KIPS Tr. Software and Data Eng.
Vol.2, No.12 pp.841~848 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2013.2.12.841

Design of Efficient Inspection Scope in e-banking System

Park Hae Yoon† · Yoo Hae Young††

ABSTRACT

Nowadays, the finance changes so rapidly that the time period of developing a e-banking system has been shorter. As a result, the risk of developing in the new e-banking system has increased. Therefore financial institutions constantly ask to review the quality test of their system during the project development. However, from the developer's point of view, additional system quality inspection will delay the development time for a new project, so financial investors and developers, because this problem will cause conflict. In order to solve this problem, in this paper define priority that considered characteristics of e-banking system and thereby design range priority decision criteria for efficient code inspection. Even in the e-banking system development, it is expected that using newly designed range of code inspection will allow high efficient quality performance.

Keywords : e-Banking System, Inspection, Testing, Inspection Scope

e-뱅킹 시스템의 효율적인 인스펙션 범위 설계

박 해 윤† · 유 해 영††

요 약

오늘날 금융이 급격하게 변화함에 따라 e-뱅킹 시스템을 개발하는 프로젝트의 구축 기간은 점차 짧아지고 있고, 이로 인해 구축에서의 위험 또한 증가하고 있다. 때문에 금융사는 프로젝트 개발 중에도 시스템의 품질을 검토하길 요구한다. 하지만 개발사 입장에서 이와 같은 추가적인 품질 검토는 투입되는 노력으로 인해 개발을 지연시키는 요인이 되고, 결국 금융사와 개발사의 갈등을 야기한다. 이와 같은 문제를 해결하기 위해 이 논문에서는 e-뱅킹 시스템에 특성을 고려하여 우선순위를 정의하고, 이를 통해 효율적인 코드 인스펙션을 위한 범위 우선순위 판단 기준을 설계한다. 설계된 코드 인스펙션 범위를 이용하여 적은 노력으로 효율적인 코드 인스펙션을 수행할 수 있으며, 해당 범위를 활용하여 좀 더 고품질의 프로덕트를 개발할 수 있을 것으로 기대된다.

키워드 : e-뱅킹 시스템, 인스펙션, 테스팅, 인스펙션 범위

1. 서 론

오늘날 금융은 금융의 혁신, 규제 강화, 글로벌화, 업종 겸업화, 상품/서비스의 복합화, 금융기관 대형화 등에 따라 점차 급격하게 변화하고 있다[1]. 이에 대응하기 위한 e-뱅킹 시스템 역시 점차 복잡해지고 대형화되는 추세에 있다[2]. 하지만, 이와는 반대로 빠르게 변화하는 금융 환경은 더욱 더 민첩하게 e-뱅킹 시스템을 구축하도록 요구한다. 실제로 최근 이슈가 된 금융상품 인 '재형저축'을 온라인으로 가입할 수 있도록 제출서류를 전자화하고 e-뱅킹 시스템에 서비스를 추가하는데 안정화를 포함하여 4주 정도의 개발기간 만이 제시되었다.

이와 같은 현상은 e-뱅킹 시스템 구축의 위험을 증대시키고 있고, 개발기간의 지연 및 개발 시스템의 품질 저하, 금융사와 개발사 간의 소송 등을 야기하고 있다. 2012년 조사된 소프트웨어 납기 준수 현황에 따르면, Fig. 1과 같이 29.8%의 프로젝트가 납기일을 미준수하고 있었으며, 이 중 납기일 1~10% 초과는 46%, 11~20% 초과가 39%, 21~30% 초과는 15%의 빈도를 보인다[3]. 이와 같은 이유로 개발사는 품질 측정에 소홀해지는 경향을 보이고 있다. 하지만, 금융사는 개발기간의 단축과는 반대로 더 높은 품질의 프로덕트를 요구하며, 이를 위해 추가적인 품질 측정을 요구한다.

추가적인 품질 측정 방법에 하나인 코드 인스펙션(Code Inspection)은 개발팀에서 작성한 개발 소스 코드를 분석하여 개발 표준에 위배되었거나 잘못 작성된 부분을 수정하는 작업을 말하며[4], 이와 같은 코드 인스펙션은 개발사에게

† 준 회 원 : 단국대학교 컴퓨터학과 박사과정
†† 정 회 원 : 단국대학교 컴퓨터학과 교수
논문접수 : 2013년 8월 30일
수 정 일 : 1차 2013년 9월 30일
심사완료 : 2013년 10월 4일
* Corresponding Author : Park Hae Yoon(park8312@dankook.ac.kr)

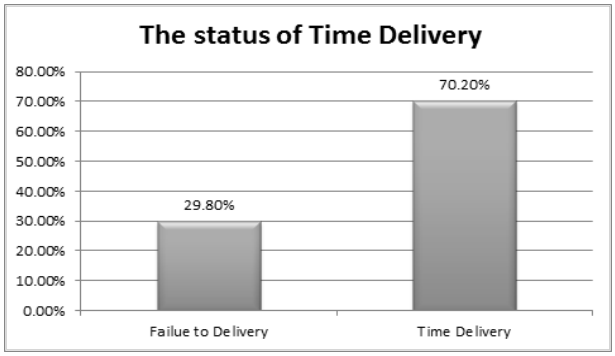

Fig. 1. The status of Time Delivery

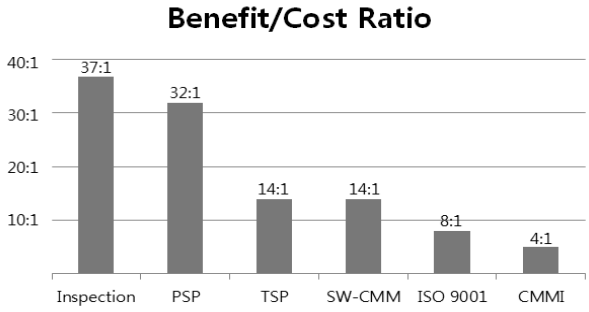

Fig. 2. Benefit/Cost Ratio Between Quality Techniques

많은 노력을 요구하고, 이를 수행하기 위해 프로젝트 전체의 개발 일정을 지연시키는 결과를 초래한다.

