MAPPING TABLE을 이용한 금융 PRODUCT FACTORY 분석기법

박진희 O,a, 유준범 b, 차성덕 a

a 고려대학교 컴퓨터학과, b 건국대학교 컴퓨터공학부

jin20000@korea.ac.kr, jbyoo@konkuk.ac.kr, scha@korea.ac.kr

An analysis method on finance product factory using mapping table

Jinhee Park O,a , Junbeom Yoo b, Sungdeok Cha a

aDept. of CS and Engineering. Korea Univ.

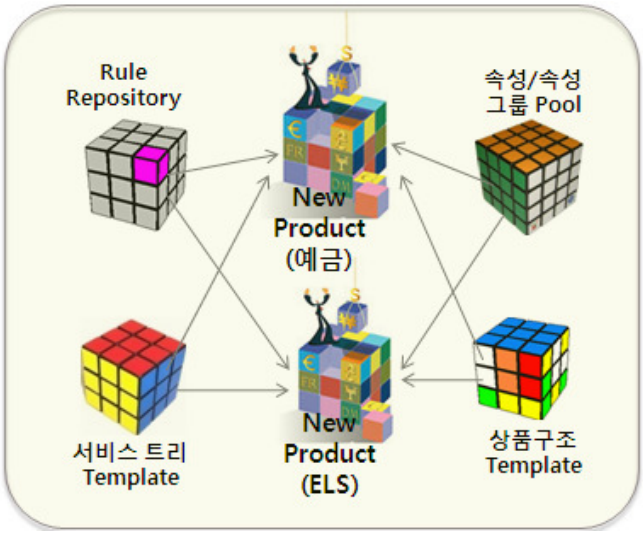
bDiv. of CS and Engineering. Konkuk Univ.

요 약

금융 Product factory는 금융상품 구성을 위한 각 컴포넌트들을 정의하고, 상품개발이 필요할 때 필요한 컴포넌트를 조합하여 쉽고 빠르게 상품을 개발할 수 있도록 지원하는 시스템을 말한다. 이와 같은 시스템의 구축을 위해서는 분석, 설계자의 상품에 대한 통찰력과 함께, Legacy System을 효율적으로 분석하여 Rule과 데이터로 모델링 할 수 있는 방법이 필요하다. 본 논문은, Legacy system의 분석정보를 Mapping Table로 데이터화하여, factory 모델링 대상 정보를 체계적으로 분석하고 분류하는 방법을 제시하였다. 또한, 국내 금융기관의 분석기법 적용사례를 통하여 본 연구의 구현효과를 검증하였다.

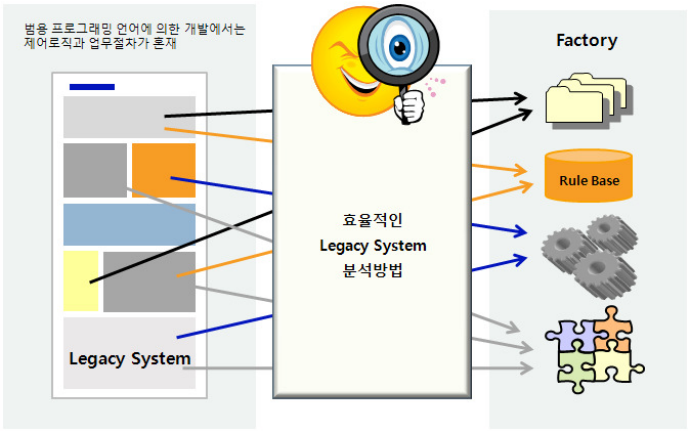
1. 서 론

금융 Product factory는 금융상품 구성을 위한 각 컴포넌트들을 정의하고, 상품개발이 필요할 때 필요한 컴포넌트를 조합하여 쉽고 빠르게 상품을 개발 할 수 있도록 지원하는 시스템을 말한다. [1]



