소프트웨어 분석체계

박대성[0], 강성원
한국정보통신대학교 공학부
{grayger, kangsw}@icu.ac.kr

Towards the Discipline of Software Artifacts Analysis

Daesung Park[0], Sungwon Kang
School of engineering, ICU

요 약

소프트웨어 분석은 어떠한 관점을 가지고 소프트웨어 산출물의 속성을 평가하고 평가결과에 대한 원인을 밝히는 행위를 말한다. 이 논문에서는 분석의 체계를 정립하기 위하여 다음을 수행하였다. 첫째 분석의 정의를 내리고, 종합과 비교되는 분석의 일반적 개념, 평가, 측정, 측정법과의 관계를 밝혔다. 둘째, 분석의 관점이 되는 속성을 이해하고자, 속성들을 특징에 따라 분류하였다. 마지막으로 속성별, 산출물별로 기존의 분석 방법을 조사하고 기존의 분석 방법을 평가하였다.

1. 서론

본 연구에서는 소프트웨어 산출물의 분석을 위한 체계를 제시한다. 소프트웨어 산출물의 도출은 기본적으로 한번에 기계적으로 이루어지는 작업이 아니라, 분석과 구축이 반복됨으로써 고품질의 결과를 얻을 수 있는 창조적인 작업이다. 따라서 개발의 다양한 단계의 산출물에 정확한 분석을 수행하는 능력은 궁극적으로 고품질의 시스템을 구축하는 추진력(driving force)이 된다.

보통 소프트웨어라고 말할 때, 소프트웨어 프로그램 또는 실행시스템을 말한다. 소프트웨어공학에서는 소프트웨어는 개발생명주기에 도출되는 모든 소프트웨어 산출물(software artifacts)을 총칭하는 용어로, 특히 이 중에서도 분석의 대상이 되는 산출물은 시스템명세, 소프트웨어 구조, 소프트웨어설계명세, 프로그램코드, 구현시스템이다. 본 연구에서는 본 연구에서는 이러한 소프트웨어 산출물들에 대하여 일반적으로 적용될 수 있는 분석체계를 제시한다. (그림 1.1)

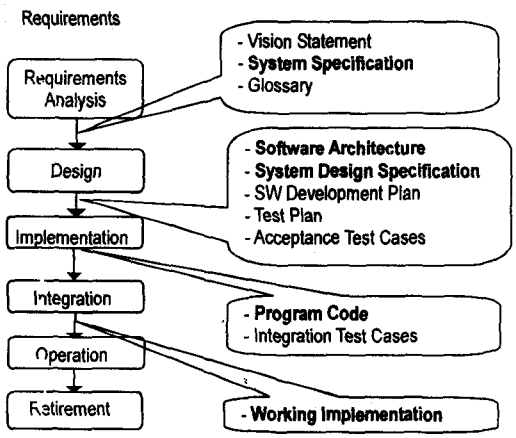

그림 1.1 소프트웨어 개발 단계와 각 단계별 산출물

2. 배경

분석(analysis)이란 전체를 구성하는 부분들로 분해하고 부분들의 상호관계를 탐구하는 활동으로 규정할 수 있다. 분석과 상반되는 것이 종합(synthesis)으로 분리된 부분들을 묶어서 큰 덩어리를 만들어 나가는 활동을 의미한다.

종합은 구체적인 부분들로부터 큰 전체를 만드는 과정이고, 분석은 큰 전체를 작은 부분들의 상관관계로 분해하고 이해하는 활동이다. 소프트웨어개발에 있어서는 개발과정이 전체에 대한 명세에서 출발하여 그 전체를 형성할 수 있는 구체적인 부분들을 만들어가는 과정으로 점차 구체화 되가는 과정으로, 이 과정을 통하여 실질적으로 존재하지 않는 전체가 존재하게 되므로, 이 과정이 종합에 해당된다. 분석은 대상을 분석적으로 이해하는 과정이다.

2.1 소프트웨어 분석의 개념

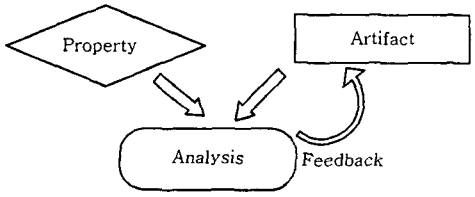

그림 2.1 소프트웨어 분석의 역할

각 개발 산출물 별로 필요한 분석의 관점과 분석의 방법이 다르다. 그러나 분석은 기본적으로 산출물로 형상화된 대상에 대한 탐구 및 평가를 의미하며 따라서 일반적으로 그림 2.1에서와 같이 분석결과를 반영하여 산출물을 개선하는데 사용된다. 여기에서 산출물이 개선되는 과정은 창조적인 과정이다.

