적용조건 매트릭스를 이용한 신뢰성 척도 식별 기법

박진희*, 최옥주*, 백종문*, 신주환**
*한국과학기술원 전산학과
**국방과학연구소

e-mail : *{jh_park, okjoo.choi, jbaik}@kaist.ac.kr, **jhshin@add.re.kr

An approach of the Reliability Metrics Identification Using an Application Condition Matrix

Jin-Hee Park*, Okjoo Choi*, Jongmoon Baik*, Ju-Hwan Shin**
*Dept. of Computer Science, KAIST
**Agency for Defense Development

요 약

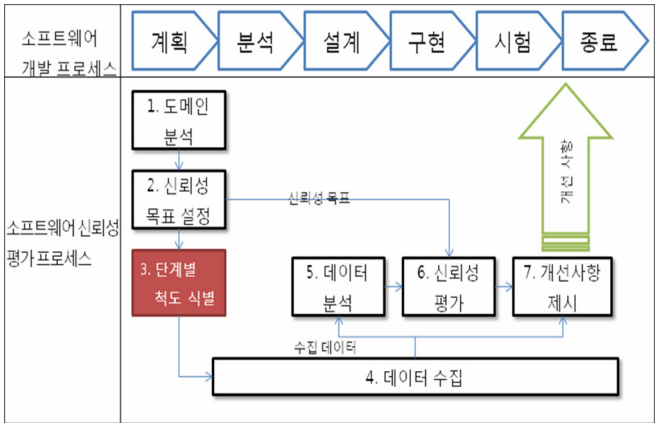
소프트웨어 신뢰성 보증을 위한 소프트웨어 신뢰성 평가 프로세스는 다양한 관점에서 대상 소프트웨어 시스템뿐만 아니라 관련된 조직 및 개발 프로세스에 대한 분석을 필요로 한다. 특히, 신뢰성 평가 프로세스 수행 단계 중 단계별 척도 식별단계는 대상 시스템의 도메인 특성을 반영해야 하는 까다로운 작업이다. 현재 신뢰성 관련 척도들이 여러 문서에 다양한 의미로 혼재해 있어 이를 조사하는데 많은 노력이 들며 대상 시스템 환경에 적합한 척도 식별을 위해 불필요하게 많은 회의와 인터뷰를 진행하고 있는 실정이다. 본 논문에서는 표준문서 및 관련문헌에 근거하여 신뢰성 척도 POOL 을 구성하고 각 척도 원시 데이터의 수집 조건에 기반한 적용조건 매트릭스를 이용하여 기존의 신뢰성 관련 척도 조사 및 수집, 척도 식별 과정에서 소요되는 자원, 시간의 낭비를 줄이고자 한다. 이 방법을 적용하였을 때 소요된 시간과 기존의 회의를 통해 소요되는 시간을 비교하여 본 논문에서 제시하는 방법의 효과를 평가한다.

1. 서론

대상 시스템의 소프트웨어 신뢰성 보증을 위해서는 소프트웨어 개발 프로세스의 특성을 반영하여 소프트웨어 신뢰성 예측 및 추정을 통해 대상시스템의 신뢰성 평가를 수행하여야 한다. 이러한 일련의 과정을 소프트웨어 신뢰성 평가 프로세스[1]라고 하며 개발 프로세스의 활동들과 연계되어 수행된다. 다음은 7 단계의 소프트웨어 신뢰성 평가 프로세스에 대한 설명이다.

- 도메인 분석: 소프트웨어 신뢰성 평가 대상 시스템 및 개발 환경의 분석
- 소프트웨어 신뢰성 목표 설정: 대상 시스템의 소프트웨어 신뢰성 목표 수준 정의
- 소프트웨어 신뢰성 관련 단계별 척도 식별: 대상 시스템의 소프트웨어 신뢰성 분석을 위한 척도 식별
- 데이터 수집: 소프트웨어 개발 단계별 데이터 수집
- 데이터 분석: 수집된 원시 데이터 및 척도를 이용하여 데이터 분석
- 소프트웨어 신뢰성 평가: 소프트웨어 신뢰성 예측 및 신뢰성 추정
- 개선사항 제시: 데이터 분석 및 소프트웨어 신뢰성 평가를 통한 개선사항 제시 및 개발 프로세스와 평가 프로세스의 개선사항 적용 및 평가

(그림 1)은 소프트웨어 신뢰성 평가 프로세스의 각 수행단계를 개발프로세스와 연계하여 도식화한 것이다.



