시험 가이드라인을 결정하기 위한 정량적인 결함 분석 사례 연구

Quantitative Analysis of Development Defects to Guide Testing

이재기(J.K. Lee) 네트워크시험팀 책임연구원
신상권(S.K. Shin) 네트워크시험팀 선임연구원
남상식(S.S. Nam) 네트워크시험팀 책임연구원, 팀장
박권철(K.C. Park) 네트워크전략연구부 책임연구원, 부장

검출된 소프트웨어의 결함에 대한 분석은 소프트웨어의 품질을 향상시키기 위한 여러 활동에 많은 도움을 주고 있다. 특히 개발중인 소프트웨어 컴포넌트들에 대한 검출된 결함 분석은 개발기간에 소프트웨어 내에 숨어있는 결함(latent defect)에 초점이 맞추어져 시험에 많은 도움을 주고 있다. 본 논문은 대형 교환 소프트웨어로부터 시험에서 검출된 결함 데이터를 이용하여 소프트웨어의 특성을 조사, 분석하여 이를 시험에 활용하고 시험의 효율성과 시험효과에 대한 가이드 라인을 제안한다.

I. 서 론

시험의 효과와 효율성(test effect & test effectiveness)은 소프트웨어 프로젝트 수행에 많은 영향을 끼친다. 설계 및 구현 단계에서 업무의 질을 평가하고 시스템 개발주기의 조정(scheduling) 등 복잡한 문제에 많은 영향을 주는 것이 사실이다.

즉, 개발기간에 발견된 결함은 시험기간 동안 발생될 결함과 밀접한 관계가 있기 때문이다. 이런 관계를 정리해 보면 아래와 같다.

- 개발기간 중 발생된 에러나 시스템 문제는 시스템 시험시 완전히 수정되지 않으면 치명적인 위험을 일으킬 수 있다.
- 사소한 결함(minor defect)들은 장시간 해결되지 않는 결함으로 발전할 수 있다.
- 문제들은 더욱 전파될 수 있다(파생 에러 및 고장 전파).
- 파생 에러(side effect error)나 전파된 고장(propagation failure)들은 시험 지침(test guide)에 활용될 수 있다.

시스템 개발중에 검출된 결함은 초기에 시스템 시험에서 걸러져야 한다. 그렇지 않으면 개발자는 남아있는 문제들을 해결하는 데 많은 시간을 소비해야만 한다. 이와 관련하여 통계적인 stopping rule에 대한 연구결과가 [1],[2]에 의해 발표되기도 하였다. 이 연구에서는 몇 개의 요인에 의존한 방법으로 시험 활동(test activity)의 우선순위(priority) 등이 고려되지 못했다.

소프트웨어 컴포넌트들은 다른 컴포넌트들과 연동(interface)되어 특수한 순서(test scenario)에 의해 시험되는데 이것은 시스템 시험을 시작하기 전 각 phase에 대한 “qualification”에 의해 순서가 정해지게 된다. 그 외에 시험의 방법, 즉 노하우에 대

한 요소를 고려하여야 한다. 즉, 시스템 시험은 운용 상태 및 규격, 대상(feature), 컴포넌트에 대해서 시험이 수행되기 때문이다. 만약 고장 보고에 의해 분류된(identified) 결함이 있는 컴포넌트들에 대해서 우선순위가 고려된 시험 활동이 뒤따르게 된다. 결국 소프트웨어에 대한 시험순위 결정은 소프트웨어 개발 및 시험의 확대에 가속화를 가져오게 된다. 이러한 절차는 water-fall 모델의 절차와 위배되는데 시스템 개발에서는 절차에 유연성을 가져야 효과적인 시스템 개발이 이루어 질 수 있다.

시험자의 중요 관심사는 post release defect에 대한 정도로 소프트웨어 배포 후 나타나게 될 소프트웨어 컴포넌트 속에 잠재하고 있는 결함의 형태이다.

본 논문에서는 Ohlsson[3]이 제안한 소프트웨어 컴포넌트 fault prone identify 기법을 사용하여 대형 ATM 교환시스템의 소프트웨어에 대한 개발기간 및 시험기간에 대해 다각도로 분석, 예측하고 시험의 효과 및 질에 대한 평가를 수행한다. 또 소프트웨어 배포 이후에 대한 현상도 분석하여 프로젝트 수행 결과를 종합 평가해 본다.

논문의 구성은 II장에서 시험의 효과를 향상시키기 위한 정보로서 소프트웨어 컴포넌트들에 대한 fault prone identify 기법들에 대해 살펴보고 III, IV장에서는 이에 대한 활용방법과 시험의 효과에 대한 평가, 시험의 가이드라인 등을 제안하고 마지막 결론에 향후 고려사항 등을 제안하고 맺는다.

II. 연구 배경

Frankl et al.[4]은 시험의 목적으로 2가지를 꼽는데 첫째는 결함을 제거하는 것이고 두번째는 신뢰할 만한 소프트웨어를 얻기 위한 품질 확인 계수이다. 이 목적에 의해 다양한 시험전략이 사용되는데 품질 확인을 위한 디버깅 시험의 품질 획득 및 운용시험이다. 디버깅 시험의 초점은 많은 발생 빈도가 높은 에러를 검출하는 것이기 때문에 그다지 중요하지는 않다. 반면에 운용시험은 현장에서 발생될 유사한 에러를 찾는 목적이며, 현장에서 사용될 비율에 의한 확률로 운용프로파일(operation profile)이 적용된 개념으로 매우 중요하다. 이 연구에서 알 수 있는 것은 품질을 향상시키기 위한 시험시 시험자의 안목있는 직관과 시험전략에 대한 통찰력을 겸비해야 한다는 것이다.

