

마르미-III 에서의 요구사항 관리 적용 사례

*조진희, **박경민, **임춘봉, *김진삼
*한국전자통신연구원
**화이트정보통신(주)
e-mail : chojh@etri.re.kr

A Case Study of Requirements Management with MaRMI-III

*Cho Jin Hee, **Park Kyung Min, **Lim Choon Bong, *Kim Jin Sam
*Electronics and Telecommunications Research Institute
**White Information & Telecommunication Co. Ltd.

요 약

본 논문에서는 마르미-III 개발방법론을 기반으로 하여 시스템 개발 프로젝트를 수행함에 있어, 프로젝트 초기 “요구획득(requirement)” 단계에 어떻게 하면 보다 효과적으로 고객의 요구사항을 관리하여 요구관리(requirements management) 목표를 달성하고, “아키텍처(architecture)” 단계의 원활한 진행을 보장할 수 있을지 그 적용 사례를 소개하고자 한다.

1. 서론

시스템 개발 업체(supplier)는 고객이 필요로 하는 기능을 가진 시스템을 적시에 개발하여 공급하는 것이 프로젝트를 수행하는 당면 과제이다. 반대로 시스템을 도입하여 업무에 활용하기를 원하는 업체(consumer)로서는 적시에 양질의 시스템을 공급 받아 업무의 효율성을 향상시킬 수 있다면, 프로젝트를 성공적으로 수행했다고 할 수 있겠다.

그렇다면 프로젝트를 성공시키는데 가장 역점을 두어야 할 부분은 어디인가? 그것은 보는 사람의 관점에 따라 다르겠지만, 최근 소프트웨어 공학(software engineering)의 추세를 살펴 보게 되면, 점차 그 해결 방안을 소프트웨어 생명주기(software lifecycle) 전체에 걸쳐 체계적인(systematic) 프로세스를 정립하고, 그 중에서도 프로젝트 초기에 수행하는 활동을 개선하여 시스템의 품질을 높이는 방향으로 가고 있다.

본 논문에서는 이러한 프로젝트 초기 프로세스의 중요성을 인식하고, 국내에서 시스템 개발에 참여했던 경험이 있는 사람이라면 누구라도 한번쯤은 부딪쳤을 만한 문제점들을 살펴 보고, 그러한 문제들을 마르미-III 의 “요구획득(Requirements)” 단계에서 제시하는 각종 활동(activity)과 작업(task)을 통해 어떻게 필요한 산출물(artifacts)을 효과적으로 구축하여 초기

프로젝트를 진행할 수 있는지, 실제 프로젝트 적용 사례를 소개하려고 한다.[1]

2. 본론

2.1 일반 문제 도메인(Problem Domain)

프로젝트 초기 “요구획득” 단계에서 언제나 부딪치는 문제중의 하나는 고객의 요구사항 대부분이 구두(verbal)로 전달되고, 그것을 회의록과 같은 문서로 기록하는 과정에서 고객의 최초 의도가 완전히(completely) 기록되지 못한다는 점이다. 더욱이 그것을 UML 과 같은 그래픽 모형화 언어로 모형화 하는 과정에서는 그 왜곡이 한층 심화되는 경향이 있다.

유스케이스 작성시 대표적인 오류의 사례로는 시스템 분해를 정보 처리 단위의 CRUD(Create, Retrieve, Update, and Delete) 수준의 기능 분해(functional decomposition)까지 진행하는 점이다. 그것은 필연적으로 유사한 유스케이스를 반복해서 생성하게 되고, 결국에는 통제 불가능하게 많은 유스케이스를 쏟아내어 문제를 더욱 악화시키게 된다.