따라서 품질 측정에 소요되는 개발사의 노력을 줄이고, 금융사의 품질에 대한 우려를 감소시킬 수 있는 품질 측정 기법이 필요하다. 이에 따라 본 논문은 e-뱅킹 시스템의 코드 인스펙션을 효율적으로 수행하기 위한 인스펙션 범위의 우선순위 판단 기준을 설계하는 것을 목적으로 하고, 나아가 품질 측정에 우선순위가 되는 코드 인스펙션 범위를 확인하여 이 범위를 집중적으로 테스트함으로써 고품질의 e-뱅킹 시스템을 구축할 수 있음을 보인다. 이를 위해 본 논문은 다음과 같이 구성된다. 2장에서는 e-뱅킹 시스템에서의 테스팅 기법들과 효율적인 코드 인스펙션에 대한 기존의 연구들을 확인한다. 3장에서는 A은행의 e-뱅킹 시스템 구축 사례를 통해 e-뱅킹 시스템에서의 품질 측정에 우선순위가 되는 코드 인스펙션 범위를 도출하려는 환경을 제시하고, 효율적인 코드 인스펙션을 위한 범위의 우선순위 판단 기준의 설계를 보인다. 4장은 기존의 인스펙션과 설계된 효율적인 인스펙션 범위를 비교하여 코드 인스펙션의 결과를 보인다. 5장은 C은행의 e-뱅킹 시스템 재구축 사례를 통해 이 범위를 집중적으로 관리하였을 때 얻을 수 있는 효과를 제시한다. 그리고 마지막으로 6장에서는 결론을 보인다.

2. 관련 연구

코드 인스펙션은 개발팀에서 작성한 개발소스 코드를 분석하여 개발 표준에 위배되었거나 잘못 작성된 부분을 수정하는 작업을 말한다[5]. 코드 인스펙션은 소프트웨어의 품질을 확인하기 위해 수행할 수 있는 기법인, PSP(Personal Software Process), TSP(Team Software Process), SW-CMM(Capability Maturity Model), ISO9001, CMMI와 비교하여 이득은 낮은 편이지만 투자비용이 낮아 ROI(Return on Investment) 측면에서 가장 효율적인 기법이다[6]. 또한 다른 기법들에 비해 프로젝트 중간 단계에서 수행하기에 용이하다는 장점이 있다.

Oliver Laitenberger[7]은 소프트웨어 품질에서 코드 인스펙션과 구조적 테스팅의 효과를 확인하기 위한 연구를 보인다. 이를 위해 C코드 모듈에서 테스트 표준을 서로 다른 커버리지로 코드 인스펙션과 구조적 테스팅을 수행하였다. 이 때 테스트를 수행하는 범위는 바이너리 범위, 루프 범위, 다중 조건 범위, 상관관계가 있는 범위로 테스트를 수행하였으며, 이 때 인스펙션은 바이너리 범위에서 가장 많은 결함을 발견하였고, 루프 범위에서 가장 적은 결함을 발견하였다. 이 연구에서 구조적 테스트는 몇몇 특별한 클래스에서 결함 발견에 취약하며, 인스펙션이 더 우수한 결함 발견을 수행할 수 있음을 확인하였고, 인스펙션이 특정 범위에서 더 높은 결함 발견을 수행할 수 있다는 것을 알 수 있었다.

또한 F. Macdonald[8]은 현존하는 인스펙션 메소드를 조사하고 이 메소드들을 기술하는데 사용될 수 있는 포괄적인 인스펙션 프로세스를 제안한다. 이 프로세스에서 인스펙션은 전체범위를 수행하기 위해 단계적 정제를 사용한다. 인스펙션 범위를 N-Fold로 나누고, 단계적으로 인스펙션을 수행함으로써 인스펙션이 좀 더 체계적으로 수행될 수 있음을 보인다. 인스펙션은 전체범위를 대상으로 수행하기 보다는 단계를 나눠서 수행하는 것이 좀 더 효율적인 결과를 얻을 수 있음을 확인할 수 있었다.

기존 e-뱅킹 시스템의 품질에 대한 연구로서, Jayawardhena와 Chanaka[9]는 서비스 품질 측정을 위해 접근성, 웹사이트 인터페이스, 관심도, 신뢰성과 같은 부문에 21개의 척도를 개발하고 이를 이용하여 e-뱅킹의 구축이 완료되고 서비스 수행 중에 서비스 품질을 측정하는 기법을 제안하였다. 이와 같이 기존의 e-뱅킹 시스템에 대한 연구들은 금융 도메인의 특성을 확인하여 시스템 개발이 완료된 후 테스팅을 수행하는 기법을 제안함으로써 e-뱅킹 시스템의 완성도에 관심을 집중한다.