시험자의 소프트웨어에 대한 인식은 컴포넌트의 fault prone에 의해 이러한 컴포넌트에 대해서 시험전략을 집중시킬 수 있기 때문이다. 소프트웨어 컴포넌트 결함 분석의 대다수 연구결과들은 컴포넌트의 fault prone 분류에 초점이 맞추어지고 있다. 이러한 연구결과로는 [5]-[7] 등이 있으며, 컴포넌트에 잠재하고 있는 결함 수 예측 방법은 [8] 등이 있고, release를 통한 fault prone 컴포넌트들에 대한 추적방법은 [9]에 의해 제안되었다. 그밖에 [3] 등에 의해 제안된 컴포넌트 간의 관계에 의한 컴포넌트의 결합된 형태의 예측 방법이 있다. 특히 [10]은 각각의 컴포넌트들에 대한 결함 수를 순위로 정해 “yellow” 또는 “red”로 분류하고 있다. Red 컴포넌트들은 대부분 fault prone 이다.

[8]은 모듈 당 결함 수의 관계를 도출하여 운용현장에서 잔존 결함을 예측하기 위해 개발기간에 발견한 결함들을 이용하고 있다. 즉, 개발기간이나 현장에서 검출된 결함 데이터와 발생빈도는 상대적인 소프트웨어의 품질을 확인하는 측정에 좋은 본보기가 된다.

[11]은 시험에 대한 소프트웨어 엔지니어링 측면의 가정으로 초기 결함 데이터를 사용하여 향후 결함데이터를 예측할 수 있는 방법으로 일부 모듈에서 발견된 에러의 대부분을 포함하고 있다는 pareto 원리를 증명하고 있다. 배포 후에 대부분의 fault prone 모듈들은 최소한 배포 전에 결함을 줄이도록 노력해야 하며, 또 수정되어야 한다. 경우에 따라서는 문제점이 노출된 모듈이 결함의 양산 장소로 변할 수 있기 때문이다.

[3]과 다른 방법으로 우리가 시도한 방법은 배포뿐만 아니라 개발기간 및 시스템 시험기간에 대해서도 잘 정의된 fault prone component에 대한 적용방법을 시도하고 [11]이 제안한 정보를 활용하여

시험의 가이드라인을 삼아 결함이 적은 코드를 생산하도록 유도하였다.

[8]의 이론과는 달리 우리의 관심사는 개발기간, 1차 배포 및 후속 배포 사의의 결함 수 관계뿐만 아니라 시스템 시험을 향상시킬 수 있는 효율성과 효과에 대한 결과 분석이다. 이렇게 다양하고 예측적인 방법과 개선 활동을 지원하는 종합분석 방법을 채택하였다. 그 밖에 소프트웨어 품질확인 계수로서 C++ 프로그램의 컴포넌트에 대한 객체지향 설계 메트릭스 분석 결과도 발표되고 있다. 이 연구 결과는 대부분의 “traditional code metrics”가 시스템 개발 마지막 단계에서 수집된 데이터를 이용하여 개발 초기단계의 각 클래스에 대한 fault prone 예측 방법인 반면에 이 방법은 개발 전 주기에 대해 다양하게 이용할 수 있는 방법을 제시하고 있다[12].

III. 사례연구 및 적용 데이터

데이터 분석은 대형 ATM 교환시스템의 소프트웨어에 대한 결과로서 총 136개의 컴포넌트(여기서는 기능 블록을 하나의 컴포넌트로 가정함) 중 132개의 소프트웨어 컴포넌트를 대상으로 분석하였다. 하나의 컴포넌트는 개발 단위로 수 개에서 수십 개의 파일로 구성되어 있으며, 논리적 및 물리적인 결합관계를 유지하고 있다.

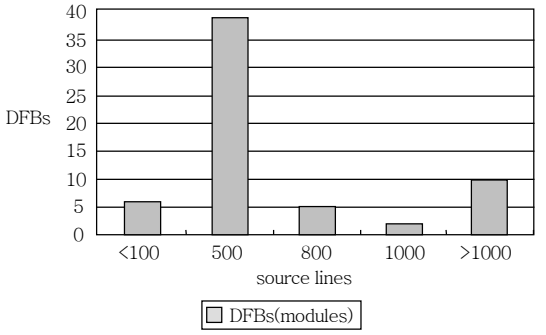
사례분석에 이용된 시험대상 소프트웨어 컴포넌트에 대한 규모는 <표 1>과 같다.

Ohlsson et al.[11] 등이 제시한 논문에서는 소프트웨어 복잡도와 사이즈와의 관계에 대한 충분한 증거를 제시하지 못했으며, 이 연구결과가 우리가 시도한 방법에는 적용이 안된다는 결론을 얻었다. 즉, 우리가 조사한 바로는 일부 모듈에서 대다수의 결함을 포함하지 않고 있으며, 결함 수의 다소(多少)에 차이는 있으나 대체로 대부분의 모듈에서 결함이 검출되고 있는 것으로 밝혀졌다. 이에 대한 상세 현황은 (그림 1)과 같다.

(그림 1)에서 보면 소스라인 규모별 결함 발생 모듈 수의 평균치 소스 라인 수가 100 라인 이상 500 라인 이하에서 가장 많이 발생하는 것으로 분석되었고 1,000 라인 이상이 두번째로 많은 것으로 나타나고, 100 라인 이하도 상당히 많이 검출되는 것으로 분석되었다. 또한 전체 소프트웨어 컴포넌트[여기서는 기능 블록(FB)으로 약칭함]의 규모(평균치)는 15,003 라인에 컴포넌트 당 결함 수는 평균 14.15개 정도로 분석되었다. 이것은 <표 1>에 언급된 3차 배포시 1,305 라인 당 1개의 결함이 검출된 것과 약간의 차이(평균 1,060 라인 당 1개의 결함 검출)를 보여주고 있다(Defect Function Blocks: DFBs).