그리고 중소규모의 프로젝트를 진행할 때 대체로 목격하게 되는 현상으로서 프로젝트 단계별로 구성원들이 수행해야 할 역할(role)을 모호하게 정의하는 분

제도 자주 볼 수 있다. 역할과 관련해서 주목할 사항은 중소기업의 프로젝트에서는 순수하게 업무 전문가(domain expert, business analyst)로서만 역할을 수행할 인력이 프로젝트 구성원에 포함되지 않는다는 것이다. 이것을 프로젝트 참여자 중 선임급이 역할을 대신할 수도 있지만, 고객사나 개발사의 그러한 선임 자급들은 각자 해당 프로젝트에서 수행해야 할 관리 역할의 비중이 높기 때문에 근본적으로 업무 전문가로서의 특정 역할에 전념하기에는 역부족이며, 특정 도메인을 제외하고 전문성을 체계적으로 습득하고 경험을 축적하여 객관적으로 검증할만한 전문가는 찾아보기 힘들다. 더욱이 시스템 개발 프로젝트에 참여하는 업무 전문가는 기본적으로 전산이나 시스템에 대한 식견(knowledge)도 겸비해야 한다.

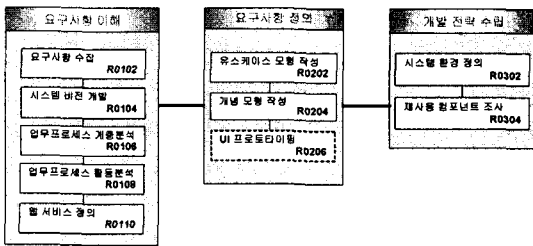
또 다른 문제점이라고 하면 인터넷 보급이 확산되면서 각종 새로운 플랫폼이나 개발 환경(Integrated Development Environment)을 도입하는 것에 대해서는 주저하지 않는 기업들이 정작 그러한 과정이나 절차를 보다 체계적으로 지원하는 적절한 도구를 도입할 때는 인색한 편이다.

이러한 현상은 국내의 기업 문화와도 크게 무관하지 않다. 가시적이고 단기적인 성과에 연연하여, 장기적인 안목에서 시스템의 효율성이나 활용도를 고려하지 않는 경향이 있다.

다음 절에는 지금까지 제기한 여러 문제점들을 해결하는데 적용할 마르미-III 의 “요구획득” 단계에 대한 내용을 2.2 절에서 간략히 살펴 보고 2.3 절에서는 실제 적용 사례를 기술하였다.

2.2 마르미-III 요구획득 단계 개요

마르미-III (버전 4.0) 전체 개발 공정 중에서 “요구획득” 단계 구성은 아래 그림과 같다.



<그림 1. 마르미-III 요구획득 단계>

“요구획득” 단계는 크게 세 가지 활동 즉, “요구사항 이해”, “요구사항 정의”, “개발전략 수립”으로 구성되어 있으며, 각 활동의 명칭은 각 활동을 통해 달성해야 할 목표로 간주할 수 있다. “요구사항 이해” 활동에서는 프로젝트를 수행하게 될 해당 도메인 업무(business)를 계층(hierarchy) 및 활동(activity)으로 구성하여 분석하게 되고, “요구사항 정의” 활동을 통해서 수집된 요구사항을 유스

케이스 모형으로 모형화하고, “개발전략 수립”을 통해 본격적인 아키텍처 단계로의 전환을 준비하게 된다.

마르미-III 는 최근의 CBD(Component Based Development) 방법론들이 공통적으로가치는 반복적(iterative)이고 점진적(incremental)인 개발 지향 속성을 가지고 있다.

“요구획득” 다음 단계인 “아키텍처” 단계에서 최초 수행하게 되는 활동은 “요구사항 분석” 활동이다. 즉, “요구획득” 단계에서 고객 요구사항과 관련된 모든 작업이 완료되는 것이 아니라, “아키텍처” 단계에서도 요구사항과 관련된 작업-여기서는 분석-은 계속된다.

다음절에서는 마르미-III “요구획득” 단계에서 제시하는 이러한 방안(guideline)을 활용하여, 어떻게 앞에서 제시한 문제점들을 극복하면서 프로젝트를 진행했었는지 그 적용 사례를 기술하였다.