분석을 위하여는 분석의 대상이 되는 산출물과 분석의 관점(viewpoint)을 제공하는 속성(property)이 있어야 한다. 속성들은 크게 기능, 성능, 품질에 관한 속성으로 분류할 수 있다. 기능은 시스템이 의도된 대로 올바르게 일을 수행하는 능력을 말한다. 성능은 시스템이 기능을 얼마나 많은 양의 처리를 단위시간에 하는 가를 말한다. 예를 들어 시스템이 트랜잭션(transaction)을 완수하는데 걸리는 시간이나, 단위시간에 처리한 트랜잭션의 수로 나타낼 수 있다. 이는

시스템의 기능적인 올바름 혹은 올바르지 않음과 독립적으로, 또는 기능을 얼마나 생산적으로 처리하는나를 떠나서, 다른 어떤 좋은 자질들을 가지고 있느냐를 말한다. 품질 속성의 예로는 신뢰성(reliability), 사용성(usability), 보안성(security), 유지보수성(maintainability), 확장성(scalability) 등이 있다.

2.2 소프트웨어 분석 - 용어

용어의 기원

Halstead는 [1]에서 소스 코드의 줄 수로 여러 요인들을 숫자로 표현하여 분석을 시도하였다. [2]는 75년에서부터 94년까지 ICSE에 실린 논문에서 '분석'이 등장하는 것들을 조사하였다. 이에 따르면 그 기간 동안 '분석'이라는 용어가 59번 등장하였으며 처음에는 요구 분석, 프로그램 분석, 데이터 흐름 분석 같이 산출물에 대한 분석을 의미하다가, 점차 대상이 함수 점 분석, 생산성 분석, 비용 효과 분석을 포함하게 되었으며 현재는 프로세스까지 대상으로 하고 있다

분석과 평가

분석은 순수한 기술과 방법을 이용하며, 평가(evaluation)는 디자인에 사용하기 위해 사용된다. 분석은 계량적 결과를 표현하는 것이 바람직하나 계량적 결과를 도출하는 것이 어려운 경우 수량화되지 않은 평가결과를 갖게 될 수 있다.

분석과 측정, 측정법

소프트웨어 측정(measure)은 소프트웨어 속성을 수량화하기 위해 소프트웨어 공학 영역의 대상을 수나 벡터 같은 수학적 구조로 매핑하는 것이다[3]. 예를 들어 프로그램의 독립적인 경로의 수를 계산함으로써 시험성(testability)과 유지보수성을 알아볼 수 있다. 측정은 소프트웨어 생명 주기에서 프로세스와 엔티티를 개선시키기 위한 피드백을 제공한다. 측정은 다음과 같이 3가지로 분류할 수 있다:

- 크기 기반 측정 - 프로그램 길이, 프로그램 분량(volume), 프로그램 레벨, 노력, 시간
- 자료 기반 측정 – 변수의 수 등
- 논리 기반 측정 – 사이클로매틱복잡도(cyclomatic complexity), 도달성(reachability), 중첩수위(nesting level)

소프트웨어 측정법(measurement)은 정의된 골을 얻기 위해 측정을 적용하는 기술 혹은 방법으로 다음과 같은 것들로 특성을 나타낼 수 있다[3].

- 측정법의 대상 –산출물, 프로세스, 프로젝트
- 측정법의 방법 – 특성화, 평가, 예측
- 측정법의 소스 – 소프트웨어 디자이너, 테스터, 관리자
- 측정할 속성 – 비용, 신뢰성, 유지보수성, 크기, 이식성
- 측정법의 상황 – 소프트웨어 산출물이 측정되는 다른 환경

이와 같이 측정은 잘 정립된 수량화된 결과로 대상에 대한 분석을 매핑(mapping)하는 활동이다. 여기서 분석이라 함은 포괄적인 의미의 "대상에 대한 이해" 활동을 의미한다.

분석의 정의

반면 분석이라고 할 경우 대상이 바람직한 속성을 얼마나 가지고 있는가 하는 평가를 포함하여 바람직한 속성을 가지고 있지 않을 경우 그 원인이 무엇인가를 밝히는 과정을 포함한다.