<표 1> 전체 대상 소프트웨어 현황 및 결함

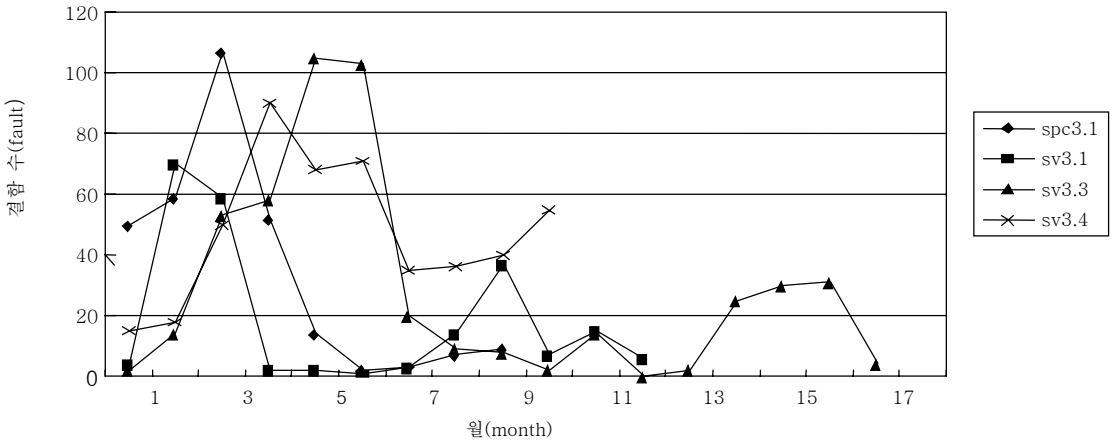
Project	Function	Block	SLOC
ACE	324	136	2,404KLOC
SLOC/defect(line)		전체	3차 배포
		1,060(line)	1,305(line)



(그림 1) 소스라인 규모 당 결함발생 모듈 수

1. 소프트웨어 버전과 기능과의 관계

적용된 소프트웨어는 5개 버전에 대해 3차례의 배포가 있었으며, SV3.1 버전은 영구가상회로연결(Permanent Virtual path Connection: PVC)에 의한 셀 전송기능 및 시스템의 기본적인 운용 및 유지보수 기능인 상태관리기능 위주로 실시되고 SV3.2 버전은 스위치에 의한 호처리(Switched Virtual path Connection: SVC) 기능을 SV3.3에서는 감사기능 및 망관리, 폭주제어 기능, 워크스테이션 포트 시험 기능 등 시스템의 M&A 기능이 강화되었고



(그림 2) 버전별/월별 결함 검출 수

SV3.4 버전에서는 PSTN 연동기능이 추가되었으며, 마지막 버전인 SV3.5는 동적 라우팅(dynamic routing) 기능을 수행하는 PNNI(Private Network-Network Interface) 연동기능 및 SVP(Switched Virtual Path) 호처리(자국호, 중계호) 기능 등이 추가 되어 시험되었다.

소프트웨어 컴포넌트의 추가는 각 버전 별로 SV3.2에 14개, SV3.3에 6개 블록, SV3.4는 43개 블록, SV3.5는 5개 블록이 추가되어 시험되었다.

2. 결함 데이터 관리 및 보고

개발자 및 시험자는 제출된 결함보고(혹은 fault report)에 의해 문제점을 분석하게 된다. 제출된 결함 보고는 소프트웨어 프로젝트 관리 툴로 관리되고 있으며, 문제점 발생 보고서에는 검출 일시, 문제점 유형, 발견자, 담당자, 관련 기능 블록 및 시험항목, 변경파일 목록, 상태처리 등을 기술하게 되어 프로젝트 참여자가 공유할 수 있도록 하고 있다. 툴에서 관리되고 있는 각 버전 별 결함 데이터 현황은 (그림 2)와 같다.

시험에서 검출된 결함 수는 버전별로 월 최고 110개부터 최소 5개 정도로 보고되고 있으며, SV3.4 버전은 소프트웨어 배포 단계에 이르러도 50여 개 정도가 잔존하고 있는 것으로 보고되고 있는데 이 이유는 앞에서도 언급한 것과 마찬가지로 PSTN 연동 기능이 추가되면서 많은 컴포넌트들이 일시에 추가되고 소스 규모(Source Line of Code: SLOC)도 크게 증가하여 소프트웨어 컴포넌트에 잔존결함이 많이 존재하고 있는 것으로 밝혀졌다.

IV. 접근 방법

많은 연구들이 실험적으로 시도되어 왔으나 현장에서 수집된 데이터를 사용하여 사례를 연구한 방법은 아래와 같은 단계로 수행되었다.

- 시스템 개발 및 시험기간 동안 fault prone predictor를 사용하여 컴포넌트의 fault prone identify를 결정한다.
- 예측이 포지티브 또는 네거티브 양상을 띨 때 이 예측에 개선을 줄 수 있는 결함 등을 결정
- Post release 문제에 대한 fault prone의 효과를 평가
- 2차 및 3차 배포 시 적용된 시험 가이드 라인의 요약
- 배포 후 상대적인 분석 수행

1. Fault Prone Component 분석

컴포넌트들은 시스템 개발기간이나 시스템 시험

기간 동안 발견된 결함 수를 가지고 순위를 정하는 방법들이 이용되어 왔다. [10]은 배포를 통해 각 컴포넌트에 잠재하고 있는 총 결함 수를 순위로 매기는 방법을 사용하는데 반해 우리가 사용한 방법은 개발기간과 시스템시험 기간 동안 검출된 결함 데이터를 이용하여 평가하는 방법이다. 즉, 단계를 세분하여 개발기간과 시험기간에 모두 임계치를 넘은 컴포넌트에 대해서는 “red”, 양쪽 모두 임계치 이내의 결함 수를 나타내면 “green”, 어느 한 쪽이 임계치를 넘으면 “yellow” 등으로 표시하여 fault prone component에 대한 평가를 하는 방법이다. 다시 말해서 fault prone component의 임계치(threshold value)를 정하는 방법은 아래와 같이 2가지 형태로 분류하여 수행하였다.