2.3 적용 사례(Solution Domain)

- 사례 개요(case overview)

A 기업의 전체 전산 시스템 중 자재 구매 및 회계 관리 시스템을 마르미-III 개발방법론을 커스트마이징하고 적용하여 웹 기반 시스템으로 재구축하는 것으로서, 재무회계의 일반적인 모든 기능과 기본적인 관리회계 기능을 구현하는 일이었다. 프로젝트 투입 인력은 고객사의 전산실 담당자들을 제외하고 개발 업체로부터 총 15 여명의 프로젝트 상주 인력이 투입되었고, 총 개발 기간은 9 개월이 소요되는 프로젝트였다.

- 구체적인 접근법(approach)

프로젝트 팀원들이 방법론에 익숙하지 않았기 때문에, 매주 1 회씩 정기적인 워크숍을 통해 진행중인 업무 파악 내용의 유스케이스화 작업을 비롯하여 UML 에 대한 전반적인 교육을 진행하였다.

일주일 동안 업무 파악을 진행한 후 워크숍을 통해 업무 내용을 정리하였다. 이와 함께 총괄 시스템 분석가는 획득된 만큼의 요구사항을 가지고 진행할 수 있는 단계별 유스케이스 작성 방안과 기법을 교육하고, 다음 워크숍 때 유스케이스 모형을 완성하여 발표하도록 하였다.

멀티미디어와 참고서적의 활용

사람의 말을 글로 옮기게 되면, 필연적으로 기록되지 못하는 부분이 생기게 되고, 특히 담당 업무를 완전히 파악하지 못하면 업무상으로 매우 중요한 내용임에도 불구하고 기록하지 않고 지나쳐 버릴 가능성이 있다. 이러한 문제를 보완하기 위해서 담당자들에게 녹음기를 활용하도록 권고하였다.

참고서를 활용한 것도 효과가 좋았다. 유스케이스 작성 시 의문사항에 대해서는 간략하게 본질과 해결

방안만을 제시하고, 구체적인 내용은 추천한 서적에서 해당 문제를 언급하고 있는 부분을 알려주어 보다 심도 있게 문제를 고민한 후 업무분석가 스스로 문제를 해결해 나갈 수 있도록 하였다. 이렇게 함으로써 총괄 시스템 분석가는 당연한 전체 업무 파악과 통합에 더 많이 집중할 시간을 가질 수 있었다.

단계별 개개인의 역할 정의

팀 구성원들을 최대한 활용하기 위해 개개인의 특성과 경험을 점검하고, 프로젝트 단계별 임무분장을 정리하였다. 특히, 고객 전산 책임자의 요청으로 고객사 전산 담당자들을 개발 업체 직원들과 동일한 수준의 업무를 분장하기로 하였다. 일단, 업무 분야별로 담당을 결정하고 간단한 파일을 작성하여 공유 서버에 자료를 올려 관리하였다.

첫 워크숍 때 마르미-III 방법론의 전반적인 흐름을 소개하고, 각 단계별로 분장된 업무와 관련해서 각자 수행할 (다중)역할에 대한 내용을 설명하였다. 가령, 프로젝트 투입 시에는 그저 단순한 개발자로 임무가 주어졌지만, 프로젝트의 “요구획득” 단계에서는 시스템 분석가로 역할을 수행해야 하고, 초기 업무 분석이 완료된 후 “아키텍처” 및 “점진적 개발” 단계에 가서는 본격적으로 개발자로서의 역할을 수행하게 된다는 사실을 전달하였다.

유스케이스 작성 지침

개발자들이 유스케이스 작성을 정보공학적 사상에 근거하여 작성하려는 경향이 있는데 이는 유스케이스 작성시 피해야 하는 CRUD 수준의 기능 분할을 초래하게 된다. 따라서 다음과 같은 몇 가지 지침을 제시하였다.

