 위한 방법과 이를 성공적인 수행하기 위한 전략을 설정하고, 추출된 재사용 요소들의 컴포넌트화 기법을 정의한다. 이를 위해 재공학 요구를 분석하여, 목표로 하는 새로운 환경을 정의하고, 재공학 시스템의 전체 아키텍쳐를 재모델링한다. 상호작용 모델링을 통해 컴포넌트들에 수직적, 수평적 계층 관계를 파악하여, 비즈니스 컴포넌트에 대한 컴포넌트 아키텍쳐를 설계한다. <표 9>는 본 활동의 특징들로 주요 산출물로는 결정된 재공학 전략에 따른 목표 재공학 시스템의 소프트웨어, 시스템, 컴포넌트 아키텍쳐가 제공된다.

3.3.3 컴포넌트 구현

본 활동에서는 추출된 컴포넌트와 식별된 컴포넌트들 간의 협력 관계성 및 컴포넌트 아키텍쳐를 바탕으로, 새롭게 정의된 시스템 환경에서 운영 가능한 컴포넌트를 설계, 구현한다. 구체적인 작업으로 컴포넌트 인터페이스 정의 및 구현, 컴포넌트 내부 구조 설계, 컴포넌트 클래스 구현, 컴포넌트



(그림 5) 컴포넌트화 단계의 전체 활동 및 작업

<표 8> 컴포넌트 마이닝 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
연관성 분석	역공학 단계의 산출 정보들을 이용하여 시스템 구성 요소들 간의 연관 관계를 파악하고 모델링 한 후, 모델로부터 연관성 높은 요소들을 그룹화 하여 컴포넌트의 핵심 요소로 인식한다.	① 시스템 구성 요소 파악 ② 시스템 구성요소들 간이 연관 관계 파악 ③ 컴포넌트 핵심 요소 파악	**연관 관계 모델링 테이블** : 호출 관계, 정의/사용 관계, 제어 흐름 관계 등의 시스템 의미 정보와 온라인 작업, 배치 작업들을 의미하는 작업의 특성 정보를 제공
영향 분석	컴포넌트 핵심 요소에 영향을 주고받는 부분을 파악하기 위해, 연관성 정보의 영향 범위를 파악하여, 그 범위의 요소들 중, 응집력이 높은 것들을 그룹화하여 컴포넌트를 식별한다.	① 컴포넌트 후보 구성요소 파악 ② 컴포넌트간 공유 요소 파악 ③ 공유 요소의 평가	**컴포넌트 후보 목록**
컴포넌트 추출	영향분석을 통해 파악된 컴포넌트 후보 구성요소들을 각각의 컴포넌트로 구축하는 과정이다..	① 공유요소의 컴포넌트화 ② 트랜젝션 구성요소의 컴포넌트화 ③ 컴포넌트 세분화 ④ 컴포넌트간의 연관관계 파악	**컴포넌트 목록** 컴포넌트 이름, 컴포넌트 기능, 다른 컴포넌트와의 관계, 운영환경, 요구조건, 제약사항 정보 제공