(그림 1) 소프트웨어 신뢰성 평가 프로세스

이 논문은 ITRC(Information Technology Research Center)의 지원으로 연구를 수행하였습니다.(IITA-2008-(C1090-0801-0032))

평가 프로세스에서 언급된 소프트웨어 척도는 소프트웨어 특징 또는 특성을 객관적인 수치로 정량화 할 수 있도록 하는 기술이며 프로젝트 상태를 평가하기 위해 필요한 중요한 요소이다.

본 논문은 소프트웨어 신뢰성 평가 프로세스 수행 단계 중 소프트웨어 신뢰성 관련 척도 식별에 집중하여 현재 방식의 문제점과 개선점을 제시한다. 2 장에서는 소프트웨어 신뢰성 관련 척도 식별 단계와 문제점에 대해 설명하고 3 장에서는 현재의 문제점을 개선하기 위해 본 논문이 제안하는 척도 POOL 과 적용조건 매트릭스를 설명한다. 4 장에서는 기존의 척도 식별 방식과 적용조건 매트릭스를 적용한 방식을 비교하는 사례 연구를 수행하였고 5 장에서는 결론 및 향후 연구 방향에 대해 설명한다.

2. 소프트웨어 신뢰성 관련 척도 식별 및 문제점

소프트웨어 신뢰성 관련 척도 식별은 조직 구성 및 문화, 개발 환경과 같은 도메인 정보를 고려하여 수집 가능한 척도들을 정의하고 식별하는 활동이다. 내부 수행활동으로는 소프트웨어 척도 조사와 소프트웨어 신뢰성 관련 척도 식별 활동이 있다.

2.1 소프트웨어 척도의 조사

소프트웨어 신뢰성 관련 척도를 식별하기 위해서는 소프트웨어 공학에서 다루어지는 일반적인 척도들에 대한 조사가 선행되어야 하며, 고려해야 할 품질 속성에 대한 조사가 이루어 져야 한다. 하지만 수많은 문헌에 다양한 척도들이 존재하며 이러한 모든 척도들을 조사하는 것은 불가능하다. 그리고 사용되지 않는 척도들이 산재해 있고 척도들이 일관성 있게 정의되지 않아 도메인에 따라 다르게 해석되기도 한다. 이는 척도에 대한 이해를 어렵게 하며 잘못된 결과를 도출할 수도 있다. 즉, 신뢰성 관련 소프트웨어 척도 조사에 많은 시간을 투입하게 된다.

2.2 소프트웨어 신뢰성 관련 척도의 식별

소프트웨어 신뢰성 척도 식별을 위해서는 우선 신뢰성과 관련된 측정 목표가 수립되고 유지되어야 한다. 그리고 측정 목표를 달성하기 위한 척도를 식별하며 이는 대상 시스템의 개발 프로세스를 바탕으로 수행된다. 식별된 척도는 문서화되며 검토과정을 거쳐 목적과 의도한 활용에 맞지 않을 경우에는 개정한다. 아래 <표 1>는 소프트웨어 척도 식별 시 고려할 사항들을 정리한 것이다.

위와 같은 일련의 작업은 이해 당사자들 간의 수많은 회의와 인터뷰 과정을 요구한다. 이 과정에서 복잡하고 불필요한 척도들을 모든 이해 당사자가 이해해야 하는 문제가 발생하고 이는 조직의 자원, 시간의 낭비로 이어진다. 만약 다루어야 하는 척도가 많다면 소프트웨어 신뢰성 관련 척도의 식별 활동에 더욱 많은 시간이 소요될 것이다.

<표 1> 척도 식별 시 고려 사항

식별 시 고려해야할 사항
■ 측정에 따른 데이터가 조직과 소프트웨어 신뢰성 측정 목표와 목적을 지원한다.
■ 수집될 측정 데이터, 이 데이터에 대한 정확한 정의, 각 측정의 용도와 분석, 데이터가 수집될 프로세스 통제 지점을 고려한다.
■ 측정은 전체 소프트웨어 생명 주기를 대상으로 한다.
■ 측정은 소프트웨어 개발 프로세스 활동과 작업 산출물의 속성, 신뢰성 평가 속성들을 다룬다.
■ 측정 데이터는 소프트웨어 프로젝트 전반에 걸쳐 동일한 방식으로 수집된다.
■ 측정은 분석 활동을 지원하기 위해 선택된다.