이와 같이 효율적인 코드 인스펙션을 수행하기 위한 연구는 지속적으로 수행되고 있으나, e-뱅킹 시스템의 특성을 고려하여 효율적인 코드 인스펙션을 수행하기 위한 연구는 부족하다.

3. 효율적인 인스펙션 범위의 우선순위 설계

효율적인 코드 인스펙션 범위의 우선순위를 설계하기 위해 먼저 사례가 되는 A은행의 e-뱅킹 시스템 구축 사례를 확인하고, 이 사례에서 개발사와 발주사의 인터뷰를 통해 우선순위 판단 기준을 식별한다. 그리고 이 판단 기준들의

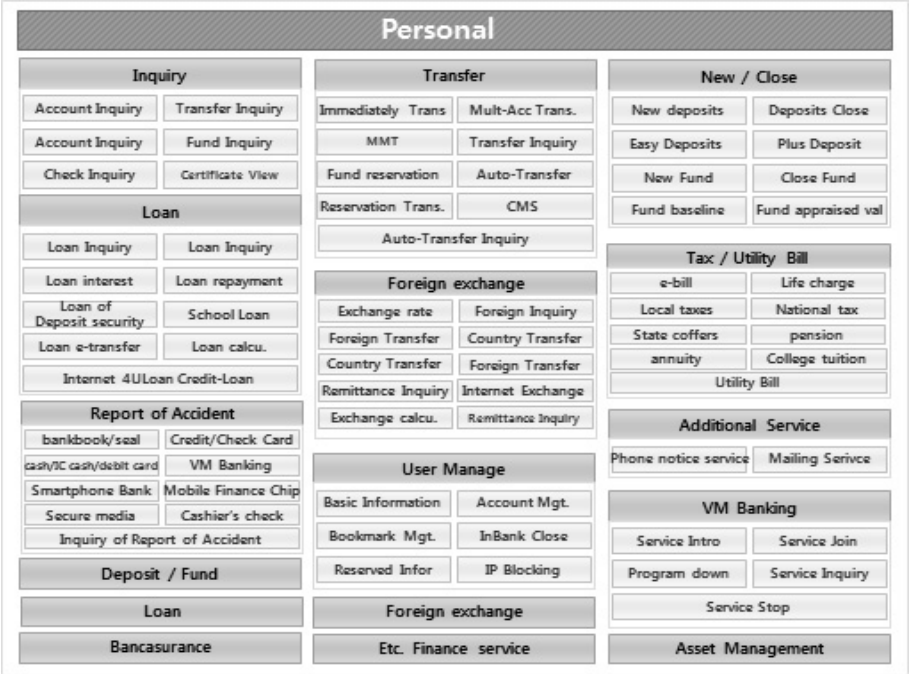

Fig. 3. e-Banking System IA in A Banking

우선순위를 객관적으로 비교하기 위해 AHP(Analytic Hierarchy Process) 기법을 이용한다. 마지막으로 설계된 우선순위 판단 기준을 통해 e-뱅킹 시스템의 코드 중 코드 인스펙션을 우선적으로 수행하여야 하는 인스펙션 범위를 식별한다.

3.1 사례1: A은행 e-뱅킹 시스템 구축

A은행은 e-뱅킹 인터넷 뱅킹 시스템을 재구축하려고 한다. 이 프로젝트의 개발기간은 M+6개월이며, A은행은 M+4가 경과한 시점에서 현재 개발 진행 상황과 품질 확인을 위한 중간보고를 요구하였다.

A은행이 프로젝트에서 수행하는 전체 개발 범위 중 개인뱅킹에 해당하는 정보 아키텍처(Information Architecture)는 Fig. 3과 같다.

A은행의 e-뱅킹 시스템은 처음 개발 제안 시 3단계 깊이 기준으로 총 263개의 메뉴로 구성되어 있었으며, A은행와의 협의에 따라 총 개수가 지속적으로 변화하고 있다.

이 중 인스펙션의 대상이 되는 기 개발된 3단계 깊이 기준 총 개수는 311개이다. 이에 대한 품질을 보증하기 위해서 B개발사의 QA팀은 품질보증 프로세스에 의한 테스팅을 지속적으로 수행하고 있으며, A은행 요구에 따른 추가적인 인스펙션은 개발 기간을 지연시킬 수 있기 때문에 문제를 제기하였다. 이에 따라 A은행 담당 PM과 B개발사 개발 PM은 모두 효율적인 인스펙션의 범위 설정하기 위한 우선순위의 필요성에 대해 인지하였다.

3.2 인스펙션 범위 우선순위 판단을 위한 기준 식별

효율적인 인스펙션 범위 설계를 위해 e-금융 시스템 구축을 발주하는 A은행 담당자 5명과 차세대 e-뱅킹 시스템 구축 경험을 보유하고 e-뱅킹 시스템 구축에서 5번 이상의 PM 경험을 보유한 B 개발사의 5명의 개발자를 대상으로 인터뷰를 수행하였다. 이를 통해 코드 인스펙션이 필요한 범위를 구분하기 위해 공통적으로 이용할 수 있는 척도를 확인하였다. 그 결과 효율적인 코드 인스펙션을 수행하기 위한 범위는 코드 사용빈도, 복잡도, 크기, 전문연계 개수, 사용 함수 개수, 영향력, 외부 연계 개수, 결함발견율을 기준으로 확인되어야 함을 식별하였다. 식별된 척도들은 A은행의 e-뱅킹 시스템에서뿐만 아니라 e-뱅킹 시스템의 인스펙션 시 공통적으로 사용될 수 있도록 선정되었다.