[그림1] 금융 Product factory의 개념

성공적인 금융 Product factory 시스템 구축을 위해서는 1)상품별 업무규칙을 Parameter Rule화 하여 프로그램을 세분화시키고 2)기능중복을 배제하여 기능관점에서 업무를 통합 할 수 있어야 하며 3)공통된 기능을 특정상품 및 업무에 관계 없이 재사용 할 수 있어야 한다. [2] [3]



[그림2] 제안 분석방법

이런 요건을 충족하기 위해서는 분석, 설계자의 상품에 대한 통찰력과 함께, Legacy System을 효율적으로 분석하여 Rule과 데이터로 모델링 할 수 있는 방법이 필요하다.

본 논문은, 범용 프로그래밍 언어로 개발되어 제어로직과 업무절차가 프로그램 내에 혼재되어 있는 1) Legacy system을 분석하여, 2)Mapping Table을 작성하고 3)이를 기준으로 정보를 체계적으로 분석하고 분류하여, Rule과 속성, 모듈로 모델링 하는 분석방법을 제시하였다.

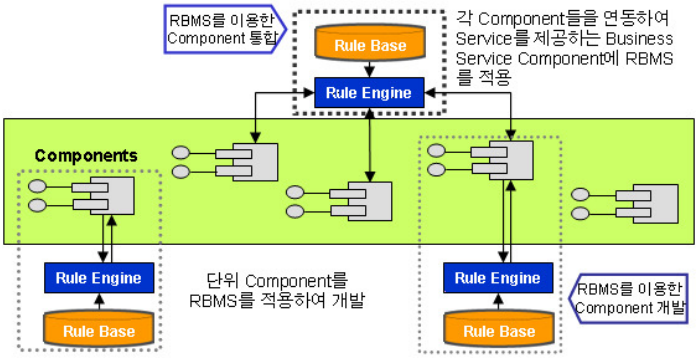
2. 관련연구

2.1 CBSE

소프트웨어 개발을 재사용이 가능한 소프트웨어 component에 기반을 두도록 컴퓨터 기반 시스템들의 설계와 구축을 강조하는 절차로서, CBSE를 위해서는 소프트웨어 개발 방법, 도메인 공학, 개발 환경 등 다방면에서의 근본적이 개념 재설정과 협력이 요구된다. [4]
소프트웨어를 component화 함으로서 1)소프트웨어개발의 자동화와 생산성 향상, 2)소프트웨어의 품질 향상, 3)표준화와 다수 사용자의 변경요청에 응대 가능, 4)불특정 다수의 다회 사용으로 long term에 걸쳐 많은 이익이 발생하는 장점을 가지고 있다.

2.2 RBMS

기업의 핵심비즈니스 룰을 시스템화하고, 활용함에 있어, 개발자, 사용자, 운용/관리자 등 3자 측면에서 지원하는 통합 비즈니스 룰 관리 환경이다.



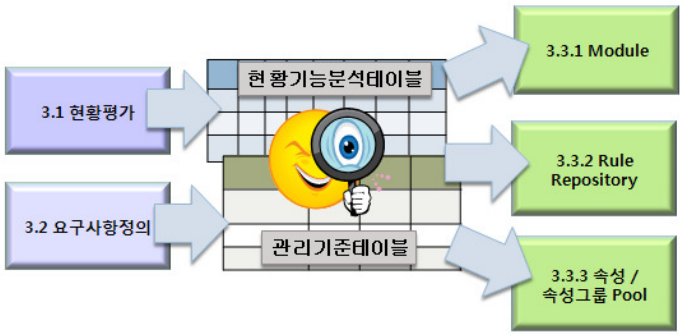
[그림3] RBMS Component 적용 방법

Rule base란 기업 내에 존재하는 업무규칙, 절차, 내용, 담당자의 지식 및 노하우 등의 Business 로직을 조건문 형태의 Rule로써 표현하여 담은 것을 의미한다. [5]