- 상위 25% 속하는 컴포넌트(비율에 의한 방법)
- 컴포넌트(기능블록) 별 총 결함 수

위의 2가지 방법을 이용하여 소프트웨어 컴포넌트에 대한 통계데이터 분석 결과는 <표 2>와 같다. <표 2>에서 3차 배포시 기능 개발기간 동안의 결함 데이터 통계는 비 적용하였는데 그 이유는 3차 배포시 중간에 시스템 개발 프로젝트에 포함되어 추진되던 PSTN 연동 기능이 패키지에서 제외되어 실제데이터 적용에 어려움이 있기 때문이다. 그래서 최종 현장 배포시 결함 수가 현저히 줄어드는 현상은 이 이유 때문이며 이때 40여 개의 관련 기능 블록이 제외되어 전체 소스 규모도 약 240만 라인 정도로 분석되었다.

PSTN 연동 기능이 포함된 전체 소프트웨어 소스 규모는 약 450만 라인 정도이다(단, 펌웨어소스는 제외됨). 데이터 분석 결과를 보면 소프트웨어 개발이 진행됨에 따라 특정 블록에서 결함이 많이 발생되고 있으며, 초기 시스템에서는 전체 소프트웨어 컴포넌트에서 발생되고 있는 결함이 골고루 분포하고 있음을 데이터 분산 데이터 값으로도 알 수 있다. 이 값은 점차 개발이 진행됨에 따라 분포도 값이 적어지는 경향을 띠며, 2차 배포 버전에서 다시 증가되는 추세를 보이는데 이는 PSTN 연동이라는 커다란 신규 기능이 수용됨에 따라 소프트웨어 컴포넌트의 증대 및 변경이 많았기 때문이다. 그러나 3차 배포 직후 현격히 줄어드는 현상은 앞에서 언급했듯이 PSTN(Public Switched Telephone Network) 연동 기능을 별도로 분리하여 개발을 추진하였기 때문에 컴포넌트 수의 감소 및 기능의 안정화 덕분에 줄어드는 것으로 분석되었다. 대체적으로 컴포넌트에서 발견되는 결함은 개발기간에 많이 발견, 수정되고 시스템 시험기간에서는 줄어드는 경향을 띤다. 이것은 시스템의 개발이 순조롭게 진행되고 있음을 단적으로 보여준다.

소스라인 당 결함 수(defects per SLOC)도 과거에 경험한(empirical data 적용) 프로젝트(결함 수: 1.2개/1K LOC)에 비해 개선된 것으로 추정되었다 [13],[14].

<표 3>은 1차 배포 버전에 대한 결함 발생 상위

<표 2> 개발기간 및 시스템 시험기간 동안 발견된 총 결함 수

	1차 배포(83 components)		2차 배포(103 components)		3차 배포(136 components)	
	개발	시험	개발	시험	개발	시험
총 결함 수	628	105	568	508	미적용	32
결함발견 컴포넌트 수	75	41	67	90		14
컴포넌트 당 평균 결함 수(25%)	17	8.5	16.895	13.130		4.5
표준편차	5.531	4.387	3.745	7.189		1.118
총 기능(function) 수 및 소스라인(SLOC) 당 결함 수	324(호처리: 165, 운용: 25, 보전: 64, TMN: 70) – 3차 배포 기준, 1305.907(= 1841/2,404, 175 SLOC)					
기능 당 결함 수(평균치)	5.682(= 1841/324)					

주) 컴포넌트는 기능 블록(Function Block: FB)을 의미함

10%(8개 블록)에 해당되는 소프트웨어 컴포넌트에 대한 결과 분석치로 8개 블록 중 개발기간에 fault prone 블록이 7개이고 하나의 블록이 시스템 시험에서 결함이 발견되지 않았다. 이 블록은 시스템의 모든 자원을 할당하고 제어하는 operating system에 대한 것으로 시스템 개발기간 중에 많은 결함을 수정하여 시스템 내에서 연동되어 동작되는 타 응용 프로그램의 안정적인 기반을 조성해준 결과이다. 특히, 시스템 개발시 기반이 되고 시스템 전체에 영향을 주는 OS 및 DBMS(Data Base Management System)의 기능 안정화가 매우 중요하다.

<표 3>의 chi-square 분석 결과 데이터가 적은 값으로 계산된 것은 관측치(observed frequency)와 기대치(expected frequency)간의 정확도가 높은 것을 의미한다. 즉, 혼합 메트릭스(mixed metrics)를 이용한 데이터 분석이 소프트웨어 컴포넌트에 대한 fault prone identify가 효과적임을 의미한다. chi-square test에 대한 결과는 (1)을 이용하여 계산할 수 있다.

<표 3> 혼합 메트릭스 데이터(1차 배포시 임계치상위 10%를 포함하는 컴포넌트)

임계치= 10%(8 components)	Prediction(system test)	
	Fault prone	Normal
Development fault prone	7	1
Development normal	41	33
Chi-square results	χ^2=1.816; p ≤ 0.001	

$$\chi^2 = \sum_{i=1}^{k} [(o_i - e_i)^2 / e_i] \qquad (1)$$

o_i : Observed frequency,

k : Check cells(수행 횟수),

e_i : Expected frequency

<표 4>의 소프트웨어 배포 기준별 컴포넌트 당 최대 검출 결함 수는 2차 배포의 시스템시험에서 발견된 33개로 시스템 개발 초기의 32개보다 많다. 즉, 소프트웨어 컴포넌트 수의 증가에 따른 결함이 급증한 예를 보여주며, 기존 PSTN 교환시스템에서 제공되는 일반적인 서비스를 ATM 교환시스템에서 다른 형태의 데이터 구조로 구현하여 시험한 경우로 경험 미숙에서 오는 시행착오 등이 결합되어 많은 에러가 유발되었다. 특히 순수 ATM 셀 교환(cell switching)을 위한 소프트웨어 규모에 상당하는 음성교환용 소프트웨어의 수용에 따른 파생에러가 많은 것으로 분석되었다.

<표 5>의 결함데이터에 대한 해결 난이도(by severity 1) 분석 결과는 대체로 개발기간에 발생한 컴포넌트들에 대해 시스템 시험에서도 결함이 50% 가까이 발생되고 있으며, 전체 시스템 개발 기간에 대해서는 17.5% 정도가 정상적으로 처리된 것으로 분석되었다. 이는 20% 미만의 컴포넌트에 대해서만 결함이 발견되지 않은 것으로 개발초기에 많은 문제가 발생되고 수정되고 있음을 알 수 있다. 즉, 초기 시스템의 개발이 한창 진행되고 있음을 의미한다.