코드 사용빈도는 해당 코드가 다른 코드들에서 재사용되는 빈도가 빈번해짐에 따라 전체 프로덕트의 품질에 더 큰 위협을 가할 수 있기 때문에 선정되었다. 또한 복잡도는 복잡도가 증가할수록 결함을 가질 확률이 증가하기 때문에 선정되었다. 그리고 크기는 코드의 크기가 클수록 범위 각각의 품질 측정이 더 어렵고, 품질 측정에 소홀할 수 있기 때문에 선정되었다. 또 전문연계 개수는 e-뱅킹 시스템의 특성상 전문이 연계될 때 데이터 이동이 빈번해지며, 실제 전문으로 소프트웨어의 품질을 측정하기 어려운 문제가 발생할 수 있기 때문에 선정되었다. 또한 사용 함수 개수는 사용되는 함수가 많아질수록 개발자가 고려하여야 하는 이슈가 많아지며, 함수를 통해 다른 코드와 연계됨으로써 좀 더 철저히 코드 인스펙션이 필요하기 때문에 선정되었다. 그리고 영향력은 해당범위가 서비스의 품질에 영향을 미치는 정도를 의미하며, 해당 범위의 개발 완료 시 해당 범위가 빈번하게 사용됨에 따라 결함이 존재할 때 더 큰 영향을 끼칠 수 있기 때문에 선정되었다. 또 외부 연계 개수는 해당 범

Table 1. Pairwise Comparison of each Metric

	Frequently used code	complexity	size	number of telegram linkage	number of used functions	influence	number of external linkage	fault detection rate
Frequently used code	1	3	3	2	3	1	2	3
complexity	1/3	1	2	1/2	1	1/3	1/3	2
size	1/3	1/2	1	1/3	1/2	1/3	1/3	1/3
number of telegram linkage	1/2	2	3	1	1	1/3	1	1/2
number of used functions	1/3	1	2	1	1	1/3	1/3	2
influence	1	3	3	3	3	1	3	3
number of external linkage	1/2	3	3	1	3	1/3	1	3
fault detection rate	1/3	1/2	3	2	1/2	1/3	1/3	1
Weight (W)	0.222	0.078	0.046	0.099	0.085	0.246	0.149	0.075

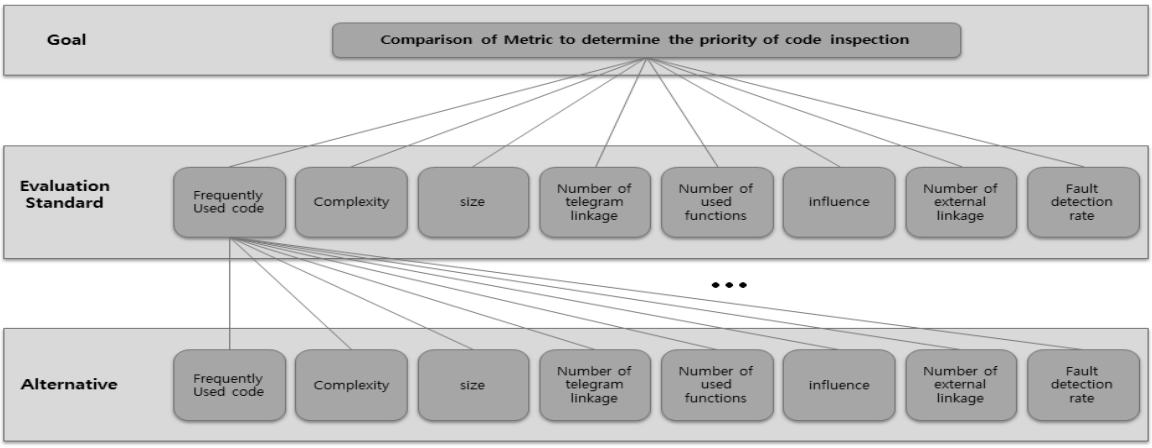

Fig. 4. Hierarchical layer for AHP analysis

위가 외부와 연계되는 정도에 따라 좀 더 철저한 품질 확인이 지속적으로 필요하기 때문에 선정되었다. 마지막으로 결함 발견율은 결함의 발견 정도에 따라 지속적으로 결함이 발생하기 쉽고, 해당 범위에 대한 재확인이 필요하기 때문에 선정되었다.

3.3 AHP를 통한 판단 기준 비교

코드 인스펙션 범위의 우선순위를 판단하기 위한 기준을 객관적으로 도출하기 위해, 인터뷰를 통해 수집된 판단 기준을 AHP기법을 이용하여 비교하였다. AHP는 다기준 의사결정 문제에 대안을 선택하기 위한 기법으로써 AHP의 상대비중을 통해 객관적으로 코드 인스펙션 범위를 판단하기 위한 우선순위 기준을 선택한다. AHP는 두뇌의 단계적 또는 위계적 분석과정을 활용한다는 사실에 착안하여 Thomas L. Saaty가 고안한 계산모델로서, 의사결정의 전 과정을 다단계로 나눈 후 이를 단계별로 분석 해결함으로써 최종 의사결정에 이르는 방법이다[10]. 이를 위한 계층적 구조를 Fig. 4와 같이 정의하였다.

이 계층적 구조를 통해 인터뷰로 도출한 효율적인 코드 인스펙션 범위의 판단 기준을 쌍대비교 함으로써, e-뱅킹 시스템에서 우선적으로 고려하여야 하는 코드 인스펙션 범위를 위한 판단 기준의 우선순위를 결정한다. Table 1은 쌍대비교를 수행하고, 이를 통해 각 척도들에 대한 가중치를 산정한 결과를 보인다.

