RUP 기반 CIS 소프트웨어 개발 프로세스

A Software Development Process of Core Instrumentation System Based on the Rational Unified Process

이 길 섭* 이 태 공**
Kil Sup Lee Tae Gong Lee

요 약

RUP(Rational Unified Process)는 객체 지향적이며 사용사례 및 아키텍처 중심의 반복적인 개발방법론이다. 이전에 수행된 대부분 공공분야의 대규모 소프트웨어 개발은 폭포수형 개발 프로세스를 적용하였으나 최근에는 소프트웨어 개발에 따른 위험을 최소화 하고 품질을 향상하기 위하여 RUP와 같은 반복 개발방법의 적용을 시도하고 있다. 그러나 대규모 복합 체계의 개발 프로세스와 그 일부인 소프트웨어 개발 프로세스로서 RUP를 적용한 국내의 연구 자료는 미흡한 실정이다.

본 논문에서는 RUP를 기반으로 복합 체계의 일부인 소프트웨어를 개발하는 프로세스를 고찰하고자 한다. 이를 위하여 국내 CIS 소프트웨어 개발 사례를 통하여 체계개발 프로세스와 통합된 RUP 프로세스를 제시하고 기존의 폭포수형 프로세스 및 RUP와 비교 평가를 한다. 본 논문의 연구결과는 공공기관에서 대규모 복합체계의 소프트웨어를 개발하는 경우에 RUP 기반 프로세스의 조정, 개발관리에 있어서 위험의 최소화와 최종 제품의 품질향상에 기여할 것으로 믿는다.

Abstract

RUP(Rational Unified Process) is a development process which is based on object-oriented, use case centric, architecture centric, and iterative approach. Public projects performed previously adopt waterfall lifecycle model for development of large scale software. However, recently various projects adopt an iterative approach to minimize risks of a project and to enhance quality of software. But few research result on practices of RUP as the subprocess of system development process is available.

This paper presents a system development process which uses RUP as the subprocess for a subset of the system. Thus we introduce a tailored RUP for K-CIS(Korean Core Instrumentation System). Moreover, we assess the application result of K-CIS with typical waterfall lifecycle model and RUP. We believe that the results of our work are useful for tailoring a system development process with RUP, reducing risks of development, and enhancing the quality of a final product.

Keyword : Software Development LifeCycle Model, Waterfall LifeCycle Model, RUP (Rational Unified Process), Project Management, CIS (Core Instrument System)

1. 서 론

사회 전반에 걸쳐서 정보화가 본격화 되면서 소프트웨어의 수요가 급증하고 개발에 의한 소프트웨어 획득도 증가하고 있다. 이에 따라 소프트웨어를 개발하기 위한 패러다임에도 구조적 분석방법[15], 정보공학방법론[11], 객체지향분석방법론[3] 및 컴포넌트기반 개발방법론[2,7,12,14,15]으로 발전이 되어 왔다. 이러한 개발 패러다임과 아울러 소프트웨어의 수명주기 프로세스에도 많은 발전이 있어왔다. 가장 먼저 일괄개발을 목적으로 한 폭포수형(Waterfall) 모델[1]이 보편적으로 사용되어 왔으나, 최근에는 나선형(Spiral), 통합 프로세스(Unified Process) 등 반복 개발을 기반으로 한 수명주기 모델의 적용도 증가되는 추세이다.

이중에서 RUP(Rational Unified Process)[13]는 객체지향적이며, 반복 개발에 기반을 둔 통합 소프트웨어 개발 프로세스이다. 여기에 아키텍처와 사용사례를 기반으로 한 개발 프로세스이기도 하다. 이러한 RUP는 대부분 소프트웨어 중심의 사

* 정 회 원 : 국방대학교 전산정보학과 순환직 교수
gislee@kndu.ac.kr(제 1저자)
** 정 회 원 : 국방대학교 전산정보학과 교수
tglee@kndu.ac.kr(공동저자)

업에 적용되어 왔으며 복합체계 사업에 대한 국내 적용사례는 잘 알려져 있지 않다.

따라서 대규모의 소프트웨어 개발을 위해서 국내 사정에 부합하는 수명주기 모델 및 개발 프로세스의 설정, 사업관리를 위한 계획 수립에 있어 경험적인 자료에 근거하는 체계적인 계획 수립이 제한되어 왔다. 또한 대규모의 소프트웨어는 소프트웨어 단독으로 운영되기 보다는 통신망을 기반으로 한 전산장비 및 기타 특수 목적 장비들이 통합되어 운영이 되므로 별도의 수명주기 모델 및 통합된 개발 프로세스가 요구된다.

본 논문에서는 CTC(Combat Training Center)의 CIS(Core Instrumentation System)의 개발사례를 통하여 RUP를 기반으로 복합체계의 일부로 소프트웨어를 개발하기 위한 프로세스를 고찰하고자 한다. 이를 위하여 객체지향 분석 및 설계의 세부적인 절차와 산출물의 표현 보다는 RUP를 기반으로 공정, 일정, 자원 및 품질 관리 측면에 중점을 두어 기술하고자 한다. 그 결과 RUP를 국방 분야의 체계개발 프로세스와 통합하여 적용하는 표준 프로세스를 제정하는데 기초 자료로 활용될 것이 기대된다.

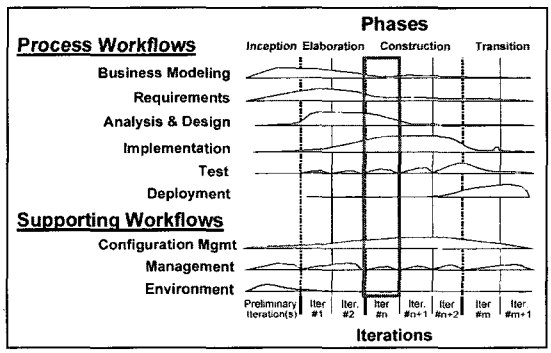
이후 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 RUP 및 CIS 체계에 대해서 살펴보고, 3장에서는 K-CIS 소프트웨어 개발 프로세스에 대하여 기술하고, 4장에서는 K-CIS 소프트웨어 개발에 대한 평가를 논의하며, 마지막으로 결론 및 향후 연구과제에 대해서 서술한다.

2. 관련 연구

2.1 USDP와 RUP

USDP(Unified Software Development Process)[9]는 객체지향적이며, 사용사례(Use case) 및 아키텍처(Architecture) 중심의 반복적(Iterative)인 소프트웨어 개발 및 배치를 위한 프로세스 이다. USDP 내에서는 UML(Unified Modeling Language)[3]을 소프트웨어의 모델을 구축하고 문서화하는 언어로 사용하고 있다. 또한, USDP 개발주기는 개념(Inception), 정교화(Elaboration), 구축(Construction), 전이(Transition)의 4 단계(Phase)로 이루어진다. 또한, 각 단계는 반복(Iteration)으로 분할되며, 각 반복은 다시 요구사항, 분석, 설계, 구현 그리고 시험의 5개 핵심 작업흐름(Core Workflows)[1]으로 구성된다.

이러한 USDP는 Rational사의 RUP(Rational Unified Process)의 축소형으로서 실제 소프트웨어 개발 프로젝트에 적용하기 위해서는 많은 요소들이 필요하게 된다. 여기에는 소프트웨어의 전체 수명주기를 고려하여 업무 모델링(Business Modelling) 작업흐름이 요구사항 작업흐름에 앞서 이루어지고 프로젝트 관리(Project Management)와 형상관리(Change Management), 개발환경(Environment) 등의 추가적인 작업흐름이 존재한다. 이 외에도 각 작업흐름별 가이드라인(Guideline), 산출물 양식(Template), 품질지표(Quality Indicator), 척도(Metrics) 등이 제공되고 있다. 그림 1은 앞에서 설명된 RUP의 개요를 보여주고 있다.



〈그림 1〉 RUP의 개요

2.2 CIS 체계

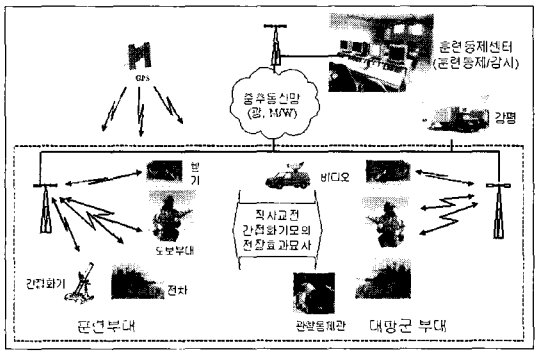
과학기술의 발전에 따라 세계 각국에서는 군사

1) USDP의 Core Workflows는 RUP의 Process Workflows의 부분집합이다. 업무 모델링과 배치 작업흐름이 없으며 분석과 설계가 분리되어 있다.