2.3 Software Factory

Software Factory는 확장 가능하게 만들어진 도구들과 프로세스, 그리고 자동으로 원형 제품을 적합하게 수정하고, 조립하며, framework-based 컴포넌트들을 구성함으로써 다양한 제품을 만들고, 유지 보수할 수 있도록 software factory schema에 기반한 software factory template을 사용하는 소프트웨어 생산라인이다. [6]

3. Mapping Table을 이용한 Product Factory 분석기법



[그림4] 분석방법 개념도

Legacy system의 분석정보를 Mapping Table로 데이터화 하여, factory 모델링 대상 정보를 체계적으로 분석하고 분류하는 방법을 제시 한다.

3.1. Legacy System 현황평가

Legacy System을 분석하는 단계로, 어플리케이션, 프로세스, 데이터 3가지 관점으로 분석을 하고 그 결과를 현황 기능분석 Table에 작성 한다.

3.1.1 어플리케이션 분석

프로그램 목록을 유형별로 분류하고 유형별 프로그램의 주요기능 및 부가기능을 파악한다.

3.1.2 프로세스 분석

(1) 업무규정을 분석하고, 업무규정이 명확하게 정의되지 않은 프로세스는 리스트로 작성한다.
(2) 유형별로 분류된 프로그램을 프로세스가 동일한 것으로 재분류한다.
(3) 프로세스를 기능단위로 구조화한다.
(4) 기능단위의 처리내용을 분석한다.

3.1.3 데이터 분석

(1) 테이블 관리항목의 속성을 분석하여 도메인 정의 대상을 분류한다.
(2) 상품별 계수현황을 조사한다.
상품의 유지여부 및 비율을 확인하고 예외 프로세스의 처리방안을 결정하는 판단기준으로 사용 할 수 있다.

3.1.4. 현황 기능분석 Table 작성

[표1] 현황기능분석테이블

	프로세스	기능	기능설명	참고 Data	참고 항목1	참고 항목2	계산식
상품1							
상품2							

[표1]과 같이 유형별로 분류된 프로세스의 기능단위 처리내용을 항목단위로 구분하여 Table에 상세하게 작성한다.

3.2. 요구사항정의

사용자의 요구사항에 충족하는 주기능을 정의하고, 관리 및 변동의 기준을 정의하는 절차이다. [7]

3.2.1 기능요구사항 정의

(1) 프로세스가 명확하지 않은 상품의 업무규정을 정비한다.
(2) 프로세스/기능/항목 각각의 요소에 대한 변경 주체(내부. 외부)를 정의한다.
(3) 기능분석 항목에 대하여 변동 유형 및 주기를 정의한다.

3.2.2 인터페이스 요구사항정의

주기능을 정의하고, 연계업무의 사용용도를 분석하여 기능의 분류기준항목으로 사용한다.

3.2.3 기능구현 범위결정

(1) 예외처리 방식에 의하여 프로세스의 복잡도가 높아지는 경우 처리대안을 검토한다.
데이터 분석을 통해 1)현재 판매되지는 않으나 좌수를 보유하고 있는 계좌이거나 2)이후 판매가 중단된 상품의 유지를 위한, 행정적 처리 가능방법을 검토한다.
(2) 프로세스가 복잡하고 변경 가능성이 많지 않은 경우는 별도의 관리모듈로 구성하는 방안을 검토한다.

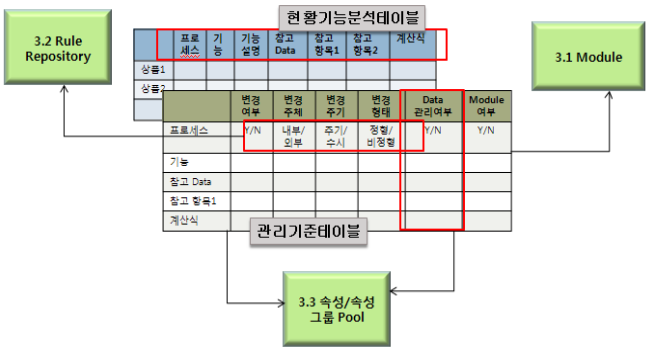
3.2.4 관리기준 *Table* 작성

[표2] 관리기준테이블

	변경 여부	변경 주체	변경 주기	변경 형태	Data 관리여부	Module 여부
프로세스	Y/N	내부/ 외부	주기/ 수시	정형/ 비정형	Y/N	Y/N
기능						
참고 Data						
참고 항목1						
계산식						