- 유스케이스 명칭 작성 시 CRUD 용어를 사용해서는 안 된다.
- 유스케이스 명칭 작성 후, 시스템 사용자 입장에서 해당 유스케이스를 실행하고자 하는 목적이 무엇인지 간략하게 기술한다.
- 해당 유스케이스가 하나의 서비스로서 발생하기 위한 조건(사전조건)과 종료 후 변경되는 사항(사후조건)에 대해서 기술한다.
- 데이터 관점은 멀리하고 유스케이스의 기본흐름만을 작성하되, 자신이 앞에 놓인 컴퓨터와 대화(communication)를 한다고 생각하고 문장으로 기술해야 한다.

요구사항 수집(requirements gathering)

고객의 요구사항이 좀처럼 체계적으로 제공되지 않는 것이 국내 프로젝트에서 흔히 일어나는 현상이다. 여기서 “체계적”이란 말은 시스템화 하려고 하는 업무의 단위 업무 별 필요한 기능을 빠짐없이 기록하여 제공하는 것을 의미한다.

고객의 요구사항은 주로 회의를 통해 수집된다. 회의록에 기록된 내용은 일정한 양식의 문서를 통해 일차적으로 분류하여 기록하게 된다.

“요구사항기술서”를 통해 요구사항을 정리하다 보면 중복되거나 추가적인 확인이 필요한 사항들을 발견하게 되고, 고객의 요구사항을 보다 심층적으로 접근할 수 있는 실마리를 제공한다. 추가 확인이 필요한 사항을 구분하기 위해 마르미-III 에서 제공하는 기본 요구사항기술서 양식에 “추가확인” 항목을 추가하였다.

회의록만으로 모든 고객 요구사항을 추출할 수는 없는 경우 설문지를 활용한다. 설문지 작성시 반드시 고려할 사항은 설문지를 통해 고객으로부터 유도하려는 내용이 무엇인지 사전에 미리 계획하고 설문 항목을 구성해야 한다는 것이다. 그래야만 앞에서 두서없이 획득된 요구사항을 보강하여 보다 유기적으로 파악할 수 있게 요구사항을 획득할 수 있기 때문이다.

유스케이스 목록 통합 조정

요구사항기술서를 통해 고객의 요구사항을 정제하였으나 문서 템플릿이 단순 문서 파일로 만들었기 때문에 유기적으로 요구사항을 추적하기 어려웠다. 당시 프로젝트에는 관련 CASE 도구의 활용이 고려되지 않았기 때문이다. 그래서 범용적으로 활용되는 스프레드시트인 MS 사의 엑셀을 이용해 요구사항기술서의 내용을 정리하였다. 엑셀에서 제공하는 간단한 매크로(macro) 기능을 활용하면 기본적인 요구사항 관리가 가능할 것으로 판단했다.

	A	B	C	D	E	F	G	H
1						Requirement Registry		
2	No	FQ	Sub	Who	Identified	Description/Description	Solution	Filed
3	1F	As		XXX	이항목은 [1]에 언급된 동일한 자산에 대한 1개의 원래번호로 묶어서 관리할 수 있도록 해야 한다. 한편, 이 항목을 수정할 수 있는 권한을 관리자로 한정한다.	기존 시스템 데이터 전송 문제, 개발할 시스템은 개발로 관리		
4	2F	As		XXX	이항목은 [1]에 언급된 자산에 대한 2개의 원래번호로 묶어서 관리할 수 있도록 해야 한다. 이항목은 [1]에 언급된 자산과 동일하다.	(자산-취득등록) 기본 운용 4		
5	3F	As		XXX	이항목은 [1]에 언급된 자산에 대한 3개의 원래번호로 묶어서 관리할 수 있도록 해야 한다. 이항목은 [1]에 언급된 자산과 동일하다.	자산관리코드(세사 관리, 등록, 변경, 삭제, 조회, 등)의 업무 기능		
6	4F	As		XXX	이항목은 [1]에 언급된 자산에 대한 4개의 원래번호로 묶어서 관리할 수 있도록 해야 한다.	자산-취득등록(취득, 등록, 변경, 삭제, 조회)의 업무 기능 2		