3. 적용조건 매트릭스 제안

3.1 척도 POOL 구축

우선 2.1 절에서 언급한 문제점을 보완하고자 표준 문서[2]와 대표적인 신뢰성 척도 관련 문헌[3-8]을 바탕으로 총 56 개로 구성된 신뢰성 관련 척도 POOL 을 구축하였다. <표 3>의 적용조건 매트릭스의 M1~M56 이 척도 POOL 의 리스트이며 새로운 척도의 발견이나 필요성이 없어짐에 따라 지속적으로 수정, 보완될 것이다.

3.2 적용조건 매트릭스

3.2.1 구성 방법

각 척도는 여러 원시데이터들을 이용하여 도출될 수 있다. 그러므로 척도 계산을 위해 필요한 원시 데이터는 템플릿이나 수집 툴 등의 다양한 방법을 통해 수집되어야 한다. 원시 데이터를 수집할 수 없으면 해당 척도는 대상 시스템의 개발 프로세스 환경에 적용할 수 없는 척도이다. 척도 POOL 구성을 위해 참고한 자료 [2-8]에 명세 되어 있는 척도의 사전조건과 원시데이터 수집조건을 분석하여 <표 2>적용조건 리스트를 도출하였다. 결국, 대상시스템의 개발 프로세스 환경이 척도적용조건 리스트에 나열된 조건을 갖추고 있느냐 없느냐에 따라 해당 척도를 적용할 수 있느냐 없느냐를 판단할 수 있다.

<표 2> 적용조건 리스트

No	적용 조건	응답
C1	심각도에 따른 결점 수집이 가능하다.	O
C2	KSLOC, KSLOD 수집이 가능하다.	O
C3	객체지향 설계를 통한 개발을 수행한다.	O
C4	요구사항의 Entity-Relationship-Attribute model 표현이 가능하다	X
C5	요구사항에 대한 원인 결과 그래프형태로 도식화가 가능하다	X
C6	요구사항 준수율 검증을 위해 system verification diagrams(SVDs)를 사용한다.	X
C7	변경된 요구사항을 구현하기 위해 필요한 메모리 공간을 측정할 수 있다.	X
C8	Inspection 수행에 소요된 시간의 정확한 측정이 가능하다	O
C9	디자인 및 테스트 단계에서 요구사항에 대한 추적이 가능하다.	O
C10	복잡도 계산이 가능한 툴을 사용한다.	O
C11	각 모듈에서의 entry, exit 포인트 수를 측정할 수 있다.	X
C12	시스템의 모든 모듈이 DB를 사용한다.	X
C13	같은 도메인에 과거 인도된 제품의 기능변경에 대한 정보를 얻을 수있다.	X
C14	결점이 삽입된 단계와 제거된 단계를 기록할 수 있다.	O
C15	테스트 조직이 따로 존재한다.	X
C16	테스트 시, 실패간의 시간 측정이 가능하다.	O
C17	테스트 시, 주어진 시간 간격에서의 실패수를 측정할 수 있다.	O
C18	테스트 시, 각 모듈내부 논리적 흐름이나 연산자의 수를 측정할 수 있다.	X
C19	이전 SW release로부터 추가,삭제,수정된 소스 코드의 크기를 구할 수 있다.	X
C20	release 후의 결점을 추적할 수 있다.	O
C21	테스트 수행시, 시스템 특정기능의 평균지속시간을 파악할 수 있다.	X
C22	SW와 HW 실패율을 구분하여 측정할 수있다.	X
C23	제품 특성에 대한 설문조사를 모든 개발자에게 수행할 수 있다.	O