훈련을 하는 데 있어서 과학기술을 이용하여 실 전장과 유사한 환경을 제공하는 CTC(Combat Training Center)를 구축하여 준 전투 체험을 가능하게 하고, 훈련 동안 자료 수집, 분석 및 사후검토를 통한 훈련효과의 극대화를 추구하고 있다. 이러한 CTC는 미국의 NTC(National Training Center), JRTC(Joint Readiness Training Center), 독일의 GÜZ(Getechts Übungs Zentru) 등 6개국이 이미 구축하여 운용중이며, 한국을 포함한 30여 개국이 구축중이거나 구축을 고려중이다.[19]

CTC를 이루고 있는 제반 설비를 통제하는 중앙통제장비(Core Instrumentation System)는 훈련장과 훈련통제본부내에 있는 하부체계로 구성된다. 먼저, 훈련장에는 각 인원 및 장비에 부착되는 GPS(Global Positioning System), 레이저를 이용하여 교전을 묘사하는 교전훈련장비, 훈련상황을 관찰 및 통제하기 위한 장비, 훈련의 결과를 분석 및 강평하기 위한 장비, 이들 장비들 간의 통신을 위한 무선데이터통신망, 무선중계소간의 통신을 위한 광 및 마이크로웨이브로 이루어진 중추통신망이 있다.

훈련통제본부에는 통신망을 통하여 수집된 교전, 관찰 및 통제 자료를 수집 및 처리를 위한 전산장비, 훈련 계획·준비·실시·분석·사후검토를 위한 훈련통제 소프트웨어, 그리고 일반 행정 및 체계의 운영 및 장애상태를 관리하기 위한 체계지원 소프트웨어로 구성된다. 그림 2는 일반적인 CIS 체계의 개념도 이다.



〈그림 2〉 일반적인 CIS 체계 개념도

3. K-CIS 소프트웨어 개발 프로세스

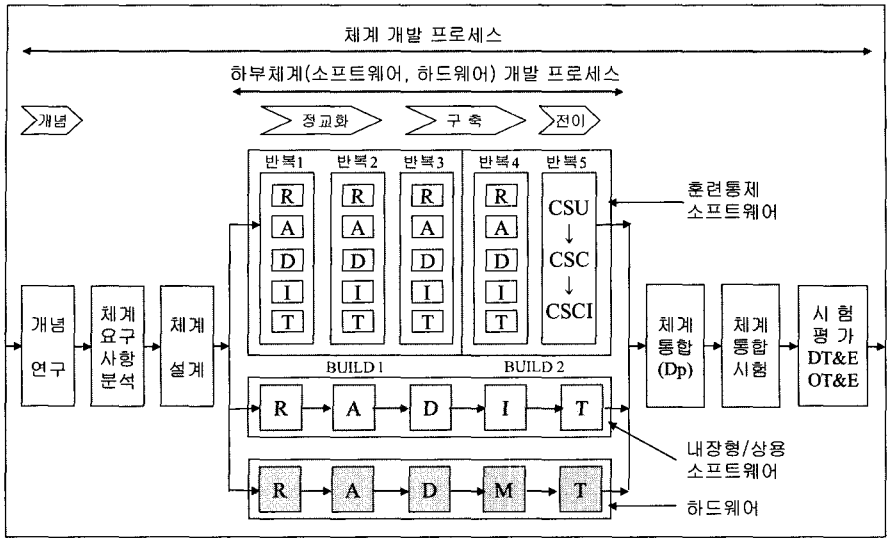
3.1 RUP 적용을 위한 고려사항

본 절에서는 K-CIS 훈련통제 소프트웨어를 개발하는 데 있어서 RUP를 적용하기 위하여 체계의 특성, 개발 프로세스 설정, 적용 프로세스의 효율성 등의 고려사항에 대하여 기술한다. K-CIS 체계는 소프트웨어, 하드웨어, 통신시설, 도로, 건축, 전기 등이 네트워크를 통하여 상호연동 및 통합된 복합체계(System of Systems)이다.

이 중에서 소프트웨어는 일반 응용인 훈련통제 소프트웨어 및 상용위주의 체계지원 소프트웨어, 교전훈련장비, 통신 및 연동장비에 포함되는 내장형(Embedded) 소프트웨어로 구성된다. 또한 훈련을 위해 인원 및 차량의 위치 및 교전정보가 일정시간 간격으로 주기적으로 보고 및 모의가 필요한 근실시간(Near Realtime) 특성을 갖는다.

이러한 복합체계인 K-CIS의 체계개발을 위한 프로세스와 그 일부분인 훈련통제 소프트웨어의 개발을 위한 RUP의 정합을 위하여 체계 프로세스의 요구사항, 설계, 통합 및 통합시험과 RUP의 업무 모델링 (Business Modelling) 시기의 통합, 하드웨어와 병행 개발이 요구되는 내장형 소프트웨어와 상용 소프트웨어의 커스터마이징 (Customizing)을 통한 개발에 RUP 적용 여부 등에 대한 결정이 요구된다.

이 외에도 RUP 내부에서 단계의 설정, 반복의 수, 빌드의 수, 작업흐름별 활동(Activities), 작업(Tasks), 적용 모델(Models) 및 산출물(Artifacts)에 대한 정의가 요구된다. 또한, 점증적이고 반복적 개발을 통한 산출물의 연계와 통합, 아키텍처, 시험, 형상, 위험 및 품질을 관리하는 방안이 필요하다. 마지막으로 적용된 RUP의 효율성을 평가하기 위하여 통계자료를 제시하고 전형적인 폭포수형 프로세스 및 RUP와 비교가 필요하다. 그 결과



범례: R(Requirement), A(Analysis), D(Development), I(Implementation), M(Manufacturing), T(Test), Dp(Deployment), CSCI(Computer Software Configuration item), CSC(Computer Software Component), CSU(Computer Software Unit), DT&E(Development Test & Evaluation), OT&E(Operational Test & Evaluation)

〈그림 3〉 K-CIS 개발 프로세스

에 따라 향후 유사 사업의 개발 프로세스에 적용을 위한 교훈을 도출할 필요가 있다.

3.2 개발 프로세스 설정

앞 3.1절에서 살펴본 바와 같이 K-CIS 체계는 복합체계로 K-CIS 소프트웨어를 포함한 전체 개발 프로세스가 필요하다. 그림 3은 K-CIS 개발을 위하여 결정된 프로세스이다. 소프트웨어를 포함한 체계 개발 프로세스는 미 국방성의 MIL-STD-498[6]으로 알려진 소프트웨어 개발 및 문서화 표준을 적용하였다. 여기에 업무 모델링은 운용구조(OA: Operational Architecture), 체계구조(SA: System Architecture) 및 기술구조(TA: Technical Architecture)를 근간으로 하는 C4ISR AF[2][4]에서 제시하는 업무분석 방법을 적용하였다. 이 방법은 사용사례 중심의 업무 모델링 방법 보다 정밀하고 구체적이라 할 수 있다.

먼저, 훈련통제 소프트웨어의 개념연구는 RUP의 업무 모델링을 대체한다. 전체 빌드는 두 개를 두었으며, 각 반복은 요구사항, 분석 및 설계, 구현, 그리고 시험 등을 포함하여 작업흐름으로 통합되었다. 특히 반복 5단계는 통합을 위하여 별도로 설정된 기간이다.

그리고 소프트웨어 분야에서는 하드웨어와 병행하여 개발이 요구되는 내장형 소프트웨어와 상용제품을 커스터마이징 하는 체계지원 소프트웨어 그리고 통신망 및 단말장비는 폭포수형 프로세스를 적용하여 개발을 한다. 그렇지만, RUP의 업무 모델링과 배치 작업흐름은 체계개발 프로세스에 통합되도록 나타나 있다. 또한, 그림 3의 K-CIS 프로세스에 근거하여 훈련통제 소프트웨어를 위한 활동, 작업 및 산출물이 표 2에 정의되어 있다.

3.3 아키텍처 관리

RUP[12,13]에서는 아키텍처를 설계함에 있어서 4+1 뷰(View)를 적용하는 데, 사용사례(Use Case),

2) 2003년 8월에 DoD AF[5] Ver. 1.0으로 개정되었다. 2004년 4월 발표된 한국군 국방 아키텍처 프레임워크(일명 MND-AF)[20]의 근간이 되었다.

논리적(Logical), 프로세스(Process), 구현(Implementation) 및 배치(Deployment) 뷰가 이에 해당한다. 그림 4는 4+1 아키텍처 모델을 나타내고 있다.

먼저, 사용사례 뷰는 다른 뷰와는 달리 특별한 뷰이다. 분석가가 사용자의 요구사항을 바탕으로 식별된 사용사례 및 액터에 대한 전체 뷰(Global View)를 제시하고 나중에 시험단계에서는 시험자에 의해서 다른 뷰를 검증하는 데 사용된다. 논리 뷰는 최종 사용자 입장에서 사용사례를 실체화(Realization)하여 도출되는 설계 하부 시스템, 패키지 및 클래스를 수용하기 위한 레이어 구조를 제시한다.

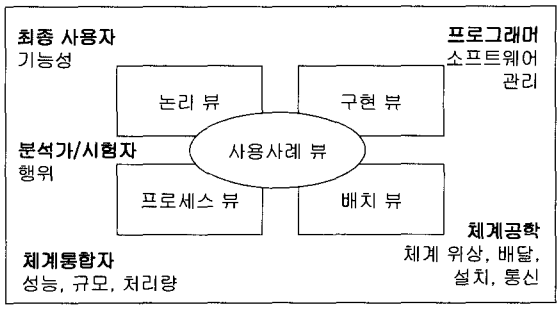
구현 뷰는 프로그래머 입장에서 정적인 소프트웨어 모듈의 조직을 기술한다. 프로세스 뷰는 체계통합자 입장에서 실행시에 체계의 동시적이며 병행적 측면을 접근한다. 배치 뷰는 체계공학적인 측면에서 체계 위상, 배달, 설치 및 통신 측면을 다루며 실행 및 실행시간 컴포넌트가 하부 플랫폼이나 계산 노드에 매핑되는 것을 보인다. K-CIS 소프트웨어 개발에서 적용된 4+1 뷰의 활동 및 산출물이 표 2에 굴림체로 표현되어 있다. 이들에 대한 표현 대상, 표현 수단 및 산출물 간의 관계를 포함한 아키텍처 뷰는 표 3과 같다.