<그림 2> 엑셀을 이용한 요구사항의 분류>

위의 <그림 2>의 세부항목을 보게 되면, 기존의 요구사항기술서의 항목과는 약간의 차이가 있는데, 각 항목을 살펴 보면, “FQ”는 해당 요구사항(Requirement Description)의 유형(type)을 기능(functional) 및 비기능적(non-functional) 요구사항으로 분류하고, 비기능적 요구사항은 시스템의 기능성(functionality)을 제외한 시스템의 품질(quality)로 규정하였다. “Sub”는 업무 기능의 단위 업무 패키지를 의미한다. 가령, “As”는 “Asset(고정자산)”의 약어로서 일반적인 재무회계 업무 중 고정자산 업무와 관련된 요구사항으로 분류한 것이다.

이렇게 정리된 요구사항은 약 275 개 정도가 도출되었고, 각 항목에 대해서는 각 단위 업무별 현업 담당자들과 개발팀 전원이 참석한 가운데, 하나하나 내용을 점검하며 중복된 내용을 통합하고, 정리하여 200 개 정도의 요구사항을 도출하였다.

전체 이해당사자(stakeholder)의 합의(consensus)를 통해 결정된 요구사항을 다시 비기능적 요구사항 즉, 시

스텝 품질 요구사항을 별도로 도출하고, 나머지 기능적 요구사항은 사전에 분류한 단위 업무별로 분류하여 정리하였다. <그림 3>과 <그림 4>은 각각 정리된 요구사항을 보여준다.

Req. No.	Requirement Description	Priority	Type	Impact
110	주요 소스 자료 사용자 대관영역 성능상의 부하가 발생할 것으로 판단된다.	고	기능	중
126	문자 인코딩 처리시 성능이 많이 떨어질 소지가 있다.	중	기능	중
137	간혹 보고서 출력시 역발이 많이 발생할 소지가 있다.	중	기능	중
140	메시저 서버는 보안이 강화된 메일 송수신 기능을 가질 수 있다.	중	기능	중
160	주요 소스 인코딩 및 디코딩 시 성능이 약한 지점에서 필요한 일정을 반영하여 개발 가능해야 한다.	중	기능	중
184	일기장과의 방화(중독, 징벌, 비양자, 침입, 연상각 등)에 자주 영향을 소지가 있다.	고	보안	중

<그림 3. 비기능적 요구사항의 도출>

No.	Requirement Description	기능적	비기능적	중요도	개발도
110	주요 소스 자료 사용자 대관영역 성능상의 부하가 발생할 것으로 판단된다.	X	X	고	중
126	문자 인코딩 처리시 성능이 많이 떨어질 소지가 있다.	X	X	중	중
137	간혹 보고서 출력시 역발이 많이 발생할 소지가 있다.	X	X	중	중
140	메시저 서버는 보안이 강화된 메일 송수신 기능을 가질 수 있다.	X	X	중	중
160	주요 소스 인코딩 및 디코딩 시 성능이 약한 지점에서 필요한 일정을 반영하여 개발 가능해야 한다.	X	X	중	중
184	일기장과의 방화(중독, 징벌, 비양자, 침입, 연상각 등)에 자주 영향을 소지가 있다.	X	X	고	중

<그림 4. 단위 업무별 요구사항 정리>

여기서 비기능적 요구사항은 크게 5 가지로 분류하여 정리하였는데, 성능, 변경용이성, 가용성, 보안, 그리고 사용편리성이 그것이다. 위의 <그림 4>는 기능적 요구사항들을 <그림 2>의 "Sub" 항목별로 분류하여 정리한 것이다. 그리고 <그림 3>과 <그림 4>의 "No" 항목은 <그림 2>의 "No" 항목 번호와 동일하다.