(그림 3)은 결함 검출데이터에 대한 심각성 별, 버전 별 분석 결과이다. 결함의 심각성(혹은 심각도)은 영향도(影響度)에 의해 4단계로 사소한 결함(小),

<표 4> 임계치 값 크기 기준 Fault Prone 통계 데이터

임계치 크기(magnitude) 기준 * threshold: 1st & 2nd release(12개), 3rd release(5개)	1차 배포(83 component)		2차 배포(103 component)		3차 배포(136 component)
	개발	시험	개발	시험	시스템 시험
최대 결함 수(컴포넌트 당)	32	15	24	33	6
임계치 초과 컴포넌트 수	20	2	21	10	2
임계치 초과 컴포넌트 비율(%)	24.1	2.4	20.4	9.7	1.5
임계치 초과 컴포넌트의 평균 결함 수	17	13.5	16.5	19	5.5
임계치 초과 평균 분산(sigma)	8.337	1.5	3.905	6.74	0.5

보통(中), 주요(大), 심각한(파생적) 결함 순으로 구분된다.

심각성(도) 분석 결과에 따르면 사소한 컴포넌트의 변경(severity 1)에 대해서는 최소 42%~63%까지 차지하며, 점차 증가하다가 2차 배포 이후부터 감소하는 추세를 보인다. (그림 3)의 x축 값은 소프트웨어 버전을 표시하고, 개발기간(1,3)과 시스템 시험 기간(2,4,5)의 버전을 표시하며, 예를 들면 2번은 소프트웨어 1차 배포, 4번은 2차 배포, 5번은 3차 배포가 된다.

심각한 결함(severity 4)은 초기에 2% 정도를 차지하나 1차 배포 이후에는 거의 발생이 안되고 있으며, 결함의 심각성이 중간 정도인 severity 2가 마지막 배포단계에서는 사소한 결함보다 많아지는 경향을 보인다. 즉, 사용자 요구사항의 추가수용과 현장 운용시 문제점을 해결하기 위한 기능의 수정이 가해지면서 소프트웨어 컴포넌트에도 많은 변경이 가해지고 있음을 알 수 있다.

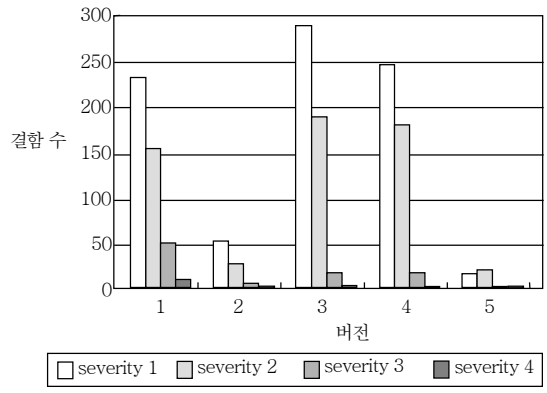
<표 6>의 내용은 1차 배포 후 발생되고 있는 결함에 대한 심각도(深刻度)를 분석한 결과로 주요결함(severity 3, 4)이 개발기간에 20% 이상 검출되어 수정되고 있음을 나타낸다. 즉, 초기 설계단계에 고려하지 못했던 사항들이 시험을 통해 많이 개선되고 있음을 보여주고 있다.

단순 결함인 severity 1의 임계치 값이 5이상 되는 fault prone component는 개발기간에서는 주로 스위치 장애관리, DB 처리, UNI/NNI 관리 기능, PVC 호처리 및 통계데이터 처리, 국 데이터 처리기능 등에 많은 결함을 가지고 있는 것으로 분석되었고 시스템시험 기간 동안은 주로 중계선 호 처리 및 루팅제어(routing control), Interface Module(IM) 및 링크 상태관리 등에 많은 에러가 발생되고 있다. 1차 배포까지 시스템 시험으로 발견된 결함은 <표 7>과 같이 분석되었다.

<표 8>의 분석 결과는 개발기간에 가입자 데이터 처리, 시스템 운영체제(OS) 및 점대점(point to point: p-t-p), 점대다중점(point to multipoint: p-t-mp) 셀 전송 기능에 오류가 많고 시스템 시험기간에는 특정 컴포넌트보다는 call control 및 운용기능에 전반적으로 분포되는 경향을 보였다. 이것은 문제점 해결에 가장 많이 발생되고 있는 보통 문제의 실례를 말해준다.

<표 5> 난이도에 따른 분석 현황(by Severity 1)

임계치 ≥ 10 defects	Prediction(system test)	
	Fault prone	Normal
Development fault prone	47	12
Development normal	24	3



(그림 3) 심각성 별 결함 데이터

<표 6> 1차 배포 후 Severity 별 통계 데이터

	Grade of severity			
	1	2	3	4
Development	54	41	21	6
System test	29	24	5	0

<표 7> Severity 1에 의한 1차 배포 분석 결과

임계치 ≥ 5 defects	Prediction(system test)	
	Fault prone	Normal
Development	15	6
System test	9	53

<표 8> Severity 2에 의한 1차 배포 분석 결과

임계치 ≥ 5 defects	Prediction(system test)	
	Fault prone	Normal
Development	9	5
System test	4	65

심각한 결함(severity 3)인 <표 9>의 데이터는 개발기간에 과금정보 수집 및 관리, 이중화 제어 등에 결함이 많이 분포하고 있으며, 시스템 시험기간은 자국호 제어 부분에 많은 결함이 검출되고 있다. 이것은 교환시스템의 핵심 기능인 호제어와 이에 따른 과금수록, 또 시스템의 주요부분에 장애 발생시 이에 대한 대처 능력을 보여주는 이중화 제어 부분으로 다양한 서비스 수용 및 새로운 표준(ATM-fourm) 규격을 수용하는 등 사용자 요구사항의 변화에 따른 영향으로 분석되었다.