3.4 프로세스 작업흐름 (Process Workflows)

본 절에서는 K-CIS 개발을 위하여 RUP의 업무 모델링, 요구사항, 분석 및 설계, 구현, 시험, 배치 작업흐름에 대하여 적용한 결과를 기술한다.

3.4.1 업무 모델링 (Business Modeling)

업무 모델링은 조직의 구조와 동적인 요소를 이해하고, 현재의 문제점과 개선 잠재성을 식별하여 고객, 최종 사용자, 개발자들이 공통의 인식과 체계의 요구사항을 도출하는 데 있다. 이러한 업무 모델링을 하기 위하여 RUP에서는 업무 비젼, 규칙, 업무 사용사례 모델, 업무 아키텍처 문서, 객체 모델, 보조 업무 명세서 등의 산출물을 작성



〈그림 4〉 4 + 1 아키텍처 모델

토록 하고 있다.

K-CIS 체계개발에서는 소프트웨어 외에도 하드웨어 개발, 상용장비(COTS: Commercial Off-The-Shelf), 관급장비(GOTS: Government Off-The-Shelf) 등이 포함된 점을 고려하여 C4ISR AF[4]를 적용한 개념연구를 통하여 업무의 운영구조, 체계구조, 기술구조를 식별한 후에 MIL-STD-498[6]에서 정의된 산출물인 운영개념기술서(Operational Concept Description), 체계규격서(SSS: System Subsystem Specification) 연동요구명세서(Interface Requirement Specification) 등의 산출물을 작성하였다. 이 과정에서 작성된 C4ISR AF의 산출물은 표 4에 나타나 있다. 총 16종이 제시된 양식에 따라 작성되었으며, 8종은 대체 작성 되었고, 2종은 미작성 되었다.

3.4.2 요구사항 (Requirements)

요구사항 작업흐름의 목적은 고객과 다른 이해당사자들이 시스템에서 무엇이 이루어지고 왜 행해져야 하는지에 대한 합의의 설정 및 유지, 체계 개발자들의 요구사항에 대한 보다 나은 이해, 체계 범위의 정의, 반복에 대한 기술적 사항을 계획, 비용 및 기간 추정을 위한 기반의 제공, 사용자의 요구와 목적에 중점을 둔 사용자 인터페이스를 정의한다.

K-CIS 소프트웨어의 요구사항은 전체 체계의 요구사항을 먼저 작성하고, 체계 및 부체계 설계가 완료된 이후에 소프트웨어 요구사항을 도출하고 세부 요구사항은 구체화를 통하여 요구사항

〈표 2〉 K-CIS-RUP 활동, 작업 및 산출물

작업 흐름	활 동 (Activity)	작 업 (Task)	산 출 물 (Artifact)
요구사항	공통용어 정의		용어집, 약어집
	요구사항 정의	정보수집	요구사항기술서
		요구사항 상세화	
	체계운용 시나리오	하부체계 식별	체계운용시나리오
		체계운용시나리오 작성	
	사용사례 모델 정의	액터와 사용사례 식별	사용사례 도표
		패키지 식별	
		사용사례 도표 작성	
		사용사례명세서 작성	사용사례명세서
		보조명세서 작성	보조명세서
	아키텍처 사용사례 뷰 기술		아키텍처 사용사례 뷰
분석/설계	사용사례 분 석	시퀀스 도표 작성	시퀀스 도표
		콜레보레이션 도표 작성	콜레보레이션 도표
		클래스 도표 작성	클래스 도표
		패키지 도표 작성	패키지 도표
	사 용 자 인터페이스 설계	사용자 인터페이스 도표 작성	사용자 인터페이스 도표
		화면 레이아웃 설계	화면 레이아웃
		보고서 설계	보고서
	연동요소 식별		연동 명세서
	아키텍처 논리 뷰 기술		아키텍처 논리 뷰
	사용사례 설 계	시퀀스 도표 정제	시퀀스 도표
		콜레보레이션 도표 정제	콜레보레이션 도표
		클래스 도표 정제	클래스 도표
		패키지 도표 정제	패키지 도표
	클래스 설계		클래스 명세서
	하 부 시 스 템 설 계	하부시스템 식별	하부시스템 명세서
		인터페이스 설계	
		하부시스템내부구조설계	
	데 이 터 모 델 설 계	엔티티 클래스 테이블 매핑	테이블 도표
		데이터 모델 최적화	
		무결성 정의	테이블 명세서
		저장 절차 정의	
	배치 모델 정의		배치 도표
	아키텍처 프로세스 뷰 기술		아키텍처 프로세스 뷰
	아키텍처 배치 뷰 기술		아키텍처 배치 뷰
구현	아키텍처 구현 뷰 기술		아키텍처 구현 뷰
	컴포넌트 모 델 구 조 화	컴포넌트 식별	컴포넌트 도표
		컴포넌트 도표 작성	
		컴포넌트 명세서 작성	컴포넌트 명세서
		인터페이스 설계	인터페이스 메소드 정의서
	컴포넌트 구 현	인터페이스 구현	소스 코드
		클래스 구현	소스 문서
시험	단위시험	단위 시험 / 결함수정	시험계획/결과서
	단위체계 시험	단위체계 시험/결함수정	단위체계 시험 계획/결과서
	통합시험	통합 시험 / 결함수정	시험계획/결과서

〈표 3〉 체계개발에 적용된 4+1 아키텍처 뷰

뷰	표현 대상	표현 수단	산 출 물
사용사례	행위	핵심 시나리오 사용사례	사용사례 도 표
논 리	체계 기능성	패키지(논리관점 레이어링), 클래스, 하부시스템(설계 패턴)	패키지 도표 클래스 도표
프로세스	동시성, 병행성, 시작 및 종료, 장애 허용성, 객체 분산, 교착, 반응시간, 처리량, 기능분리, 결함	타스크, 쓰레드, 프로세스, 상호작용	콜레보레이션 도표
구 현	소스코드, 자료파일, 컴포넌트, 실행모듈, 기타 산출물	패키징(구현 관점 레이어링), 형상관리(소유권, 출하전략등)	컴포넌트 도표
배 치	실행모듈, 실행시간 컴포넌트, 플랫폼, 계산 노드	체계공학측면 배치, 설치, 성능 모습	배치 도표

기술서를 작성한다. 이러한 요구사항을 기반으로 체계 전체의 운용을 가정하여 시나리오를 작성한다. 그러한 시나리오 상에서 액터와 사용사례를 식별하고 사용사례 도표를 작성한다.

이러한 사용사례는 주로 체계의 기능적인 요구사항을 중심으로 업무가 일어나는 순서에 의해서 절차를 포함하여 사용사례명세서를 작성한다. 반면에 체계 전체와 관련이 있는 성능, 안정성, 보안성 등의 요구사항은 보조명세서에 기술한다. 아키텍처 사용사례 뷰는 체계의 활동에 대한 도식으로서 전체적인 관점의 사용사례 도표로 표현된다. 표 2의 요구사항 단계에는 작업흐름, 활동, 작업 및 산출물에 대한 내용을 나타내고 있다.

3.4.3 분석 및 설계 (Analysis and Design)

분석 및 설계 작업흐름의 목적은 요구사항을 명세서로 변환하는 것이다. 이를 위하여 요구사항을 이해하고 최상의 구현 전략을 선정하여 체계의 설계로 진행한다. 사업의 초기 단계에는 이해, 구축 및 진화를 위한 견고한 아키텍처가 필요하며, 이에 따라 구현 환경에 맞도록 설계를 조정한다. 이 외에도 성능, 견고성, 확장성, 시험성 등 비기능적 요구사항들에 대한 설계도 필요하다.

K-CIS 관련 분석 및 설계 작업흐름의 활동, 작업 및 산출물은 표 2에 나타나 있다. 주요 활동으로는 사용사례 분석, 사용자 인터페이스 설계, 아키텍처 논리 뷰 작성, 사용사례 설계, 클래스, 서브시스템, 데이터 모델 설계가 있다. 요구사항 작업흐름에서 식별된 사용사례는 분석을 통하여 클래스와 관련 시퀀스 및 콜레보레이션 도표로 작성한다. 이중에서 사용자 인터페이스에 해당하는 클래스에 대하여 화면 및 출력 양식을 설계한다. 아키텍처 논리 뷰는 기반, 미들웨어, 업무 서비스, 응용 등의 계층적 레이어링을 고려한 논리적 패키지 도표로 표현된다.

사용사례 설계는 분석과정을 통하여 작성된 클래스는 책임과 행위를 가지는 것으로 구현환경을 고려하여 정제(Refinement)된 시퀀스 도표, 콜레보레이션 도표, 클래스 도표, 패키지 도표를 작성한다. 클래스 설계는 식별된 클래스에 대한 속성(Attributes)과 오퍼레이션(Operations)에 대한 시그니처(Signature)에 대한 정보를 완성하는 작업이다. 서브시스템 설계는 재사용 컴포넌트에 대한 설계를 하는 부분이다. 데이터 모델 설계는 식별된 엔티티(Entity) 클래스를 적용하는 데이터베이스 관리체계(DBMS)에 맞도록 매핑하는 작업을 수행을 한다.