가중치는 각 항목의 모든 쌍을 쌍대비교하여 Table 1과 같이 행렬로 나타내고, Equation (1)을 이용하여 각 값들의 기하평균(A)을 산출하며, Equation (2)을 이용하여 산출한 기하평균 전체의 합에 각 항목의 기하평균 값을 나누어 각각의 가중치(W)를 산출한다.

$$A = \left(\prod_{i=1}^{n} a_i\right)^{\frac{1}{n}} = (a_1 \cdot a_2 \cdots a_n)^{\frac{1}{n}} \quad (1)$$

$$W_i = \frac{A_i}{\sum_{k=1}^{n} A_k} \quad (2)$$

Table 1에서 보이는 것처럼, 영향력, 코드 사용빈도, 외부연계 개수가 가장 가중치가 높은 것으로 조사되었다. 이에 따라 영향력, 코드 사용빈도, 외부연계 개수를 확인함으로써

효율적인 코드 인스펙션의 범위를 설계할 수 있음을 확인하였다.

특히 쌍대비교시 일치율은 0.06으로써 0.1이내이기 때문에 수집된 수치가 일관성있게 수집되었음을 확인할 수 있다. AHP는 일치율이 0.1 이내일 경우는 일반적으로 받아들여지지만 0.2 이상일 경우는 자료입력을 다시 할 것을 권하고 있다[11].

3.4 설계된 판단 기준을 통한 범위 우선순위 식별

이절에서는 3.2절에서 식별된 영향력, 코드 사용빈도, 외부연계 개수를 코드 범위를 비교함으로써 효율적인 코드 인스펙션을 수행하기 위한 인스펙션 범위 우선순위를 식별한다.

1) 영향력

코드 인스펙션 범위를 판단하기 위해 가장 우선적으로 판단되어야 하는 척도인 영향력을 측정하기 위해서는 프로덕트가 운용될 때 해당 범위가 사용되는 정도를 확인한다. 이를 확인하기 위해 기존에 e-금융에서 고객이 자주 이용하고 가장 많은 거래가 발생하는 논리적 범위를 고려하였다. 가장 많은 거래가 발생하기 때문에 e-뱅킹 시스템에서 가장 안정적인 부분이어야 한다. 또한 가장 많은 거래가 발생하기 때문에 은행의 담당자도 중요하게 생각하는 범위가 된다. 따라서 A은행과 B개발사가 동의할 수 있는 핵심적인 범위로 설정할 수 있다. 이 거래 빈도는 기존에 운영되는 부분이 아니기 때문에 시중은행 C에 사례를 이용하였다. 시중은행 C의 1달 평균 거래건수는 Table 2와 같다.

Table 2. one month average number of transactions in C Bank

Rank	Number of Transactions	Content of Transactions
1	42217661	Transactions Inquiry
2	27300605	Log-in
3	16185620	All Account Inquiry
4	7909181	Another Bank Account Inquiry
5	7900513	Security Validation
6	6405416	Received Transfers Inquiry
7	5927300	Withdraw Amount of Available Inquiry
8	4046275	Balance Inquiry
9	2142990	Salary Transfer Execution
10	1933459	3 Months Average Balance Inquiry
…	…	…

이 중 보안검증은 개발범위가 아닌 솔루션 연계범위이기 때문에 핵심범위에서 제외한다. 또한 거래내용은 개발 범위에 따라 Fig. 5와 같은 상관관계를 가진다.

영향력에 따라 식별된 인스펙션을 위한 논리적 범위를 이체(개인/기업), 대량거래, 전계좌조회, 로그인, 거래내역조회로 구분할 수 있다. 이는 개발된 코드의 기능적 분류에 의해 분류되었다. D금융사의 경우에도 이체, 전계좌조회, 거래내역조회가 전체 거래의 85%를 차지한다는 통계가 있으며, 로그인은 거래가 수행되기 전에 우선적으로 수행되고, 대량거래는 사고 발생 시 위험성이 높기 때문에, 설계된 인스펙션 범위에 기반이 된 C금융사의 거래 내역이 핵심 범위로 선정하기에 유의 수준이라고 판단할 수 있다.

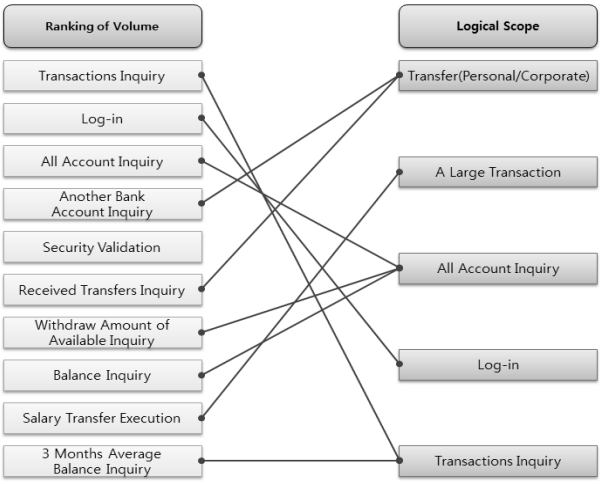

Fig. 5. Correlation between Ranking of Volume and Logical Scope

2) 코드 사용빈도

각 척도에 대한 객관적인 수행을 위해 코드 인스펙션 툴인 CodePro AnalytiX를 이용한다. CodePro AnalytiX는 구글에서 제공하는 무료 툴로서 여러 가지 검사 규칙을 제공한다[12]. 금융사의 요구에 따른 효율적인 인스펙션 수행을 위해 상용 툴을 사용하는 것은 프로젝트의 비용을 증가시키기 때문에 오픈소스 툴을 사용함으로써 객관성과 동시에 경제성을 얻을 수 있다.