(1) 프로세스, 기능, 현황평가 분류기준항목들에 대한 요구사항 정의 기준을 Table로 작성한다.
(2) Legacy system에서의 Data관리 여부, Module단위 처리여부도 기준항목으로 추가하여 재구성여부를 판단한다.

3.3 *Mapping table*을 분석을 통한 신논리모델 구축



[그림5] Mapping Table

Text 형태의 분석자료를 Mapping Table 화하여 프로세스, 기능 및 처리 항목단위 내용까지 가시화 함으로써 1)항목단위 코드화가 가능하고, 2)기능분류 기준서로 활용할 수 있으며, 3)기능 누락 방지 효과와 4)패턴분석을 용이하게 할 수 있는 장점을 가지고 있다.
분석자료는 이후 이행 테스트를 위한 테스트 기준서로 활용이 가능하다.

3.3.1 *Module*

3.3.1.1 단위업무 프로세스모듈 단위 결정

유형별로 분류된 프로세스에 분류기준을 적용하여 모듈단위를 결정한다.

3.3.1.2 기능모듈 단위 결정

(1) 테이블에 작성된 기능들은 모듈화 대상으로 기능 독립적으로 소단위 기능으로 분리한다.
(2) 동일한 기능이라도 잠재적 위험을 제거하기 위하여 변경주체 또는 변경형태가 다른 경우 분리한다.
(3) 테이블 분류항목에서 구조화된 패턴을 가지고 있거나, 일정한 변경 형식을 가지고 있는 경우 기능모듈로 작성한다.

3.3.2 *Rule Repository*

(1) 상품별 프로세스 Rule 정보관리
상품코드, 프로세스, 기능을 도메인으로 정의하고, 상품별 처리프로세스를 관리하는 Rule 정보를 정의한다.
(2) 기능분석 테이블의 항목속성을 관리하는 기능단위 Rule 정보를 정의한다.
(3) 기능분석 테이블에 작성된 관리기준 분류항목에서 변경기준항목을 도출하고, 패턴을 분석하여 Rule 정보로 정의한다. [8]

3.3.3. 속성 / 속성그룹 *Pool*

(1) 프로세스, 기능을 도메인으로 정의하여 관리한다.
(2) 기능분석 테이블의 항목별 속성을 코드로 정의하여 관리한다
(3) Legacy System의 Data Structure에 관리기준 요소를 반영하여 Re- structuring 한다.

4. 분석기법 적용사례

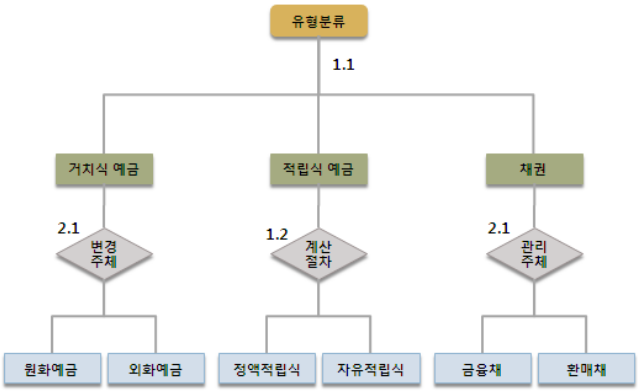
4장은 국내 금융기관의 저축성 상품 이자계산 시스템 구축시 분석기법을 적용한 사례이다.