〈표 4〉 업무 모델링을 위한 C4ISR AF 산출물

구분	번호	산 출 물	K-CIS 적용
공통	AV-1	개요 및 요약 정보	구조요약서술
	AV-2	통합 사전	용어 및 약어집
운용구조	OV-1	고수준운영개념도	○
	OV-2	운용노드연결기술서	○
	OV-3	운용정보교환매트릭스	○
	OV-4	조직관계도표	○
	OV-5	운용활동모형	○
	OV-6a	운용규칙모형	운용활동분할도
	OV-6b	운용상태전이기술서	액티비티흐름도
	OV-6c	운용사건추적기술서	체계운영시나리오
	OV-7	논리데이터모형	○
체계구조	SV-1	체계인터페이스기술서	체계연동기술서
	SV-2	체계통신기술서	○
	SV-3	체계-체계간 매트릭스	○
	SV-4	체계 기능성 기술서	○
	SV-5	운용활동-체계기능 추적 매트릭스	○
	SV-6	체계자료교환매트릭스	○
	SV-7	체계성능인자매트릭스	○
	SV-8	체계진화기술서	×
	SV-9	체계기술예측서	×
	SV-10a	체계규칙모형	HLA[3] 적용
	SV-10b	체계상태전이기술서	상태모드설계
	SV-10c	체계사건추적기술서	○
	SV-11	물리적 스키마	○
기술구조	TV-1	기술표준 프로화일	○
	TV-2	기술표준 예측서	○
범례 : ○ 작성, × 미작성, 기타는 대체 작성			

아키텍처 프로세스 뷰는 시스템의 실행시간에 타스크, 쓰레드, 프로세스 및 그들의 상호작용들에 대한 동시성의 문제점들을 다룬다. 아키텍처 배치 뷰는 여러 프로세싱 노드에 걸친 시스템의 물리적인 배치를 나타낸다. 이러한 아키텍처 프로세스 및 배치 뷰는 하나만 존재하며 개발 중에 지속적으로 정제된다. 표 3에는 아키텍처 프로세스 및 배치 뷰에 대한 내용을 보여주고 있다.

3.4.4 구현 (Implementation)

구현 작업흐름은 레이어별로 조직화된 구현 하부시스템을 코드의 조직으로 정의하고, 소스 코드, 이진 파일, 실행파일 등의 컴포넌트로 클래스나 객체를 구현하며, 개발된 컴포넌트를 단위로 시험하며, 개별 및 팀 단위의 개발 결과물을 실행

3) HLA(High Level Architecture)는 K-CIS 개발 이후에 다른 워게임 프로그램의 추가적인 연동을 고려하여 적용되었다.

체계로 통합하기 위한 목적을 가진다. 표 2에서 보는 바와 같이 적용된 구현 작업흐름에서는 아키텍처 구현 뷰를 작성하고 컴포넌트 식별, 컴포넌트 도표 작성, 컴포넌트 명세서 작성, 인터페이스 설계 및 구현 그리고 클래스 구현 등의 활동이 이루어진다. 그 결과 컴포넌트 도표, 소스 파일 등의 산출물이 작성된다.

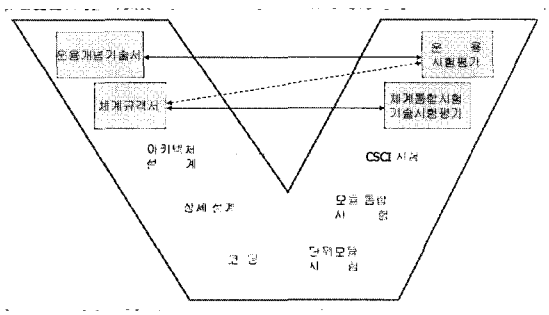
더 나아가 K-CIS 개발 중에 구현을 위하여 빌드, 통합 및 프로토타입 개념이 적용되었다. 빌드는 최종 제품의 부분 또는 운용단계의 버전으로 2개의 빌드가 정의되었다. 그림 3에 보는 바와 같이 구축 단계의 반복 3과 반복 5가 완료되는 시점으로 정하였다. 빌드 1에서는 정보가 전달되는 핵심 라인에 관련된 장비와 소프트웨어가 통합된 상태이고, 빌드 2에서는 전체 장비와 소프트웨어가 통합된 상태로 정하였다. 또한, 통합은 소프트웨어 통합과 체계통합으로 분류된다. 소프트웨어 통합은 그림 3의 반복 5단계에서 CSU, CSC 및 CSCI 단계를 거쳐 통합된다. 체계통합은 교전훈련장비, 통신망, 하드웨어, 훈련통제 소프트웨어 그리고 시설을 포함은 전체 체계가 반복 5단계 이후에서 통합이 된다.

프로토타입은 체계의 핵심 요구사항에 대한 구현 가능성, 주요 구성품간의 연동 가능성 그리고 전체 체계 구성의 적절성을 확인하기 위하여 적용되었다. 이를 위하여 업무 모델링 단계에서는 그림 1에 소개된 교전훈련장비, 통신망, 훈련통제 소프트웨어를 묘사하는 파일럿 과제(Pilot Project)로서 소규모 실험적 프로토타입이 적용되었으며, 본 사업(Real Project)에 들어서면서 소프트웨어 반복 1에서는 소규모의 범위에 대해서 점증형 프로토타입을 적용하며 초급 개발인력에 대해서 RUP 개발 프로세스를 숙달하는 단계로 활용되었다. 이후 범위를 확장하며 전체 체계로 진화하는 전략을 적용하였다.

3.4.5 시험 (Test)

시험 작업흐름의 목적은 제품의 품질을 평가하는 것이다. 최종 제품뿐만 아니라 초기에 아키텍처 등 산출물의 품질도 평가하며 컴포넌트의 상호작용, 요구사항의 바른 구현 정도, 발견된 결함이 배치 이전에 제거되는지 여부를 식별하고 확인하는 과정이다. 이러한 시험은 단위 시험, 통합 시험, 체계 시험, 수락 시험의 단계를 통하여 이루어진다.

K-CIS 소프트웨어는 복합체계 시험의 일부로 이루어졌다. 시험의 과정은 체계개발에 적용되는 "Vee" 모델[1]에 의하여 이루어졌다. 그림 5는 K-CIS 개발을 위한 Vee 모델을 보여주고 있다. 여기서 운용개념기술서와 체계규격서는 전체체계와 관련된 부분이며, 아키텍처 및 상세 설계, 코딩은 소프트웨어와 관련되어 있다. 이와 대응되는 단위 모듈시험, 모듈통합시험 및 CSCI 시험은 소프트웨어 대상 시험이며, 체계통합시험, 기술 시험평가 및 운용 시험평가는 체계를 대상으로 하는 시험이다.



〈그림 5〉 K-CIS 개발을 위한 Vee 모델

3.4.6 배치 (Deployment)

배치 작업흐름의 목적은 완료된 소프트웨어 제품을 사용자에게 전환하는 것이다. K-CIS 소프트웨어는 반복 5 단계의 소프트웨어 체계까지 통합되어 실 운용 장소에 설치되고 시험되었다. 그 이후에 최종 운용자 및 사용자들에 대한 교육과 실제 운용을 위한 초기 자료가 구축되었다. 이 단계에서는 분석 및 설계 단계에서 작성된 아키텍처 배치 뷰를 반복 개발 중에 정제를 하고 모듈 및

체계 통합 과정에서 설계된 내역대로 배치가 된다.

배치과정에서는 체계통합, 체계설치, 소프트웨어 설치, 시험, 설치 지원 등의 활동을 위한 팀들의 참여로 이루어진다. 소프트웨어가 설치된 이후에 단위시험, 연동시험 및 수락시험을 거쳐 개발된 체계가 사용자 측에 인도된다.

3.5 지원 작업흐름 (Supporting Workflows)

3.5.1 형상관리 (Configuration and Change Management Workflow)

형상관리 작업흐름의 목적은 사업의 산출물이 진화하는 과정을 추적하여 무결성을 확보하는 것이다. 실제 소프트웨어 개발 과정에는 많은 산출물이 만들어진다. 그렇지만 이들이 변경되고 진화하는 과정을 잘 유지하는 것은 운영 및 유지보수 단계에서 결함을 제거하거나 차후 성능개선 또는 재사용을 위하여 반드시 필요한 활동이다.

K-CIS 형상관리를 위한 조직으로 형상통제위원회(CCB: Change Control Board)를 두어 요구사항 변경, 베이스라인 변경 등의 의사결정을 하였다. CCB에는 개발자와 발주자가 동시에 참여하여 기능, 일정, 비용 등의 요소를 검토한 후에 합의에 의해 의사결정을 하였다. 또한 형상관리 절차는 형상 식별, 형상 통제, 형상자료 유지, 형상 감사 순서로 이루어진다. 이를 위하여 기술변경요청서(ECR: Engineering Change Request), 기술변경제안서(ECP: Engineering Change Proposal), 규격 완화/면제 요청서(RFD/W: Request for Deletion / Withdraw), 규격변경통지서(SCN: Specification Change Notice), 형상 상태 및 감사 보고서 등의 양식과 보고서가 작성된다.