이렇게 요구사항을 정리함으로써, 각 단위 업무에 대한 유스케이스를 작성해야 할 담당자들이 추적관리를 해야 할 요구사항 획득이 완성되고, 지속적인 관리를 통해 실제적으로 고객의 요구사항이 유스케이스에 반영되고 있는지 여부를 현실적으로 관리할 수 있는 토대가 마련되었다.

아래 <그림 5>의 도표는 회계 도메인의 전체 유스케이스 목록에 대해 각 유스케이스의 업무 처리 범위를 현업 담당자들에게 제시하고, 각 담당자들이 해당 유스케이스와 관련하여 자신이 처리하는 업무의 가중치를 부여하도록 하여 통계를 도출한 것이다. 아래 <그림 5>의 우측 상단에 화살표로 표시된 수치는 상대적으로 임계치(0.5) 이상의 가중치를 갖게 되는 유스케이스임을 나타낸다.

사실 아래 <그림 5>를 작성하기 이전에 현업 담당자들과 개발 팀원들 간에 협의를 통해 요구사항의 완성도와 시스템 아키텍처를 검증하기 위해 아키텍처 프로토타이핑을 할 때 구현 대상이 될만한 주요 유스케이스 후보들은 언급이 되고 있었다. 하지만, 어느 누구도 단정적으로 결정을 내리지 못하는 상황이었다.

유스케이스 A				유스케이스 B				유스케이스 C				총합	
소계	비율			소계	비율			소계	비율			총합	비율
0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.11	0.10	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.10
0.18	0.15	0.03	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.15
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

<그림 5. 업무 가중치 분석 도표>

그런 상황에서 위와 같은 절차를 통해 우리는 아키텍처 프로토타이핑을 할 주요 유스케이스 후보를 확실하게 결정할 수 있었다. 개발팀에서 일반적으로 프로토타입을 선정하여 진행하는 것도 아니고, 고객의 의사를 최대한 수용하여 진행되는 과정을 고객들에게 제시할 수 있었으며, 프로토타입 선정의 정당성을 확보할 수 있었다.

3. 결론

소프트웨어 공학이 점점 세분화 되고 전문화 되어 가면서 최근 중요시 되는 분야 중의 하나가 요구공학 분야이다. 그만큼 프로젝트의 성공 여부는 프로젝트 초기에 고객의 요구사항을 어떻게 잘 반영하는냐에 따라 크게 좌우될 수 있음을 반증하는 것이기도 하다.

일반적으로 고객의 요구사항이 시스템화 하기에 충분한 정도의 정형화된 형태로 제공되는 경우는 흔하지 않으며, 정형화된 요구사항을 강요하는 것은 현실적인 대안이 되지 못한다. 우리는 본 논문에서 제시하는 접근법을 이용하여 소기의 목적을 이룰 수 있었다. 그러나 스프레드시트로 하는 데는 한계가 있었다. 그 보다는 적절한 CASE 도입에 대한 필요성이 대두되었다.

문제는 구두나 문장 형태로 자유분방하게 제시되는 요구사항을 어떻게 체계적으로 유도하여 고객과 개발팀원들이 모두 공감할 수 있는 방안을 도출하는냐에 있다. 이번에 소개하는 적용 사례는 특별한 CASE 도구의 도움을 받아 원활하게 진행하지 못했지만, 나름대로 주변에서 강구할 수 있는 최대한의 자원을 이용하여 고객과 공감대를 형성하며 프로젝트를 진행한 사례로 보여진다. 요구사항을 관리하는데 어떤 CASE 도구를 활용할 것이냐는 중요하지 않다. 중요한 것은 무엇보다도 고객과의 공감대를 형성할 수 있는 논리적이고 체계적인 고객 요구사항에 대한 접근법이라고 해야 할 것이다.

참고문헌

- [1] 컴포넌트 기반 시스템 개발 방법론 마르미-III," 프로젝트관리기술 논문집" 제 4 권 p1~14