시스템의 치명적인 결함(severity 4)인 <표 10>의 데이터는 OS 및 DB 관련 사항과 시스템의 주요 장애 발생시 호 제어의 restart 처리 기능 등으로 시스템 전체에 영향을 주거나 트랜잭션 데이터의 consistence 등이 요구되는 결함이다. 이런 결함들은 수정에 장시간이 소요되는 결함으로 시스템 전체에 발생되고 있는 결함의 약 2%를 차지하고 있다. 이 결함들은 시스템 개발초기에 발생하고 있다.

위와 같이 시스템 개발기간에 시스템 시험에서 검출된 결함데이터의 심각성에 대한 분석 결과 및 다각도로 소프트웨어를 분석한 사례[15]도 있었지만 좀더 다양한 방법으로 정량적인 분석방법을 도입하여 소프트웨어에 대한 체계적인 분석이 계속 수행되어야 한다. 다시 말해서 소프트웨어 복잡도에 대한 분석을 수행하여 효율적인 시스템 개발에 활용하여야 한다.

<표 9> Severity 3에 의한 1차 배포 분석 결과

임계치 ≥ 4 defects	Prediction(system test)	
	Fault prone	Normal
Development	3	2
System test	1	77

<표 10> Severity 4에 의한 1차 배포 분석 결과

임계치≥ 2 defects	Prediction(system test)	
	Fault prone	Normal
Development	2	4
System test	4	73

2. 2차 배포와 Post Release 버전과의 비교

2차 배포 버전의 소프트웨어는 현장에 배포된 바로 전의 버전으로 개발기간에는 PSTN 연동기능이 포함되어 많은 결함이 검출된 버전이다. 그러나 시스템 시험을 거쳐 배포된 버전은 이 기능이 제외된 소프트웨어로서 상당히 안정된 소프트웨어이다.

현장 배포기준으로 소프트웨어 컴포넌트에 잠재하고 있는 fault의 임계값(≥2)을 적용하여도 8개 정도의 컴포넌트에 대해서만 존재하는 것으로 분석되었다. 이 버전의 주요 문제점 발생 부분은 UNI/NNI 관련 라이브러리 처리부분과 망 관리(망 상태 제어) 및 폭주제어, 국 데이터처리 등 PSTN 연동 수용에 따른 관련 데이터의 변경 등이 많은 부분을 차지하고 있으며, 현장 배포 후 일시적인 bulk 트래픽 처리 및 망의 상태 관리, 예기치 않은 상황 발생에 따른 교환시스템의 적절한 운용 상태 유지 등에 필요한 기능이 수용되어 집중적으로 검증되었다.

Post release 버전은 중계선 호처리와 관련된 SVP(Switched Virtual Path) 호처리 기능과 PNNI 기능이 수용되어 현장에 서비스된 버전이다. <표 11>의 데이터를 살펴보면 2차 배포시 컴포넌트의 fault prone 상황이 마지막 배포 단계인 post release 단계보다 우수한 것으로 분석되는데 그 이유는 이 시기에 ATM-Fourm에서 권고하고 있는 기능(UNI 3.1/4.0/B-ICI)들이 새롭게 수용되며 기존에 수용되어 개발 완료된 ITU-T 권고안 기능(Q.2931/2971/B-ISUP)들과의 상호 연동 등으로 인해 결함이 상승하는 추세를 보였다. 이때 수정이 가해진 컴

<표 11> 2차 배포와 Post Release 비교 분석 결과

Component version	임계치 ≥ 2(pr) defects, 4(2nd st) defects	Prediction (system test)
2nd release	System test fault prone	3(new 6)
	System test normal	100(new 13)
Post release (field service start)	System test fault prone	12
	System test normal	124(new 5)

주) ( ) 안의 pr은 post release, 2nd st는 2차 배포 system test를 의미함

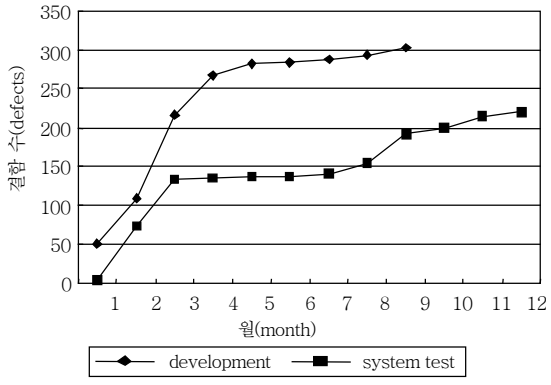
포넌트들은 호 제어 관련 기능들로 컴포넌트의 통합과 재사용(reuse)으로 인한 원인으로 밝혀졌다.

시험의 효과와 효율성을 판단하고 소프트웨어 컴포넌트 내의 결함들을 제거하기 위해 시도되는 시험에 대한 가이드 라인은 아래와 같다.

※ 효과적인 시험을 위한 가이드 라인

- 개발기간(development phase) 동안 fault prone component 들은 철저히 시험
- 신규 블록이나 통합된 블록은 더욱 철저히 시험(타 컴포넌트와 연동시 많은 에러를 발생할 확률이 높아 stub file 등을 이용, 사전시험을 통해 에러 발생 소지를 조치)
- 초기 시스템 개발기간에 결함이 대량 발생하는 컴포넌트는 가능한 한 시험을 중단(시험 효율성 문제)하고 문제점을 보완

(그림 4)의 결함데이터는 소프트웨어 1차 배포까지 개발기간과 시스템시험 기간 동안 검출된 결함에 대한 월별 누적 결함 수이다. 개발기간에 발견되는 결함 수는 시험 시작 후 6개월까지 지속적으로 많은 결함이 발견되고 있으며(초기 기능개발 확인시험에서 PVC 점대점(p-t-p) 셀 전송기능에 많은 에러 발생), 반면에 시스템 시험에서는 3개월까지 급증하다가 이후 5개월간은 완만한 곡선을 이룬다. 시험시작 후 9개월 때 많은 결함이 발생하는 원인은 영구가상