형상변경과 관련하여 베이스라인을 설정하는데, 그림 3에서 전체 체계의 분석 및 설계 이후에 체계설계검토회(System Design Review)가 종료되는 시점에 기능기준선(Functional Baseline), 소프트웨어 상세 설계가 완료되는 빌드 1 (반복 3)에서 단위체계시험검토회(Test Readiness Review)가 종료되는 시점에 할당기준선(Assignment Baseline), 기술시험결과검토회(Development Test Result Review) 후에 제품기준선(Product Baseline) 그리고 운용시험결과검토회(Operational Test Result Review) 이후에 인도기준선(Delivery Baseline)이 설정되었다. 그 외에도 요구사항, 소스코드, 시험자료, 문서 등의 변경을 효과적으로 관리하기 위하여 RequesitePro, ClearQuest 그리고 ClearCase 등 자동화 관리 도구를 이용하여 변경내용을 관리하였다.

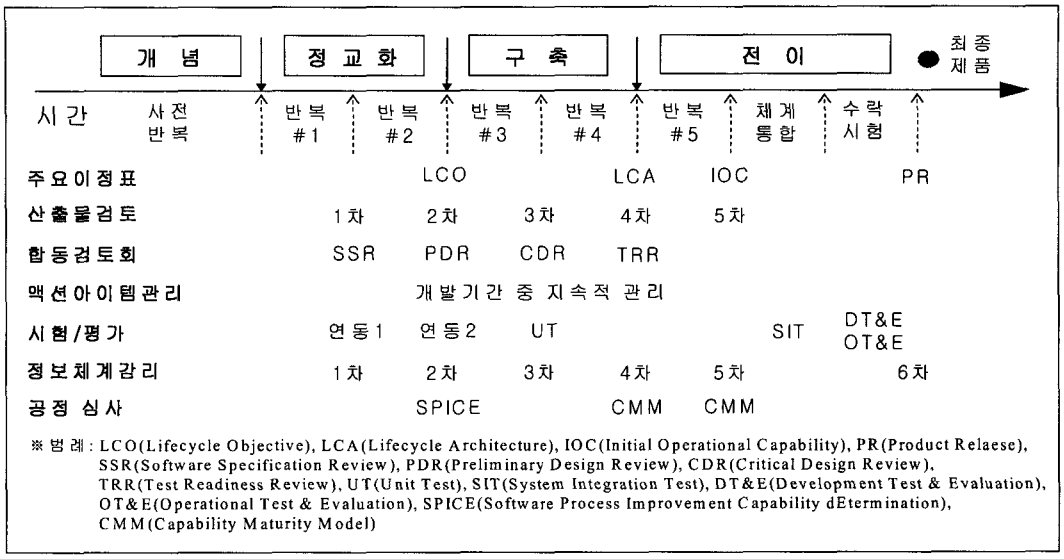
3.5.2 사업관리 (Project Management)

소프트웨어 사업관리는 고객과 사용자의 요구를 충족하는 제품을 공급하면서 목표 달성, 위험관리, 제한사항의 극복 등의 균형을 이루는 활동들의 집합이다. 본 사업관리 작업흐름은 소프트웨어 중심의 사업을 관리하기 위한 프레임워크 제공, 사업 계획, 인력 충원, 실행 및 사업 모니터링을 위한 실질적인 지침서의 제공 그리고 위험관리를 위한 프레임워크의 제공을 목적으로 하지만, RUP는 인력 고용, 훈련, 지도, 예산 수립 및 할당, 공급자와 고객간의 계약 관리 등에 대해서는 다루지 않는다.

3.5.2.1 위험관리 (Risk Management)

소프트웨어 개발에 있어서 위험은 대규모의 통제가 요구되는 직접 위험, 통제가 미미하거나 없는 간접 위험으로 나뉜다. 여기에 발생 가능성과 사업의 영향성은 위험의 정량적 단일 지표로 조합이 되며, 아주 높음, 높음, 보통, 낮음, 아주 낮음 의 5단계 값으로 표현이 가능하다.

K-CIS 개발을 위한 위험관리 조직은 분야별 개발팀장이 주관이 되어 분야별 위험요소를 식별 및 관리를 하고, 사업관리팀에는 위험관리자를 두어 전체적인 현황을 유지하고 개발자 위험통제위원회를 두어 위험을 관리하였다. 이러한 위험현황은 발주측 위험관리자에게 매월 정기적으로 보고되고 협조가 필요한 사항은 발주측 주관 합동위험통제위원회에서 위험관리에 대한 제반 활동을



〈그림 6〉 K-CIS 품질관리를 위한 활동

통제 및 감독하도록 하였다.

위험관리 절차는 관리착수, 식별, 분석, 조치계획, 추적통제 순으로 이루어져 있다. 먼저, 위험관리를 위한 계획 및 조직을 정의하며 담당자를 지정한다. 그리고 위험에 대한 식별, 중요도 분석, 이에 따른 조치 담당자, 계획 및 일정을 지정, 처리 내용에 대한 추적통제가 이루어지도록 하였다. 이를 위하여 위험징후목록, 위험관리계획서, 위험발생보고서, 위험등록대장, 위험완화계획서, 위험상태보고서, 위험관리위원회회의록 등의 문서를 유지관리 하였다.

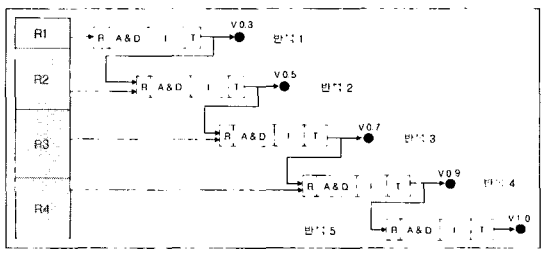
3.5.2.2 품질관리 (Quality Management)

품질관리의 활동은 프로세스 또는 제품이 명시된 요구사항에 적합하며 설정된 계획에 충실함을 보증하는 품질보증 활동이라 할 수 있다. 이러한 활동을 하는 조직 또는 사람은 제품 및 프로세스를 수행하는 사람으로부터 조직적인 자유와 권한이 요구되며 검증, 확인, 합동검토, 감사 및 문제해결 등의 프로세스와 연계하여 수행이 된다.

K-CIS 소프트웨어의 품질보증을 위하여 개발자 측에는 품질보증 팀을 두고 RUP의 주요 이정표는 반복 개발 주기와 연계 하였으며, 산출물 검토는 매 반복 시 마다 시행되었다. 합동검토회는 4회, 액션아이템 관리는 개발기간중 지속적으로, 시험 및 평가는 연동시험, 단위체계시험, 체계통합시험, 기술 및 운용 시험평가로 관리되었다. 감리는 매 반복 완료 후, 운용시험평가 후 및 운용단계에서 사후감리로 추진되었다. 프로세스심사는 SPICE[8]와 CMM[10]을 적용하여 실시되었다. 그림 6은 K-CIS를 위한 품질관리 활동들을 보여주고 있다.

3.5.2.3 프로세스관리 (Process Management)

K-CIS 소프트웨어를 개발하는 데 있어서 요구사항과 반복 개발과의 관계는 그림 7과 같다.



〈그림 7〉 K-CIS 요구사항과 반복 개발 연관성

전체 요구사항은 4 부분으로 나누어 처음 반복에서는 쉽고 작은 범위(R1)를 선정하여 수행한다. 두 번째 반복에서는 체계의 핵심부분(R2)을 개발한다. 그 이유는 체계의 위험요소를 먼저 평가하고 난관을 식별한 경우에 대응이 용이하기 때문이다. 그리고 반복 3, 4는 통상적인 부분(R3, R4)을 개발한다. 반복 5에서는 앞선 반복에서 개발된 컴포넌트들을 통합하고 시험하는 절차이다. 이와 같이 요구사항을 점증적으로 확대하면서 프로세스를 수행한다.

점증적 프로세스에서는 요구사항의 범위를 확대하지만 앞 단계에서 전체적인 요구사항의 확정, 분석, 설계가 이루어져야 한다. 예를 들면, 반복 1에서는 전체 요구사항이 확정되고, 반복 2에서는 전체 요구사항이 분석이 되며 반복 3에서는 전체 요구사항에 대한 설계가 이루어져야만 반복의 효과를 달성할 수가 있다. 그렇지 않으면 반복의 효과보다는 전체적인 모습을 그리지 못하여 통합과정에서 추가적인 수정이나 보완사항이 발생하여 전형적인 폭포수형 프로세스보다 효율이 저하될 가능성이 있다.

4. K-CIS 소프트웨어 개발 평가

본 절에서는 K-CIS 소프트웨어를 개발하기 위하여 기존의 체계개발 프로세스에 RUP를 접목한 독특한 프로세스에 대하여 기존의 폭포수형 및 전형적인 RUP 개발 프로세스와 비교를 통한 평가를 한다. 객관적인 비교를 하기 위해서는 동일 사업에 대하여 적용된 결과를 비교 및 평가하여야 하나 현실적으로 제약사항이 많아 기존에 발표된 자료와 K-CIS 소프트웨어 개발 사례에서 수집된 자료를 근거로 평가를 한다.