4.1 분류기준 적용을 통한 모듈화 단위결정 방법

저축성 상품은 일반적인 3가지 형태로 분류가 가능하다.
▪ 거치식예금: 목돈을 예치하고 이자수익을 목적으로 하는

예금

- 적립식예금: 목돈마련을 목적으로 원금을 분할 납부하는 예금
- 채권: 거액의 자금을 조달하기 위한 목적의 증권

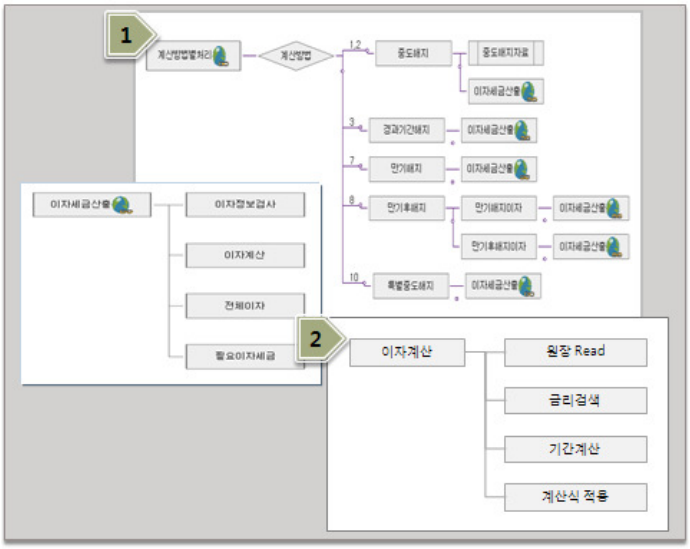


[그림6] 모듈화 단위결정

[그림6]은 관리기준 Table의 분류기준 항목을 적용하여 모듈화 단위를 결정하는 과정이다.

(1) 1.1 어플리케이션 분석으로 일반적인 저축성상품의 분류기준에 의하여 유형을 분류한다.
(2) 1.2 프로세스 분석과정으로 적립방식이 다르면 이자계산 프로세스도 달라지므로 프로세스가 동일한 것으로 재분류하는 과정이다.
(3) 2.1 기능요구사항에서 정의한 프로세스의 관리 및 변경 주체에 따른 분류과정이다. 상품의 변경주체가 다르면 관리의 기준이 달라지고. 관리부서가 다르면 변동의 형식이 상이 할 수 있다. 그러므로 모듈을 분류하여 관리하는 것이 변경의 용이성과 관리의 효율성을 높일 수 있다.

4.2 프로세스 기능단위 구조화



[그림7] 저축성상품의 이자계산 프로세스 분석

[그림7]은 프로세스를 분석하고 기능단위 Table 작성을 위하여 기능화단위로 구조화한 예로, 작성절차는 아래와 같다.

(1) 유형별 프로그램을 Function단위로 분류하여 프로세스를 정의한다.
(2) 동일한 프로세스는 공통 기능으로 분류하고, 처리 내용을 정의한다.