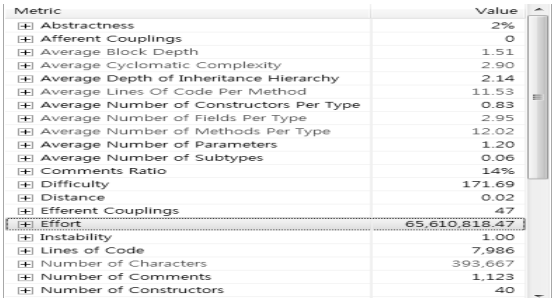

Fig. 6. CodePro AnalytiX Run Screen

코드 인스펙션 범위를 판단하기 위해 다음으로 판단되어야 하는 척도인 코드 사용빈도를 측정하기 위해 각 범위들과 다른 범위의 코드 유사도를 확인하였다. 코드 유사도는 코드 인스펙션 툴의 “Similar Code”기능을 이용하였다. 이에 따라 상위에 랭크된 해당 범위는 Fig. 7과 같다.

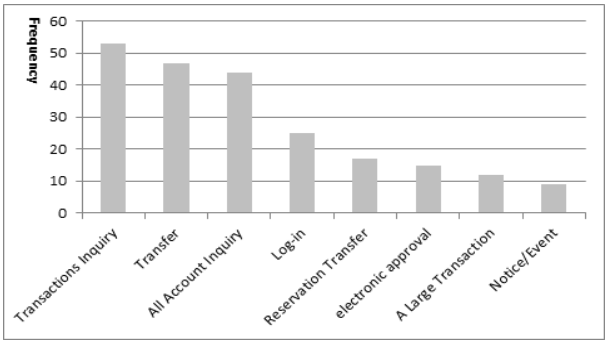

Fig. 7. Frequently uesed code in appled scope

코드 사용빈도에 따라 식별된 효율적인 인스펙션을 위한 범위는 전체 범위에서 53번 재사용된 계좌내역 조회, 47번 재사용된 이체를 비롯하여 전계좌조회, 로그인, 예약 이체, 전자결재, 대량 이체, 공지사항/이벤트 등으로 식별할 수 있었다.

3) 외부연계 개수

코드 인스펙션 범위를 판단하기 위한 마지막 척도인 외부연계 개수는 해당 범위의 커플링(Couplings) 정도를 이용함으로써 확인하였다. 이에 따라 상위에 랭크된 해당 범위는 Fig. 8과 같다.

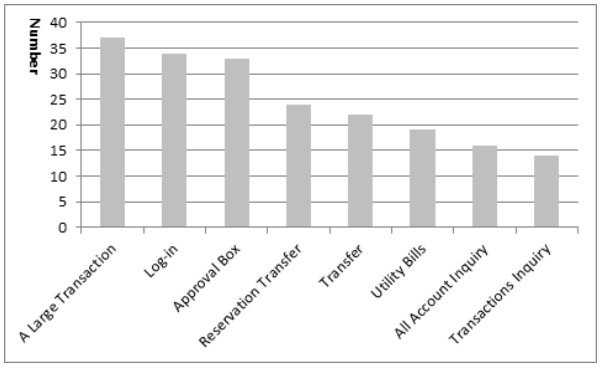

Fig. 8. Number of external link in appled scope

외부연계 개수에 따라 식별된 범위는 총 37개의 연계를 가진 대량 거래, 34개의 연계를 가지는 로그인을 포함하여 결재함, 예약 이체, 이체, 지로, 전계좌조회, 거래내역 조회 등으로 식별할 수 있었다.

4) 효율적인 코드 인스펙션 핵심 범위의 결정

핵심 범위를 판단하기 위한 척도인 영향력, 코드 사용빈도, 외부연계 개수를 적용하여 공통적으로 추출된 효율적인 코드 인스펙션의 범위는 이체, 대량거래, 전계좌조회, 로그인, 거래내역조회이다.

이에 따라 인스펙션을 위한 효율적인 핵심 범위를 이체(개인/기업), 대량거래, 전계좌조회, 로그인, 거래내역조회로 설계하였다. 이는 개발된 코드의 기능적 분류에 의해 분류되었다. 이와 같은 범위는 타은행에서의 e-뱅킹 시스템 개발 시에도 이 범위가 가지는 중요성 때문에 개발 초기에 우선적으로 개발되며, 또한 이를 토대로 다른 코드들에서 재사용되기 때문에 중요하다. 하지만 개발 초기 시에도 각 시스템간의 연계나 각 시스템의 프레임워크가 상이하기 때문에 주의를 기울이는 부분이기도 하다. 따라서 인스펙션의 우선순위가 되는 부분이 된다. 이와 같은 범위를 식별함으로써 코드 인스펙션을 수행하는데 우선적으로 품질 측정을 수행하여야 하는 범위를 식별하였다.

4. 기존 범위와 설계된 인스펙션 범위의 비교

4.1 인스펙션 규모

인스펙션의 효율성을 확인하기 위해서 먼저 품질측정의 대상이 되는 인스펙션의 규모를 기준으로 기존의 전체 범위와 설계된 범위를 CodePro AnalytiX의 “Line of Code” 척도를 통해 비교하였다.