4.1 프로세스 작업흐름

체계의 일부인 소프트웨어를 개발함에 있어서 현행 국방획득관리절차[17]에는 폭포수형 모델을 암시적으로 적용하고 있으며, 반복 개발에 대해서는 명시적으로 제한을 두지는 않았다. 그렇지만 국내에서 반복 개발을 적용한 성과사례[4]를 찾아보기는 쉽지 않다. 그리고 전형적인 RUP도 소프트웨어 개발만을 가정하고 있으므로 체계개발 프로세스와 조화를 위해서는 별도의 프로세스 조정이 요구된다.

한편, 폭포수형 프로세스는 일괄 개발을 고려하고 있으나 RUP는 반복 개발을 기반으로한 점증형(Incremental model)[5] 또는 진화형(Evolutionary model) 적용이 가능하여 대규모 체계의 개발에 적합하다. 이러한 RUP 특성은 폭포수형 프로세스에 비하여 사업 이해당사자간의 의사소통을 용이하게 해준다. 각 모형별 세부 활동, 작업, 산출물을 포함한 절차에 대해서 폭포수형은 세부 절차[1]가 준비되어 있으며, RUP에 대해서는 구체적인 활동, 작업, 산출물을 별도로 정의하여 적용이 필요하다. K-CIS 소프트웨어 개발을 위한 세부 절차는 표 2에 제시되었다.

업무 모델링 측면에서 폭포수형 프로세스는 고려가 되어 있지 않으며 요구사항 단계부터 시작을 한다. 그리고 RUP에서는 조직 구성도, 도메인 모델링, 다수 체계를 가진 하나의 업무, 일반 업무 모델, 새로운 업무, 재공학(Reengineering) 등 5가지 시나리오를 제시하고 있다. 아울러 업무 비전 문서, 업무 사용사례 모델, 업무 객체 모델, 목표 기관 평가, 업무 규칙, 보조업무명세서, 업무 용어집을 작성하도록 제시되어 있다. 그렇지만, K-CIS 적용 RUP는 표 4에 제시된 C4ISR AF 산출물을 적용하였으며, 운용구조, 체계구조 및 기술구조로 구분된 세부 산출물을 작성하여 전형적인 RUP 보다 구체성을 가지고 있다.

요구사항 측면에서 폭포수형 프로세스의 경우에는 한 번 확정이 되면 후속 단계에서 수정이

4) RUP를 적용한 사례인 경우도 모델링에 중점을 두어 소개하고 있으며, 반복개발 성과는 언급되지 않고 있다.

5) 폭포수형 프로세스는 BUILD를 하나만 가지고, 점증형은 BUILD(N) = BUILD(N-1) + 추가기능, 진화형은 BUILD(N) = BUILD(N-1) + 정제된 요구사항의 관계를 가진다.[1]

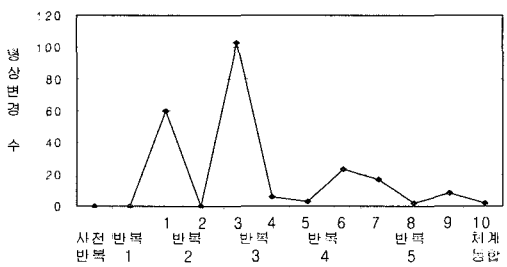
용이하지 않다. 일부 개선된 폭포수형 프로세스에는 피드백을 고려한 프로세스를 허용하지 있지만 대폭적인 수정은 불가하다. 그러하기 때문에 요구사항이 명확하고 한꺼번에 체계를 개발할 경우에는 유리하다. 반면에 RUP 방법은 반복 개발을 기반으로 하기 때문에 이전에 프로세스에서 습득된 경험과 시행착오를 반영하여 후속 프로세스에서 보완하는 것이 한결 용이하다. 따라서 요구사항이 불명확 경우라도 반복 과정을 통해서 적절하게 개발을 할 수가 있다. 요구사항 프로세스와 유사하게 분석 및 설계, 구현, 그리고 시험의 용이성 측면에도 적용이 가능하다.

구성요소의 통합 측면에서 폭포수형은 일괄 개발을 가정할 경우 모든 구성품을 동시에 분석, 설계, 코딩, 시험을 하기 때문에 단계별로 시험, 구성품별 결함을 제거 그리고 회귀시험을 하게 된다. 이 때 구성품별로 결함이 상호 어떠한 영향을 줄지 판단이 되지 않으므로 시험 대상 범위가 확대된다. 프로토타입의 경우는 위험이 예상되는 구성품에 대하여 사전에 작성하여 검증할 필요가 있다. 반면에 반복 개발인 경우에는 단계별로 잘 시험된 구성품에 새로운 구성품을 추가하면서 시험이 이루어지기 때문에 통합이 용이하며, 프로토타입은 목표체계로 진화하면서 개발이 가능하다. 한편, 배치 측면에서는 특별한 차이를 찾을 수 없다.

4.2 지원 작업흐름

형상관리는 요구사항의 변경에 대한 관리를 목적으로 하기 때문에 잦은 변경이 있는 경우가 불리하게 된다. 반복 개발은 일괄 개발보다 결함 또는 개선 사항을 식별하기가 용이하여 결과적으로 많은 변경이 발생하게 된다. 요구사항 단계의 변경은 분석, 설계, 구현, 시험, 통합 단계에 영향을 주게 되며, 관련 문서화 및 코딩 등에 추가적인 노력이 요구된다.

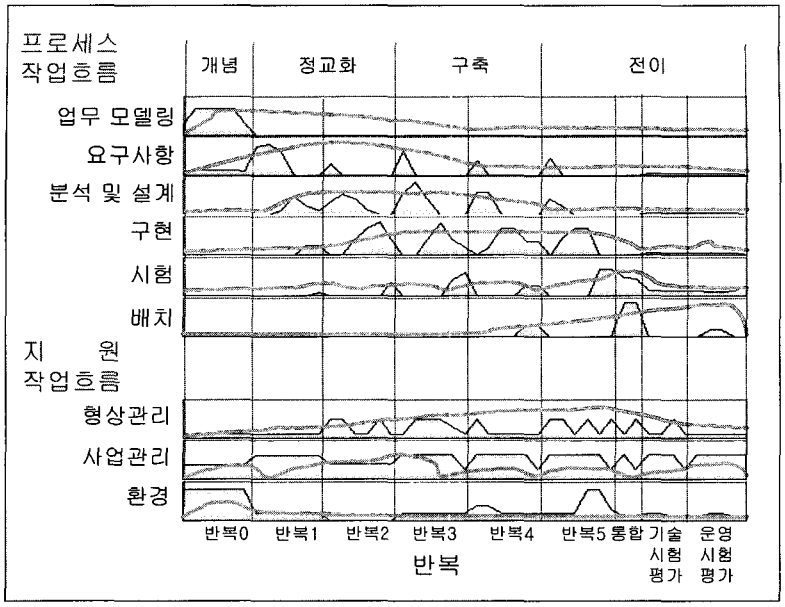
그림 8은 K-CIS 사례에서 반복 개발단계별 형상변경 수를 보이고 있다. X축의 1~10까지 숫자는 형상통제위원회가 열린 순서에 대한 번호이다. 사전 반복단계는 개념연구를 수행하여 조직의 운영구조, 체계구조 및 기술구조를 작성하고 요구사항을 정의하는 단계이며, 반복 1은 시범 개발의 의미를 가지고 있어 형상변경이 제기된 사항이 없다. 그러나 소프트웨어 요구사항의 재정의, 분석, 설계가 집중적으로 일어나는 반복 2, 3에서 형상변경이 집중됨을 알 수 있다. 그리고 구현 및 통합과정인 반복 4, 5 단계에서도 일부 형상변경이 일어나고 있음을 알 수 있다.



〈그림 8〉 반복 개발단계별 형상변경 수

위험관리 측면에서 폭포수형 프로세스는 위험으로 식별된 부분에 대한 시험이 완료되어야 완전하게 위험의 정도를 확인 할 수 있다. 그래서 프로세스의 후반부에 식별된 위험요소를 완화하기 위하여 추가적인 노력이 요구된다. 반면에 반복 개발인 경우에는 핵심 위험요소를 선행 반복 프로세스에 할당하여 위험도를 완화할 수 있다.

K-CIS에서는 사전 반복단계에서 사업의 분야별 위험요소를 식별하고 위험관리절차를 수립하였다. 그리고 반복 1단계에서는 시범 개발로 기존의 폭포수형 개발에 익숙하거나 초급 인력이 새로운 방법론에 대하여 친숙화 하는 단계를 통해 위험도을 완화하였다. 또한 개발의 위험도가 높은 핵심 부분을 반복 2단계에 할당하여 개발 실패에 따른 여유를 갖도록 하였다. 또한 소프트웨어 내부적으로는 그림 2의 반복 5단계에서는 CSU, CSC, CSCI로 소프트웨어의 통합을 수행하고, 교전훈련장비, 통신망, 훈련통제 소프트웨어를 대상



〈그림 9〉 K-CIS 개발 단계별 투입 노력

으로 단계별 연동시험을 통해 체계통합의 위험도를 완화하였다. 그리고 이러한 단계별 연동시험 이후에는 그림 6에 제시된 각종 품질관리 활동을 계획하여 단계별 위험도를 재평가하였다.