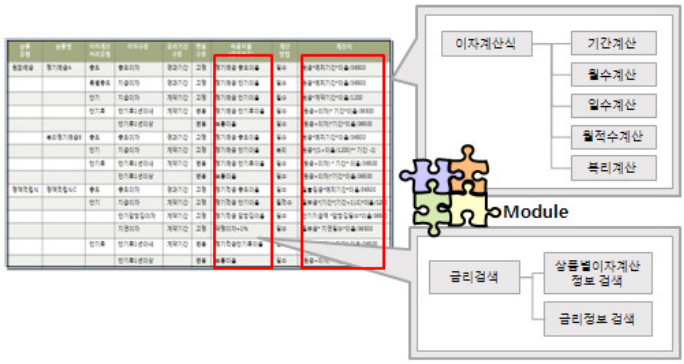
4.3 현황 기능분석 *Table* 작성사례

상품 유형	상품명	이자계산 처리유형	이자구분	금리기간 구분	변동 구분	적용이율 (금리코드)	계산 방법	계산식
원화예금	정기예금A	중도	중도이자	경과기간	고정	정기예금 중도이율	일수	원금*예치기간*이율/36500
		특별중도	지급이자	경과기간	고정	정기예금 만기이율	일수	원금*예치기간*이율/36500
		만기	지급이자	계약기간	고정	정기예금 만기이율	월수	원금*계약기간*이율/1200
		만기후	만기후1년이내	계약기간	변동	정기예금 만기후이율	일수	(원금+이자)* 기간*이율/36500
			만기후1년이상		변동	보통이율	일수	(원금+이자)*기간*이율/36500
	복리정기예금B	중도	중도이자	경과기간	고정	정기예금 중도이율	일수	원금*예치기간*이율/36500
		만기	지급이자	계약기간	고정	정기예금 만기이율	복리	원금*((1+이율/1200)** 기간 -1)
		만기후	만기후1년이내	계약기간	변동	정기예금 만기후이율	일수	(원금+이자) * 기간* 이율/36500
			만기후1년이상		변동	보통이율	일수	(원금+이자)*기간*이율/36500
정액적립식	정액적립식C	중도	중도이자	경과기간	고정	정기적금 중도이율	일수	월불입금*예치기간*이율/36500
		만기	지급이자	계약기간	고정	정기적금 만기이율	월수	월부금*(기간*(기간+1)/2)*이율/1200
			만기말할증이자	계약기간	고정	정기적금 말할증이율	일수	만기지급액 *말할증일수*이율/36500
			지연이자	계약기간	고정	약정이자+1%	일수	월부금* 지연일수*이율/36500
		만기후	만기후1년이내	계약기간	변동	정기적금만기후이율	일수	(원금+이자) * 기간* 이율/36500
			만기후1년이상		변동	보통이율	일수	(원금+이자)*기간*이율/36500

[그림8] 저축성상품 이자계산 현황기능분석 Table Format

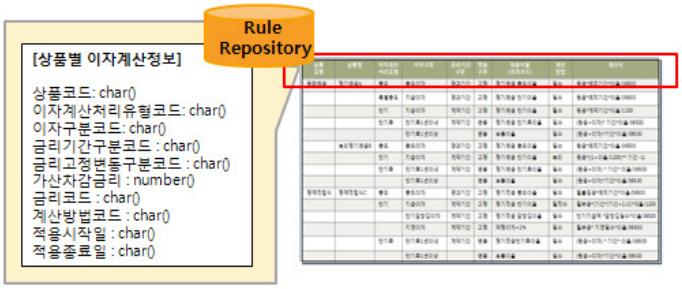
[그림7]의 과정을 거쳐 분석된 내용은 [그림8]의 예시와 같이 Table을 작성하여 기능별 처리내용을 항목구분에 따라 상세하게 기술한다.

4.4. *Table* 분석을 통한 신논리모델링



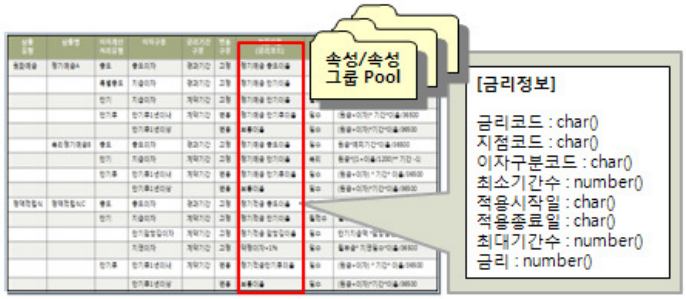
[그림9] 기능모듈 모델링

[그림9]는 Mapping table에서 패턴을 분석하여, 소단위 기능모듈로 모델링 하는 과정이다.
테이블에서 계산식 항목내용을 살펴보면 금액과 기간, 이율 3가지 속성으로 구성되어 있으며, 몇 개의 동일한 계산식이 반복 된다는 것을 알 수 있다. 일정한 형식을 갖는 내용은 소단위 기능모듈로 분리하여 일관성과 효율성을 도모한다.