<표 3> 적용조건 매트릭스

No	Metrics Name	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23
IEEE-Std 982.1-1988 [2]																								
M1	Fault Density	X	X																					
M2	Defect Density	X	X																					
M3	Cumulative Failure Profile																	X						
M4	Fault-Days Number														X									
M5	Functional or Modular Test Coverage									X														
M6	Cause and Effect Graphing					X																		
M7	Requirement Traceability									X														
M8	Defect Indices	X																X						
M9	Error Distribution(s)	X													X									
M10	Software Maturity Index													X										
M11	Manhoues per Major Defect Detected								X															
M12	Number of Conflicting Requirements				X																			
M13	Number of Entries and Exits per Module											X												
M14	Software Science Measures										X								X					
M15	Graph-Theoretic Complexity for Architecture										X								X					
M16	Cyclomatic Complexity										X													
M17	Minimal Unit Test Case Determination																		X					
M18	Run Reliability															X								
M19	Design Structure												X											
M20	Mean Time to Discover the Next K Faults																X							
M21	Software Purity Level																	X						
M22	Estimated Number of Faults Remaining (by Seeding)															X								
M23	Requirements Compliance						X																	
M24	Test Coverage									X														
M25	Data or Information Flow Complexity																		X					
M26	Reliability Growth Function																X	X						
M27	Residual Fault Count																X	X						
M28	Failure Analysis Using Elapsed Time																X							
M29	Testing Sufficiency																	X						
M30	Mean-Time-to-Failure																X							
M31	Failure Rate																X	X						
M32	Software Documentation and Source Listings																							X
...	...																							
Stephen H.Kan "Metrics and models in software quility engineering" second edition [5], Daskalaonakis, M.K., "A practical View of Software Measurement ... [6]																								
M40	Total Defect Containment Effectiveness														X						X			
...	...																							
Schneidewind Model, Fisher and Walker [7]																								
M46	Time to next failure																X	X						
...	...																							
LINDA M.et al, "Software Measurement and Estimation:A Practical Approach", Wiley interscience, 2006 [8]																								
M51	Defect Removal Effectiveness														X									
M52	Operational Profile Testing																					X		
M53	Inspection의 효과성								X															
M54	Module defect density																X	X						
M55	Information Flow Complexity											X												
M56	Object Oriented Design			X																				

3.2.2 매트릭스 적용 방법 및 효과

<표 2>의 C1~C23 은 신뢰성 관련 척도들이 대상 프로젝트에 적용되기 위한 조건들을 나열한 것이다. 대상 조직의 실무자들에게 <표 2>적용조건 리스트에 O, X 로 응답하도록 한다. X 로 응답된 조건을 <표 3> 적용조건 매트릭스에서 찾고 해당열의 X 로 표시된 척도를 최종 척도 리스트에서 제외시킨다.

<표 2>에서 X 로 응답된 모든 적용조건들을 고려하여 제거해 나가면 대상 시스템의 개발 프로세스 환경에서 수집될 수 없는 척도를 파악할 수 있게 된다.

이와 같은 방법으로 대상 시스템의 개발 프로세스 환경에 적용 불가능한 척도를 제외시킬 수 있고 고려해야 하는 척도의 수가 훨씬 줄어들기 때문에 회의시간과 인터뷰 횟수를 단축 시킬 수 있다. 이렇게 줄어든 척도리스트를 바탕으로 수집노력대비 효율성이나 적용가능성을 따져 가며 최종 척도 리스트를 추출해야 할 것이다.

4. 사례 연구

본 장에서는 3 장에서 제시한 방법과 기존의 방법을 비교하여 적용조건 매트릭스의 효과를 확인해 보고자 한다. 신뢰성 보증을 수행하려는 국방 도메인의 한 업체를 대상으로 사례연구를 수행하였으며 소프트웨어 조직의 실무자들과 회의 및 설문조사를 진행하였다.

4.1 적용조건 매트릭스를 이용한 방법

대상 조직의 소프트웨어 개발자들에게 <표 2>의 적용조건 리스트에 응답하도록 하였다. 하나의 조건에

대한 응답이 다를 경우는 더 많은 응답이 제시된 쪽으로 선택하였고 그 결과를 <표 2>의 적용조건과 함께 나열하였다. 이를 바탕으로 <표 3>의 적용조건 매트릭스에 적용하였고 수집될 수 없는 척도를 제외시켰다. 그 결과, 24 개의 척도를 척도 POOL 에서 제외시켰고 나머지 32 개를 적용 가능한 척도로 추천하였다. 이 과정에서 총 20 분의 시간이 소요되었다.

4.2 기존 방법

위의 적용조건 매트릭스를 통해 제외된 24 개의 척도에 대한 회의를 진행하였다. 이를 통해 적용조건 매트릭스의 적용 결과를 검증하고 기존 방법을 통해 24 개의 척도 적용여부를 판단하는데 드는 시간을 파악하고자 한다. 회의 참석자는 컨설턴트 2 명과 적용조건 매트릭스를 위한 조사에 참가하지 않은 실무자 1 명이다.