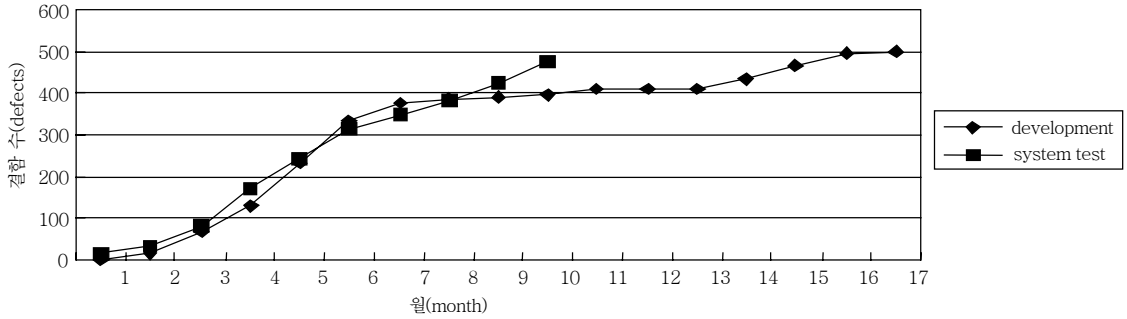
(그림 4) 1차 배포시 개발기간 및 시스템시험 기간동안 검출된 누적 결함 수

회로에 대한 PVC 점대다중점(p-t-mp) 기능과 B-ISUP(B-ICI) 중계호 처리 기능에 대한 강화로 관련 라이브러리 파일(library file) 변경 등 소프트웨어 컴포넌트에 많은 변화가 있어 결함이 급증하는 추세를 보였으며 이후 배포직전(post release 전으로 가장 많은 결함이 제거됨)까지는 거의 변화가 완만하였다.

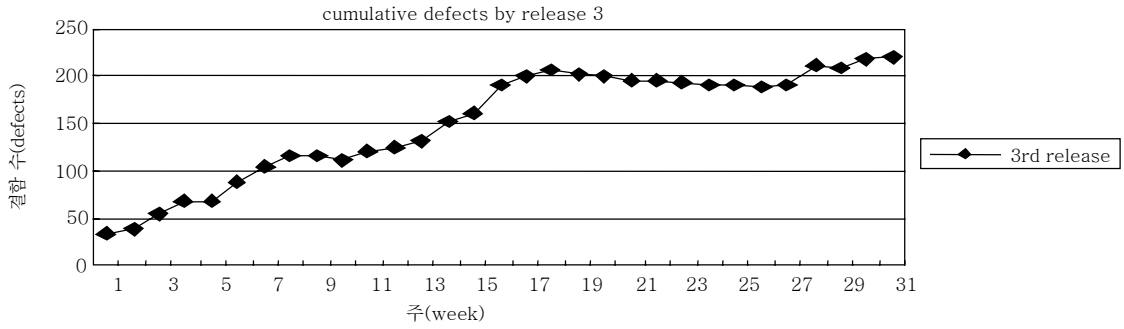
2차 배포 버전의 주요 결함 발생 부분은 국가망 1,2차 기능 수용에 따른 시그널링에 의한 SVC 호처리 강화 및 중계호 처리, 과부하 제어 및 망관리 기능 등 비정상 운용상태가 발생하는 경우에 시스템의 서비스 중단(down)없이 원활하게 처리되도록 하는 유지보수 기능의 강화에 따른 문제들이 주로 발생하고 있다. 대체로 결함 발생 경향은 개발기간이나 시스템시험 기간 둘 다 비슷한 양상을 띠나 시험시작 9개월 후 PSTN 연동기능 등의 수용에 따른 시스템 시험에 일시적으로 많은 에러가 발생하고 있음을 (그림 5)에서 보여주고 있다. 이 기능은 3차 배포 이전에 별도의 기능개발로 추진함으로써 결함데이터가 줄어드는 경향을 보인다(<표 11>의 괄호 안의 새로운 데이터는 신규로 추가된 소프트웨어 컴포넌트 수를 의미함).

(그림 6)의 3차 배포시 발생된 문제점 즉, 결함데이터는 주로 SVP 자국 및 중계호 처리 관련 소프트웨어 컴포넌트와 PNNI 연동 기능을 처리하기 위한 소스 라우팅(source routing), Hello protocol을 이용한 인접 노드의 상태관리 등 동적경로 선택(dynamic routing) 처리 및 호처리 관련 컴포넌트와 망의 계층적 구조(network topology)에 대한 데이터 처리 기능의 변경에 따라 컴포넌트의 변경이 이루어졌다.

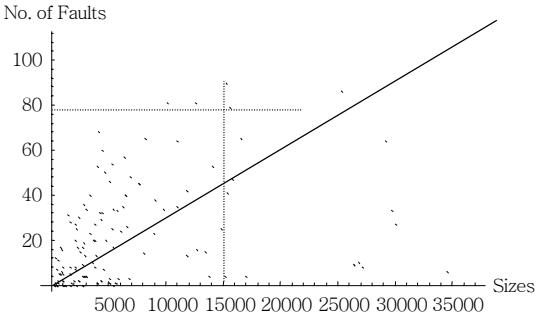
(그림 7)은 136개의 소프트웨어 컴포넌트에 대한 fault 발생 수에 대한 분석 데이터로 각 컴포넌트의 규모는 대략 500~15,000 라인 정도이며 결함 발생 수는 5~80개 전후로 밝혀졌다. 소프트웨어 규모 즉, 소프트웨어 컴포넌트의 규모가 큰 것(15,000 라인 이상인 컴포넌트)은 주로 TMN(Telecommunication Management Network) 기능의 기능 블록들로 여러 개의 컴포넌트들이 통합 수용되어 연동되기



(그림 5) 2차 배포시 개발기간 및 시스템시험 기간 동안 검출된 누적 결함 수



(그림 6) 3차 배포시 시스템 시험기간(주별) 동안 검출된 누적 결함 수



(그림 7) 소프트웨어 컴포넌트 규모별 fault 수

때문이다.