[그림10] Rule Table 모델링

[그림10]은 기능분석 Table 항목을 기준으로 Rule 테이블을 모델링하는 과정이다.



[그림11] 속성테이블 모델링

[그림11]은 Legacy System에서 관리한 금리정보를 Rule과 분리하여 Re-structuring하는 과정이다.

4.5 적용 결과

저축성 상품 종류 : 680개	Legacy system	Product factory system
Process Module	97 개	25 개
Function Module	12 개	20 개
금리적용 방식	단순코드관리 금리 검색 기준 변경 또는 신상품 추가시 개별프로그램 수정 필요	금리적용 룰과 데이터로 관리 금리 검색 기준 변경, 추가시 데이터 변경으로 유연하게 적용

[그림12] 구현결과 비교

Mapping Table을 체계적으로 분석하고 분류하여, 동일 프로세스를 통합하고, 기능단위로 모듈화 함으로서, 유지보수의 효율성을 높였으며, 변동성이 많은 항목은 Rule과 속성으로 관리되어 상품 추가 및 변경이 용이한 시스템을 구축하였다.

유용한 Product factory system를 구축하기 위해서는 상품에 대한 통찰 능력과 설계능력, 모델링 능력도 중요하지만, 사용자 관점에서의 이해와 관리 효율성 측면을 고려할 수 있어야 하고, 전 단계에 걸쳐 현업과 지속적이고 긴밀한 협조관계를 필요로 한다.

5. 결론 및 향후 연구과제

최근 많은 금융기관들이 차세대시스템을 구축하면서 factory 개념을 도입하고 있지만, 분석, 설계의 경험을 가지고 있는 기술자가 많지 않다. 또한, Product factory 시스템을 가지고 있는 금융기관의 경우 구축된 정보는 내부의 경쟁 자산으로 인식되어 분석, 설계 노하우가 공유되지 못하고 있다. 따라서, 본 논문은 분석정보를 Mapping Table을 이용하여 프로세스, 기능 및 처리 항목단위 내용까지 가시화 함으로써 factory 대상 정보를 체계적으로 분석하고 분류하는 방법을 제시하였다. Legacy system을 기능단위로 분류하고 속성을 분석하는 것은 분석자의 능력에 따라서 많은 차이가 날 수 있다. 따라서, 향후 이와 같은 분석자에 따른 편차를 줄이기 위해서 Mapping table 속성 도출을 위한 기능단위 분류 방법에 대한 연구를 수행할 계획이다.

참고문헌

[1] 이성하, 주정은, 최성철, 구상회, 금융 프로덕트팩토리를 위한 복합상품 설계시스템 개발, 한국지능정보시스템학회논문지, 제10권 제2호(pp.39~51), 2004.11

[2] M.L.Griss, Software reuse: From library to factory, IBM SYSTEM JOURNAL, VOL 32, NO4, 1993.

[3] 김강태, 임베디드 소프트웨어의 재사용성 향상을 위한 리엔지니어링 프레임워크, 컴퓨터정보학회, 2008.08.

[4] 렌 베스, 폴 클레멘츠, 릭 캐즈먼, 소프트웨어 아키텍처 이론과 실제, 에이콘출판주식회사. 2007.05.

[5] Bouziane, M. and Hsu,C. A Rulebase Management System Using Conceptual Rule Modeling, International Journal on Artificial Intelligence Tools, Vol. 6, No.1,(pp37-61), 1997

[6] The Software Factory: Contributions and Illusions. I. Aaen, P. Bøttcher & L. Mathiassen. In: Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo, 1997.

[7] Ian Sommerville, Software Engineering (8th ed.), Harlow, England: Pearson Education, 2007

[8] Joshua Kerievsky, 패턴을 활용한 리팩터링, 인사이트, 2006.07