품질관리 측면에서는 투입 일정, 노력, 공정 및 결함 등의 요소에 대하여 살펴본다. 먼저 프로세스별 투입 일정 및 노력에 대한 전형적인 자료와 K-CIS 자료를 표 5에 제시하였다. 여기에서 전형적인 폭포수형 프로세스와 RUP를 비교하면, 일정면에서는 II 단계에서 10% 정도 증가되었으며, 노력면에서는 IV 단계에서 5% 감소되었으며 III 단계에서 5% 증가되었다. K-CIS RUP 사례에서는 일정 및 노력면에서 II, III, IV 단계에 걸쳐서 증가되었음을 알 수 있다. 특히, IV(전이) 단계의 증가폭은 25~27%로 체계통합, 체계통합시험 및 기술 및 운용 시험평가 등 체계관련 공정활동으로 인한 결과이다.

〈표 5〉 프로세스별 투입 일정 및 노력 비율(%)

구분[6]	전형적 폭포수형[13]		전형적 RUP[13]		K-CIS RUP	
	일정	노력	일정	노력	일정	노력
I 단계	13	5	10	5	13	4
II단계	20	20	30	20	25	24
III단계	55	60	50	65	25	33
IV단계	12	15	10	10	37	35

6) 단계는 폭포수형 프로세스에서는 타당성(Feasibility), 설계(Design), 실행(Execution), 구현(Implementation), RUP에서는 개념(Inception), 정교화(Elaboration), 구축(Construction), 전이(Transition)를 의미한다.

다음으로 소프트웨어 규모에 따른 투입 노력을 살펴본다. 표 6은 K-CIS 소프트웨어의 규모를 기준으로 한 투입 노력의 비교결과를 보여준다.

전형적인 폭포수형 프로세스와 RUP를 비교하면 RUP의 노력이 더 소요되나 기간은 줄어드는 것을 알 수 있다. 한편, K-CIS 사례의 기간은 47개월이 소요되었으나 실제 소프트웨어 개발에 관련된 기간은 30개월로 전형적인 RUP와 유사하며, 17개월 정도의 일정 연장은 체계개발 일정과 동기화된 결과이다. 그리고 노력면에서 전형적인 RUP와 유사하나 발주자 측의 투입 노력 380 MM를 제외하면 전형적인 폭포수형 프로세스와 유사하다.

〈표 6〉 규모에 따른 투입 노력 (MM) 비교

라인 수 (KSLOC)	전형적 폭포수형[7]		전형적 RUP[8]		K-CIS RUP	
	기간	노력	기간	노력	기간[9]	노력[10]
450	40	1466	28	1,890	47	1,752

업무 모델링은 개념 단계에서만 일어나 있고 정교화 단계에서는 적용되지 않고 있음을 알 수 있다. 요구사항은 그림 8에서 보는 바와 같이 정교화 단계에서 집중하여 거의 확정하여야 되나, 실제 투입 노력은 지속적이지 않으며 매 반복 마다 간헐적으로 이루어져 있음을 알 수 있다. 이와 같은 현상은 분석 및 설계 단계에서도 비슷하게 일어나고 있다.

이러한 결과는 K-CIS RUP는 계획상으로는 RUP 프로세스를 적용하였으나, 실행단계에서는 개발 범위를 분할하여 폭포수형 프로세스를 반복한 것과 유사한 결과가 되었다. 이는 기존에 폭포수형 프로세스에 익숙한 인력이 RUP로 전환을 하면서 새로운 프로세스에 적응에 시간이 필요함을 말해준다. 이러한 근거는 표 6에서 K-CIS RUP의 전체 투입노력 1,752MM 중 순수 개발자의 투입노력이 1,372 MM로 전형적인 폭포수형 프로세스의 1,466 MM와 거의 유사함에서 찾을 수 있다.

이외에도 투입 노력과 관련하여 형상관리는 일반적으로 구축단계의 마무리 지점에 집중되나 K-CIS 사례에서는 정교화 및 구축단계에서 집중적으로 일어나고 있다. 사업관리는 일반적으로 개념 및 정교화 단계에서 집중이 되나 전 단계에 골고루 퍼져 있다. 환경 구축은 개념단계에서 프로토타입의 조성과 구축단계에서 하드웨어 설치로 일반적인 유형과 유사하게 나타난다.

한편, 전형적인 RUP는 폭포수형 프로세스 보다 더 많은 노력의 투입이 요구된다. 이러한 비용 요소에 대한 보상은 품질로 이루어질 수 있다. 이러한 사실을 확인하기 위하여 표 7에서는 프로세스별 1000 라인당 식별된 결함 수를 보여주고 있다. 일반적으로 결함의 식별은 산출물 검토, 합동 검토회, 검증, 확인, 시험, 감리, 그리고 공정심사 등을 통하여 이루어진다. 사업마다 활동, 작업, 작업자의 수준이 달라 직접적인 비교는 어렵지만 개략적인 추세를 분석하기 위한 목적으로 표 7이 제시되었다.

표 7에 나타난 결과로 전형적인 폭포수형인 경우에는 앞 단계에서 결함을 찾기가 용이하며 구현 단계에서는 결함을 발견하기 어려움을 알 수 있다. 그리고 나선형 또는 RUP와 같은 반복개발인 경우에는 반복의 수가 4개 이하일 때는 오히려 줄어들며 5개 이상이 되면서 증가함을 알 수 있다.

K-CIS 사례는 요구사항 활동에서 결함 식별이 저조하면서 구현 활동에서 많은 결함을 식별하고 있는 특이한 상황을 보여주고 있다. 이러한 사실은 별도의 엄격한 업무 모델링을 통하여 정제된 요구사항을 작성한 상태에서 소프트웨어의 개발에 착수한 결과로 보인다. 분석 및 설계 활동에서 결함 식별이 낮은 이유는 요구사항이 안정적이거

(표 7) 프로세스별 결함 수 비교

단 계	프로세스[11]			
	전 형 적 폭포수형	반복개발 (2~4)	반복개발 (5~10)	K-CIS RUP
요구사항	4.2	3.2	2.4	0.5
분 석	3.1	2.5	3.7	1.3
설 계	1.1	1.1	2.2	1.2
구 현	1.0	2.1	3.5	5.9
계	9.4	8.9	11.8	8.9

7) COCOMO 모델[3]의 Semidetached 유형을 적용하여 Effort (MM) = 2.4 * 4501.05, Duration(M) = 2.5 * 14660.38로 계산된 결과이다.

8) RUP[12] 사례(Table 7-2)에서 보간법으로 반복기간 및 인력을 계산하고 전형적인 6개 반복 [1, 2, 2, 1]을 적용한 결과이다.

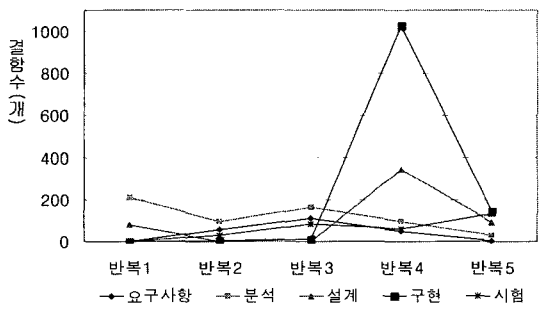
9) 개념단계 (6개월), 소프트웨어 6개월 단위 5개 반복 개발 (30개월), 체계통합시험 (2개월), 기술 및 운용 시험평가 (9개월)로 총 47개월이 소요되었다.

10) 순수 개발자에 의한 투입 노력은 1,372 MM 이며, 발주자 측의 산출물 검토 및 사업관리에 대한 노력 380 MM를 합한 결과이다.

나 RUP의 UML 모델링을 통하여 결함을 찾는 데 익숙하지 않은 경우로 보이며, 구현 활동에서 반복 개발을 통하여 2.6개, 수명주기별 공정심사를 통하여 0.6개, 액션아이템(Action Item)에 의하여 2.7개를 합한 결과이다. 전반적으로 반복이 4이하인 경우와 유사한 정도의 결함을 식별하고 있다.

다음으로 그림 10은 반복 개발단계별로 식별된 결함 수를 보이고 있다. 여기서 반복 1, 2, 3 단계에서는 요구사항 및 분석 활동을 통하여 얼마간의 결함을 찾을 수 있음을 알 수 있다. 그러나 반복 4에서는 결함이 기하급수적으로 증가하고 있다. 이러한 원인은 점증적 개발로 인하여 각 구성품 단위에서 식별되지 않았던 것이 통합을 고려하면서 설계 및 구현 활동에 반영된 것으로 보인다. 그러나 구성품 단위로 통합이 이루어지는 반복 5 단계에는 상대적으로 결함이 감소하고 있다.

또한 반복 개발중의 시험 활동을 통하여 식별된 결함은 분석 활동을 통해서 얻어진 결과 보다 저조하다. 이러한 원인은 RUP에 익숙하지 못한 경우에 짧은 반복 개발 기간동안 요구사항, 분석 및 설계, 구현, 시험을 수행하다보면 마지막에 있는 시험 활동이 제한을 받게 되는 결과로 보인다.