CodePro AnalytiX를 통해 측정된 Line of Code의 수는 Fig. 9과 같이 전체범위가 721,152 LOC이며, 설계된 범위가 30,326 LOC로써, 설계된 범위는 전체 범위의 4.205%의 규모로 측정되었다. 이 결과는 코드 인스펙션을 수행하여야 하는 코드의 범위가 전체범위인 721,152 LOC에 비해 설계된 범위인 30,326 LOC로 약 1/24배 정도의 규모로써, 이는 코드 인스펙션에 수행하는데 소요되는 노력이 그만큼 감소될 수 있음을 의미한다.

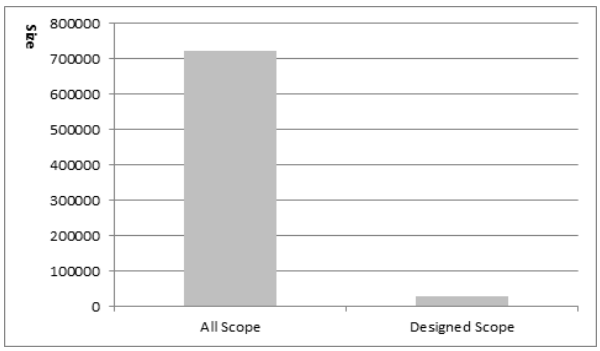

Fig. 9. Size Comparison of All Scope and Designed Scope

4.2 결함율

결함율은 프로덕트 전체 크기에서 발견된 결함의 수의 비율로 정의할 수 있으며[13], 객관적으로 결함율을 판단하기 위해 CodePro AnalytiX Tool을 이용한다. CodePro AnalytiX는 코드의 결함을 발견하기 위해 코드의 결함을 3가지 수준(High, Medium, Low)로 판단하여 표기한다. High 수준의 결함은 소프트웨어에 치명적인 오류로서 시스템의 동작을 방해하는 수준의 오류를 확인한다. 또 Medium 수준의 오류는 예외 처리 미비, String 오류, 불필요한 Import의 사용과 같은 코딩 스타일 오류를 확인한다. 마지막으로 Low 수준의 결함은 웹표준 규약 위반과 같은 결함으로 동

작에는 문제가 되지 않으나, 표준 규약 준수에 문제가 되는 결함들을 확인한다.

전체 범위와 설계된 범위의 결함율을 비교해 보면, Fig. 10과 같이 전체 범위에서 High 수준의 결함은 0, Medium 수준의 결함은 7354개(1.019%), Low 수준 결함은 2876개(0.398%)가 발견되었으며, 설계된 범위에서는 High 수준의 결함은 0, Medium 수준의 결함은 305개(1.005%), Low 수준 결함은 89개(0.293%)가 발견되었다. 이는 대부분 e-뱅킹 시스템의 특성(전문을 통한 커뮤니케이션, 보안 모듈 이용으로 인한 웹표준 위반 등)상 나타난 결함으로써 시스템 운영에 문제를 야기하지 않는 수준의 품질 수준이다. 이와 같은 결과는 각 코드가 지속적으로 단위 테스팅을 통해 대부분의 결함이 제거되었기 때문인 것으로 보인다.

여기서 결함율을 비교해 보면, 전체 범위 코드 인스펙션과 설계된 범위 코드 인스펙션이 유사한 결함율을 보이고 있으며, 설계된 범위의 인스펙션을 수행함에도 전체 범위와 유사한 수준의 품질 측정이 가능함을 보인다. 이는 e-뱅킹 시스템의 특성상 유사한 코드의 재사용이 빈번하며, 설계된 범위의 소스 코드가 시스템 전반에서 빈번하게 재사용되기 때문이다.

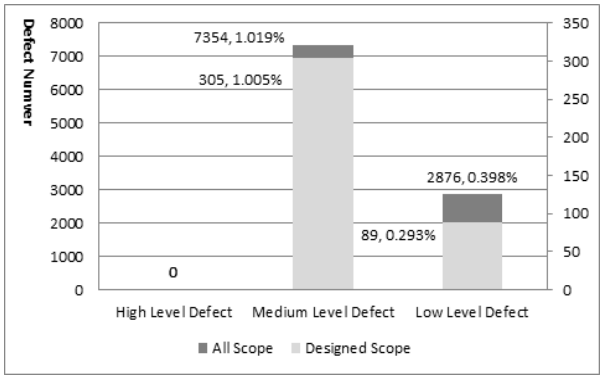

Fig. 10. Defect rate Comparison of All Scope and Designed Scope

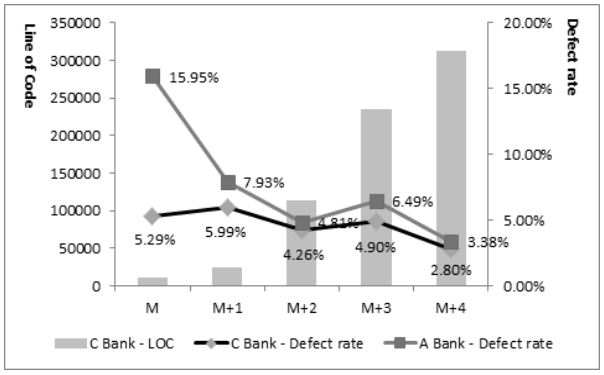

Fig. 11. result of C Bank re-building project

5. 사례2: C은행 e-뱅킹 시스템 재구축

설계된 범위인 이체(개인/기업), 대량거래, 전계좌조회, 로그인, 거래내역조회를 고려하여, C은행 e-뱅킹 시스템 재구축 프로젝트에서는 해당 범위를 집중하여 품질 측정을 수행하였다. C은행 e-뱅킹 시스템 재구축 프로젝트를 M+4까지 수행한 결과는 Fig. 11과 같다.