소프트웨어 컴포넌트의 설계 수준 및 안정도를 판단해 볼 때 안정권에 속하는 블록보다는 불안정한 블록들이 약간 많은 것으로 분석되었는데 이는 초기 시스템 설계시 많은 시행착오를 범한 것으로 판단된다. 특히, 시스템 개발시 초기 설계(design)가 매우 중요하다는 점을 시사한다. 이에 대한 극복은 체계적인 시스템 개발체계 및 우수한 도구, 개발방법론(design methodology)의 도입과 개발 경험의 축적 등이다. 이 분야에 대한 꾸준한 투자와 연구가 뒤따라야 가격경쟁력을 보유한 신뢰성과 안정성을 갖는 고품질의 시스템을 개발할 수 있는 것이다.

V. 결 론

우리가 분석한 결과는 시스템 시험에 도움을 줄 수 있는 indicator로서 시스템 개발기간 및 시험기간, post release 시기에 검출된 결함을 분석하여 시스템 개발관리 및 시스템 시험 가이드라인을 제공하는 데이터로 활용될 수 있도록 하였다. 물론 이 데이터들은 다른 프로젝트에서도 활용이 가능하다.

제안된 방법은 모든 프로젝트의 개선된 가이드라인으로 될 수는 없지만 특수한 경우 severity에 의한 분석 방법 등은 시스템 시험의 성능을 개선하는데 많은 도움이 될 것이다. 즉, 아래와 같은 용도로 사용될 때 그 효율성은 많이 개선될 수 있다.

- 미 검출된 많은 문제들의 파악
- 대부분의 알고 있는 문제들은 배포 전에 해결된다는 점
- 배포 직전에 문제점이 적어지는 점
- 컴포넌트들은 점진적으로 확장이 가능
- 특정 블록에 많은 문제가 분포된다는 점
- 코드의 품질이 향상

고품질의 소스를 개발하기 위한 방법으로 결함이 많이 존재하고 있는 컴포넌트에 대해 집중적인 시험을 가해 잠재하고 있는 결함을 검출, 수정함으로 인해 소프트웨어의 질 향상과 시스템 시험의 수행효율성의 판단근거로 컴포넌트의 fault prone 여부를 분석함으로써 자원의 배분 및 경제성을 꾀할 수 있다.

향후 연구사항으로는 다양하고 다각도로 소프트웨어 컴포넌트에 대한 결함 발생 특성을 연구하는 방법으로 이러한 경험들은 프로젝트 수행에 많은 활용이 될 것으로 기대된다.

참 고 문 헌

[1] J. Musa, "Applying Failure Data to Guide Decisions," Software Reliability Engineering, NY: McGraw-Hill, 1998.

[2] M. Sahinoglu and A.K. Alkhalide, "A Compound Poission LSD Stopping Rule for Software Reliability," *Proc. 5th World Meeting of ISBA*, satellite meeting to ISI-97, Istanbul, 1997.

[3] M. Ohlson, A. von Mayrhauser and B. McGuire, C. Wohlin, "Code Decay Analysis of Legacy Software through Successive Release," *Proc. IEEE Aerospace Conf.*, Section 7.401, 1999.

[4] P. Frankl, R. Hamlet and B. Littlewood, L. Strigini, "Evaluating Testing Methods by Delivered Reliability," *IEEE Transactions on Software Engineering*, Vol. 24, No. 8, 1998, pp. 586 - 601.

[5] T.M. Khoshgoftaar and E.B. Allen, "Predicting the Order of Fault prone Modules in Legacy Software," *Proc. Ninth Int. Symposium, Software Reliability Engineering*, Paderborn, Germany, Nov. 1998, pp. 344 - 353.

[6] T.M. Khoshgoftaar and R.M. Szabo, "Improving Code Churn Predictions During the System test and Maintenance Phases," *Proc. Int. Conf. Software Maintenance*, Victoria, BC, Canada, Sep. 1994, pp. 58 - 66.

[7] N.F. Schneidewind, "Software Metrics Model for Quality Control," *Proc. Int. Symp. Software Metrics, Albuquerque*, New Mexico, 1997, pp. 127 - 136.

[8] S. Biyani and P. Santhanam, "Exploring Defect Data from Development and Customer Usage on Software Modules over Multiple Releases," *Proc. Ninth Int. Symp. Software Reliability Engineering*, Paderborn, Germany, Nov. 1998, pp. 316 - 320.

[9] D. Ash, J. Alderete and P.W. Oman B. Lowther, "Using Software Models to Track Code Health," *Proc. Int. Conf. Software Maintenance*, Victoria, BC, Canada, Sep. 1994, pp. 154 - 160.

[10] M. Ohlsson and C. Wohlin, "Identi Cation of Green, Yellow and Red Legacy Components," *Proc. Int. Conf. Software Maintenance*, Bethesda, Washington, DC, 1996, pp. 6 - 15.

[11] N. Fenton, and N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," *IEEE Transactions on Software Engineering*, Vol. 26, No. 8, 2000, pp. 797 - 814.

[12] Victor R. Basill, Lionel C. Briand, and Walcelio L. Melo, "A Validation of Object-oriented Design Metrics as Quality Indicator," *IEEE Transaction On software Engineering*, Vol. 22, No. 10, Oct. 1996, pp. 751 - 761.

[13] 유재연, 이재기, "기능 블록으로 구성된 대형 교환 소프트웨어의 신뢰도 성장," 대한전자공학회논문지, 제35권 S편 제1호, 1998. 1., pp. 29 - 38.

[14] 이재기외 2, "고장 데이터 분석을 통한 교환 소프트웨어 특성 연구," 한국통신학회 논문지 제23권 제8호, 1998. 10., pp. 1915 - 1925,

[15] 이재기외 3, "대형 교환 소프트웨어의 복잡성과 고장분석 사례 연구," 한국통신학회논문지 제27권 제10호, 2002. 11., pp. 887 - 901.