<표 9> 소프트웨어 변환 활동의 작업 및 절차들

작업	작업 개요	절차	주요 산출물
변환 전략 수집	재공학 범위와 방법을 결정한다. 또한 추출된 컴포넌트들의 컴포넌트화 전략과 기법들을 정의하며, 그 타당성을 검토한다, 따라서 재공학 요구를 분석하는 작업과 변환 유형의 분석 및 변환 전략 수립한다.	① 재공학 요구 분석서 작성 ② 변환 유형 선정 ③ 변환 전략 수립 ④ 변환 전략서 작성	**재공학 요구 분석서** **변환 계획서**
시스템 아키텍쳐 재모델링	S/W 아키텍쳐 스타일을 바탕으로 S/W 아키텍쳐를 정의하고, 재공학 시스템의 하드웨어 및 소프트웨어 환경을 포함하는 새로운 시스템 아키텍쳐를 재정의 하는 작업을 수행한다.	① 참조할 S/W 아키텍쳐 유형 선정 ② 하드웨어 플랫폼 결정 ③ 소프트웨어 플랫폼 결정 ④ 컴포넌트 플랫폼 결정 ⑤ 데이터 변환 모형 작성 ⑥ 시스템 아키텍쳐 재모델링	**S/W 아키텍쳐** **시스템 아키텍쳐 정의서**
컴포넌트 상호작용 모델링	추출된 컴포넌트들이 어떻게 상호 연동하는지를 파악하여, 연관성 있는 논리적 단위로 그룹핑 한다.	① 시스템 Use Case 식별 ② Use Case 담당 컴포넌트 식별 ③ Use case 수행을 위해 식별되지 않은 컴포넌트 파악 ④ 상호작용 컴포넌트 식별 ⑤ 컴포넌트들 간의 상호작용 이벤트 정제 ⑥ 컴포넌트 그룹핑	**컴포넌트 상호작용도** **컴포넌트 그룹**
컴포넌트 아키텍쳐 정의	추출된 컴포넌트들의 관계를 정적인 구현 관점에서 서술하여, 수직적, 수평적 연관 관계를 분석한다. 그 결과 비즈니스 컴포넌트들의 배치와 조립이 가능한 컴포넌트 아키텍쳐를 정의한다.	① 컴포넌트들 간의 협력 계층 파악 ② 시스템 아키텍쳐에 컴포넌트 배치 ③ 배치 컴포넌트들 간의 의존 관계 표현 ④ Glue 컴포넌트 추가 ⑤ 컴포넌트 아키텍쳐 작성	**컴포넌트 아키텍쳐**

단위 시험 등이 수행되며, 산출물로서 컴포넌트 명세와 컴포넌트 구현 코드 그리고 단위 테스트 결과서가 생성된다.

3.3.4 컴포넌트 통합

본 활동에서는 개발된 개별 컴포넌트들을 프로토타이핑 구축을 통해 통합함으로써, 레거시 시스템이 가진 전체 기능성을 발휘 여부와 제약 사항들을 분석하고 점검한다. 이를 위해 변환 전략에 따라 추출된 컴포넌트들을 재공학 시스템 아키텍쳐 상에 배치하고 통합하여, 구현 컴포넌트들이 다른 컴포넌트들과 제대로 통신하는지를 평가하며, 컴포넌트 아키텍쳐와 비즈니스 요구가 잘 정의되고 구현되었는지를 테스트한다. 컴포넌트 통합 시스템 개발과, 통합 시험, 시스템 시험의 3가지 작업으로 구성된다.

4. 결론 및 향후 연구

레거시 시스템의 컴포넌트화 방법론은 변환 계획 단계, 역공학 단계, 컴포넌트화 단계, 인도 단계의 4 단계로 구성된다. 본 논문에서는 변환 계획 단계, 역공학 단계, 컴포넌트 단계에 대한 상세 절차 및 산출물들에 대하여 요약하였다. 역공학 단계는 레거시 시스템 이해를 위한 가장 어렵고 중요한 단계로서 레거시 시스템의 대부분을 차지하고 있는 COBOL 프로그램에 초점을 맞추어 기법을 정의하였다. 또한 컴포넌트 단계는 재사용성 높은 비즈니스 로직 부분을 추출하고, 재공학 전략을 수

립하여, 그에 따라 실제 재공학 시스템을 구축해나가는 과정이다. 컴포넌트화 단계는 역공학 단계에서 복구된 산출물을 토대로 컴포넌트들을 식별, 추출하여 이를 목표 시스템의 환경에 맞게 변환하거나 래핑할 수 있는 절차 및 기법을 제공한다.

향후, 현재 비즈니스 영역에서 잘 운영되는 레거시 시스템을 대상으로 정의되어진 방법론을 적용하여 검증 및 보완하여 나아갈 것이며, 관련 도구에 의한 지원 체계 구축에 대해 연구할 계획이다.

5. 참고문헌

[1]. Hurwitz Group, Integrating Your Business with The Internet: Effectively Transforming Legacy System Assets into Flexible Business Servers, Relativity Technologies, White Paper, Sep 1999.

[2]. SEI Reengineering Center, CMU, "Perspectives on Legacy System Reengineering, 1995

[3] Ivar Jacobson,Grady Booch,James Rumbaugh,Grady Booch,James Rumbaugh, The Unified Software Development Process, Addison Wesley, January 1999

[4]. Susan D. Markel, "Process Definition for Capturing Legacy System Requirement", *Aerospace Conference, 1997. Proceedings.*, IEEE , Volume: 4 , 1997