회의 결과, 적용조건 매트릭스에 의해 제외된 척도의 100%가 실제 조직의 개발 도메인 환경에서 수집될 수 없는 척도로 판단되었다. 이는 적용조건 매트릭스가 어느 정도 정확도를 가진다는 것을 의미한다. 총 회의 소요시간은 2 시간 30 분이었고 각 척도의 의미를 이해하고 현재의 개발 프로세스 환경에서 데이터 수집이 가능한지 여부를 판단하는데 많은 시간이 소요되었다.

4.3 결과

기존의 방법으로 24 개의 척도를 제외시키는데 7 시간 30 분(2 시간 30 분 • 3 명)이 걸렸다면 적용조건 매트릭스(1 시간 = 20 분• 3 명)를 이용함으로써 6 시간 30 분의 시간을 절약할 수 있었다. 이는 프로젝트 관련자들이 회의를 통해 고려해야 하는 척도의 개수를 그만큼 줄일 수 있다는 것을 의미한다. 아래의 <표 4>는 두 척도식별 방법을 비교한 표이다. 단, 기존의 회의를 통한 방법으로 도출된 결과는 100% 정확하다고 가정한다.

<표 4> 척도식별방법 비교

	기존 방법(회의)	제안한 방법(적용조건 매트릭스)
제외된 척도 리스트	M6, M10, M12, M13, M14 M15, M17, M18, M19, M22 M23, M25, M34, M35, M36 M37, M38, M39, M43, M44 M45, M47, M52, M55	M6, M10, M12, M13, M14 M15, M17, M18, M19, M22 M23, M25, M34, M35, M36 M37, M38, M39, M43, M44 M45, M47, M52, M55
부적합 척도 제거율	-	42.9%(24/56)
정확성	100%(가정)	100%
참석자	3명	3명
수행 시간	2시간 30분	20분

5. 결론 및 향후 연구

본 논문에서는 소프트웨어 신뢰성 평가 프로세스 상에서 척도식별을 보다 쉽게 할 수 있는 방법을 제안하였다. 제안한 척도 POOL 을 이용하면 척도조사에 드는 비용을 절약할 수 있으며 해당 프로젝트에 적용될 수 없는 신뢰성 관련 척도들을 적용조건 매트릭스를 통해 미리 제거함으로써 복잡하고 불필요한 척도이해에 드는 자원, 시간의 낭비를 줄일 수 있다. 또한, 절약된 시간을 바탕으로 적용 가능한 척도들에 더욱 집중한다면 이해관계자들의 척도에 대한 이해를 향상시킬 수 있고 보다 정확한 척도 식별이 가능할 것으로 기대된다. 이는 전체 신뢰성 평가 프로세스의 효과에도 기여할 수 있을 것이다.

향후 연구로는 제안된 척도 POOL 에서 비현실적이거나 사용되지 않는 척도들을 식별하고 IEEE Std 982.1-2005 문서를 적용하여 기존의 척도 POOL 을 수정하는 작업을 하고자 한다. 척도 POOL 의 척도리스트와 적용조건 매트릭스는 새로운 척도가 등장하거나 필요성이 없어짐에 따라 지속적인 갱신이 이루어져야 할 것이다. 이러한 테일러링 과정을 통해 믿을만한 척도 POOL 을 만들어 보다 정확한 척도 추출이 가능할 것으로 기대된다.

참고문헌

[1] Dalju Lee, “Software Reliability Assurance Using a Framework in Weapon System Development : A Case Study”, ACIS, 2009

[2] IEEE Std 982-2, "IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software", IEEE Computer Society, 1988

[3] ISO 9126-2, "ISO/IEC 9126-2: Software engineering-Product quality-Part 2: External metrics", ISO/IEC JTC1/SC7, 2000

[4] ISO 9126-3, "ISO/IEC 9126-2: Software engineering-Product quality-Part 2: External metrics", ISO/IEC JTC1/SC7, 2000

[5] Stephen H.Kan, “Metrics and models in software quality engineering”, second edition, Addison Wesley

[6] Daskalaonakis, M.K., “A practical View of Software Measurement and Implementation Experiences Within Motorola(1001-1004)”, IEEE Transaction

[7] IEEE Std 982.1 – 1988 : Schneidewind Model, Fisher and Walker(M48)

[8] LINDA M.et al, “Software Measurement and Estimation : A Practical Approach”, Wiley interscience, 2006