〈그림 10〉 반복 개발단계별 식별된 결함 수

4.3 종합 평가

본 논문에서는 기존의 폭포수형을 위주로 하는 개발 프로세스에서 반복을 기반으로 하는 RUP를 적용하여 소프트웨어를 포함한 체계개발과 연계하여 적용하고자 하였다. K-CIS RUP를 전형적인 폭포수형 프로세스 및 RUP와 비교한 결과는 표 8에 나타나 있다. 주요 성과로서는 전형적인 RUP와 비교할 경우 체계개발 프로세스와 조화성, 활동, 작업 및 산출물에 대한 구체적인 정의, 업무 모델링에 대한 구체성 향상 등을 들 수 있다.

더 나아가 기존의 폭포수형 공정에서는 대규모 체계에 적용 용이성, 의사소통의 용이성, 요구사항 변경의 용이성, 불명확한 요구사항에 대한 적응성, 분석, 설계, 구현, 시험 및 통합의 용이성, 위험관리, 품질관리 및 투입자원의 효율성, 기술의 급속한 변화 환경에서 개선이 이루어짐을 확인하였다. 그러나 형상관리 및 부수적인 문서화의 노력 증가와 개발인력의 잦은 변동측면에서는 효율이 저하됨을 확인하였다.

〈표 8〉 프로세스별 비교 결과

비 교 항 목	전형적 폭포수형	전형적 RUP	K-CIS RUP
체계개발 프로세스와 조화성	○	△	○
대규모 체계에 적용 용이성	△	○	○
의사소통의 용이성	△	○	○
활동,작업,산출물 구체성	○	△	○
업무 모델링의 구체성	×	△	○
요구사항 변경의 용이성	△	○	○
불명확한 요구사항 적응성	×	○	○
분석 및 설계의 용이성	△	○	○
구현의 용이성	△	○	○
시험의 용이성	△	○	○
통합의 용이성	△	○	○
배치의 용이성	○	○	○
형상관리의 용이성	○	△	△
위험관리의 용이성	×	○	○
품질관리의 용이성	△	○	○
투입자원의 효율성	△	○	○
기술의 급속한 변화	△	○	○
개발 인력의 잦은 변동	○	△	△
범례: ○ 좋음, △ 보통, x 낮음			

한편, K-CIS 소프트웨어를 RUP 기반으로 개발하면서 다음과 같은 교훈을 얻을 수 있었다. 첫째, 체계개발 프로세스와의 정합을 위하여 RUP 프로세스의 일부 작업흐름, 활동, 작업이 그림 3과 같은 조정이 요구된다. 둘째, 사업 규모나 특성에 맞도록 활동, 작업, 산출물의 정의가 요구된다. 셋째, 업무 모델링이 정교할수록 결함 발생률이 감소한다. 전형적인 RUP에서 사용사례 중심의 업무 모델링 보다 정교한 C4ISR AF 적용으로 표 7과 같이 결함 발생률이 낮아졌다. 넷째, 반복 개발은 위험 완화, 자체 경험 재활용이 용이하다. 다섯째, 형상변경은 정교화 단계에서 집중적으로 발생한다. 다섯째, 단계별 연동시험 및 품질활동의 연계로 위험 완화 및 품질의 향상을 가져온다. 여섯째, 체계개발 프로세스에 접목된 RUP의 투입 일정 및 노력 비율은 전이 단계가 25% 정도 증가하고 구축 단계가 상대적으로 25% 정도 감소한다. 일곱째, 폭포수형 프로세스에서 RUP로 최초 전환시 발생되는 현상으로 폭포수형 프로세스의 반복으로 수행한다. 이것은 통합과 분배의 노하우가 부족하기 때문이다. 또한 결함 식별 능력이 전반적으로 저하되고 요구사항, 분석 활동 보다 설계, 구현, 시험 활동에 의해 결함을 식별함으로써 결함제거 비용이 높아져 비효율성을 나타내고 있다.

5. 결론 및 향후 연구

본 논문에서는 체계개발 프로세스와 RUP를 통합한 K-CIS RUP 사례를 고찰하였다. 그 결과 RUP를 체계개발 프로세스와 잘 조화되도록 프로세스 일정 및 공정을 수립하였으며, 관련 활동, 작업 및 산출물에 대한 구체적인 정의를 제시하였다. 표 8에서 제시된 바와 같이 기존의 폭포수형 프로세스 보다 6개의 프로세스 작업흐름, 위험 및 품질관리 등의 지원 작업흐름에서 개선이 있음을 확인하였다. 그렇지만 형상관리와 수반되는 문서화, 잦은 인력의 변동이 예상되는 환경에는 부정적 요인으로 작용함을 발견하였다.

특히, 투입 일정 및 노력에 대한 통계적 자료를 획득하여 K-CIS RUP는 체계개발 프로세스에 동기화 하는 이유로 전이단계가 전형적인 프로세스 보다 25~27% 증가됨을 보였고, 기존의 폭포수형에 익숙한 개발인력이 RUP를 수행함에 있어서 폭포수형 프로세스를 단순 반복하는 시행착오를 보이는 것으로 나타났다. 이러한 시행착오는 숙련도가 높을수록 요구사항, 분석 및 설계, 구현, 시험 등 전체적인 작업흐름의 분배를 고려하고, 반대인 경우는 노력의 분배가 미흡한 것으로 나타났다. 아울러서 RUP의 경우에는 형상변경의 수가 증가하며 관리활동의 노력도 증가하는 것으로 확인되었다.

본 논문의 연구결과는 공공기관에서 대규모 복합체계를 획득함에 있어서 점증적 반복 확장 형태의 개발방법을 택하는 경우에 프로세스 조정을 위한 세부 참고자료로 활용될 수 있을 것으로 믿는다. 또한, 본 연구와 관련하여 향후에는 공공기관에서 적용한 다양한 프로세스에 대한 자료 분석을 통하여 표준적인 프로세스 및 관련 지수가 개발이 되면 보다 효과적으로 활용이 가능할 것으로 판단된다.

참 고 문 헌

[1] AEW Services, "The Role of the Project Life Cycle (Life Span) in Project Management", AEW Services, Vancouber, BC, 2003.

[2] Booch G, Kozaczynski Wojtek, "Component-Based Software Engineering", IEEE Software, pp. 34~36, October, 1998.

[3] Braude E., "Software Engineering An Object-Oriented Perspective", Wiley, 2001.

[4] C4ISR AWG, C4ISR Architecture Framework, Ver. 2.0, Dec. 1997.

[5] DoD, DoD Architecture Framework, Ver. 1.0, Aug. 2003.

[6] DoD, Software Development and Documentation, MIL-STD-498, 1994.

[7] Dsouza D.F., and Wills A.C., "Objects, Components, and Frameworks with UML", Addison-Wesley, 1998.

[8] ISO/IEC 12207.1, Information Technology - Part 1: Software Life Cycle Processes, 1996.

[9] Jacobson I., Booch G, and Rumbaugh J, "The unified software Development Process", Addison-Wesley, 1999.

[10] Mark C. Paulk, Bill Curtis, and Charles V Weber, "Capabilility Maturity Model for Software", TR, CMU/SEI-93-TR-024, Vol. 1.1, Feb. 1993.

[11] Martin J., "Information Engineering, Book I Introduction", Prentice-Hall, Inc., 1989.

[12] Philippe Kruchten, "Modeling Component Systems with the Unified Modeling Language", Available at http:\\www.sei.cmu.edu/cbs/icse98/papers/, 1998.

[13] Philippe Kruchten, "The Rational Unified Process An Instroduction", 2nd ed. Addison Wesley, 2000.

[14] Sterling, The CBD96 Standard, Ver. 2.1, Sterling, Jul. 1998.

[15] Szyperski C., "Component Software : Beyound Object-Oriented Programming", Addison-Wesley, 1998.

[16] Yourdon E., "Structured Analysis and System Specification Workshop", Yourdon Press, 1984.

[17] 국방부, 국방획득관리규정, 국방부 훈령 727호, 2003년 2월.

[18] 손태종, 김영도, 국방정보체계 상호운용성 종합구조 설계방법론 및 적용방안 연구, 2003년 12월.

[19] 육군교육사, 과학화전투훈련장 중앙통제장비 운용개념기술서, 1999년. 6월.

[20] 최남용, 진종현, 송영재, 국방아키텍처프레임워크의 개발, 정보처리학회논문지 D 제11-D권 제2호, 2004년 4월.

◐ 저 자 소 개 ◑

이 길 섭

1985년 : 금오공과대학 전자공학과 졸업(공학사)
1987년 : 한국과학기술원 전산학과 졸업(공학석사)
1992년 : 한국과학기술원 전산학과 졸업(공학박사)
1991년~1995년 : 육군전산소 개발실
1996년~1998년 : 국방부 정보체계국
2003년~현재 : 국방대학교 전산정보학과 순환직 교수
관심분야 : 정형기법, 소프트웨어공학, 분산시스템 모델링
E-mail : gislee@kndu.ac.kr

이 태 공

1976년 : 공군사관학교 전자과 졸업 (이학사)
1986년 : 미국 해군대학원 체계관리학과 졸업 (이학석사)
1990년 : 미국 Wayne State University 전산학과 조교
1991년 : 미국 Wayne State University 전산학과 졸업(공학박사)
1992년~1995년 : 공군 군수사령부 전산실장
1995년~현재 : 국방대학교 전산정보학과 교수
2001년 : 미국 해군대학원 교환교수
2003~현재 : 한국 ITA학회 회장
관심분야 : 정보기술아키텍처, 엔터프라이즈 엔지니어링, 개방체계
E-mail : tglee@kndu.ac.kr