C은행 e-뱅킹 재구축 프로젝트의 수행결과에서 보이는 것처럼, A은행 e-뱅킹 구축 프로젝트의 결함율과 비교하여 설계된 범위에 코드 인스펙션을 집중적으로 수행하였을 때 전체 LOC 대비 결함율이 전체적으로 감소하는 것을 볼 수 있다. 특히 M기간에서 분석/설계와 병렬적으로 해당 범위의 코드 개발을 우선적으로 수행하고, 지속적으로 품질 측정을 수행함으로써 전체적으로 고품질의 개발을 수행할 수 있음을 확인하였다.

6. 결 론

본 논문에서는 e-뱅킹 시스템의 특성을 고려하여 범위를 판단하기 위한 우선순위를 설계하고, 우선순위를 통해 효율적인 코드 인스펙션을 수행할 수 있는 범위를 제안하였다. 그리고 이 범위를 통해 코드 인스펙션을 수행하였을 때 전체 범위를 코드 인스펙션하였을 때 보다 효율적인 코드 인스펙션을 수행할 수 있음을 확인하였다. 또한 해당 범위를 집중적으로 인스펙션하여 프로덕트를 개발하였을 때 좀 더 고품질의 프로덕트를 개발할 수 있음을 확인하였다.

물론 금융사 프로젝트를 수행함에 있어서 전체 범위에 대해 코드 인스펙션을 수행하는 것이 더욱 정확한 품질 측정을 수행할 수 있음은 분명하다. 하지만 프로젝트 수행 중 개략적인 품질 측정을 수행하기 위해서는 투입되는 노력적인 면에서 효율적인 코드 인스펙션이 필요하다. 따라서 설계된 범위를 통해 효율적인 코드 인스펙션을 수행할 수 있을 것으로 기대된다.

향후 e-뱅킹 시스템의 특성을 고려하여 코드 인스펙션을 더 효율적으로 수행하기 위한 핵심 품질 척도를 개발함으로써 제안된 코드 인스펙션 범위와 함께 e-뱅킹 시스템 구축에 있어서 효율적인 코드 인스펙션을 수행할 수 있는 체계를 구축하려는 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] Haeyoon Park, “Design of Zachman Framework for a Financial System”, KIISE KCC 2009 Vol.36, No.2(B), pp.77-81, 2009.

[2] SoonKi Kweon, “Business Architecture for IT Integration in the Financial Convergence Era”, 2005 Active Innovation Leader, pp.113-125, 2005.

[3] NIPA, “Software Engineering White Book: Korea 2012”, pp.171-172, 2012.

[4] Stephen R. Schach, “Object-Oriented and Classical Software Engineering”, Mcgraw Hill International Eight Edition, pp.154-182, 2011.

[5] Roger S. Pressman, “Software engineering - A Practitioner's

Approach", Mcgraw Hill International Edition, pp.163-193.
[6] Rico, D. F., "How to estimate ROI for inspections, PSP, TSP, SW-CMM, ISO 9000, and CMMI.", The DoD Softwaretech News, Vol.5, No.4, pp.23-31, 2002.
[7] Laitenberger, O., "Studying the effects of code inspection and structural testing on software quality", Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on., pp.237-246, 1998.
[8] Macdonald, F., and J. Miller. "Modelling software inspection methods for the application of tool support." IEEE Annual Symposium on Foundations of Computer Science, No.16, 1995.
[9] Jayawardhena, Chanaka. "Measurement of service quality in internet banking: the development of an instrument." Journal of Marketing Management 20.1-2, pp.185-207, 2004.
[10] Thomas L. Saaty, "How to make a decision: The analytic hierarchy process", European Journal of Operational Research, Vol.48, Issue 1, pp.9-26, 1990.
[11] Minsuk Yoon, Seungbong Park, "An Empirical Study on the Quality Requirements Priorities of Packaged Software Using the AHP", the study of internet electronic commerce, Vol.8, No.2, 2008.
[12] Plösch R., Mayr A., Pomberger G., Saft M., "An Approach for a Method and a Tool Supporting the Evaluation of the Quality of Static Code Analysis Tools", Proceedings of SQMB 2009 Workshop, held in conjunction with SE 2009 conference, March 3rd 2009, Kaiserslautern, Germany, published as Technical Report TUM-I0917 of the Technical University Munich, July, 2009.
[13] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Software quality: the elusive target [special issues section]." Software, IEEE 13.1, pp.12-21, 1996.

박 해 윤
e-mail : park8312@dankook.ac.kr
2009년 단국대학교 정보컴퓨터학과(학사)
2010년 단국대학교 컴퓨터학과(석사)
2010년～현 재 단국대학교 컴퓨터학과 박사과정
관심분야 : 소프트웨어공학, 웹공학, 테스팅

유 해 영
e-mail : yoohy@dankook.ac.kr
1979년 단국대학교 수학과(학사)
1981년 단국대학교 수학과(석사)
1983년～현 재 단국대학교 컴퓨터학과 교수
관심분야 : 소프트웨어공학, 웹공학