분석, 평가, 측정의 관계를 도식화 하여보면 평가는 분석과 측정을 포함하는 개념으로 평가 중에 특히 엄밀한 수치를 결과로 얻어내는 평가를 측정이라고 하고, 분석은 평가결과에 대한 원인까지를 규명하는 활동을 의미한다.

3. 분석의 관점

대상의 분석을 위하여는 분석의 관점이 필요하다. 분석의 관점은 대상의 입장에서 볼 때에는 속성(property)를 말한다. 즉 어떤 바람직하거나 바람직하지 않은 특징이 있는지 없는지, 만일 그 정도를 따질 수 있는 종류의 속성이라며 어느 정도 들어 있는지 그리고 그러한 원인이 무엇인지를 파악하게 된다.

소프트웨어 산출물의 속성은 여러 개의 관련된 속성들의 그룹으로 분류될 수 있다. 먼저 소프트웨어시스템의 실행과 관련된 속성과 실행과는 관계없는 속성으로 나눌 수 있다.

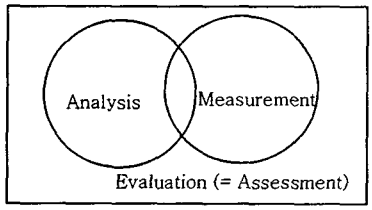

그림 3.1 분석, 평가, 측정의 관계

3.1 동적 속성

실행과 관련된 속성으로는 표 3.1과 같은 속성들이 있다.

표 3.1 동적속성(dynamic property)

속성	정의
기능(functionality)	시스템이 의도한대로 작동하는 능력
성능(performance)	이벤트에 반응하는 시간이나, 시간 당 처리할 수 있는 이벤트 수
보안성(security)	사용자에게 서비스를 제공할 때 인가되지 않은 시도를 막아내는 능력
신뢰성(reliability)	정확한 서비스를 지속적으로 제공하는 능력[4].
가용성(availability)	시스템이 살아서 동작하는 시간의 비율
사용성(usability)	사용자가 쉽고 사용하며, 시스템이 사용자에 빠르게 반응하는 능력

3.2 정적 속성

정적 속성에는 표 3.2와 같은 속성들이 있다.

표 3.2 정적속성(static property)

	속성	정의
비진화적 속성	이식성 (portability)	시스템이 다른 환경에서 동작할 수 있는 능력
	재사용성 (reusability)	시스템의 구조나 컴포넌트가 미래의 다른 응용에 재 사용되는 능력
	통합성 (Integrability)	따로 개발된 시스템 컴포넌트들이 함께 정확하게 작동하는 능력
	상호운영성 (interoperability)	통합성(integrability)의 한 종류로 시스템을 이루는 부분의 그룹이 다른 시스템과 함께 동작하는 능력이다[5].
진화적 속성	변경가능성 (modifiability)	빨리, 적은 비용으로 변화시킬 수 있는 능력
	유지보수성 (maintainability)	소프트웨어 시스템이나 컴포넌트가 오류를 수정하거나, 성능이나 다른 속성을 향상시키거나 새로운 환경에 변화시키는데 쉬운 정도

비진화적 속성과 진화적 속성

비실행속성은 다시 크게 비진화적 속성과 진화적 속성으로 나눌 수 있다. 비진화적속성은 어느 시점의 소프트웨어 형상으로부터 파악해 낼 수 있는, 파악의 대상이 되는 속성이다. 진화적 속성은 현재에서 시간의 흐름에 따라 변화하는 속성이다. 동적 속성은 모두 비진화적 속성에 속하며, 정적 속성 중 유지보수성(maintainability)와 변경가능성(modifiability)는 진화적 속성이며 확장성(scalability)와 재사용성(reusability)는 비진화적이면서도 진화적 속성이라고 말할 수 있으며 그 나머지는 비진화적 속성이다.

- 비진화적: functionality, reliability, ...
- 진화적 : maintainability, modifiability, ..
- 비진화적/진화적(hybrid): scalability, reusability, ...

3.3 기존의 분석 방법

앞에서 속성을 실행속성과 비실행속성으로 나누었다. 따라서 분석방법도 크게 실행속성의 관점에서 분석을 수행하느냐, 비실행속성의 관점에서 분석을 수행하느냐에 따라 정적 분석과 동적 분석으로 나눌 수 있다.

정적 분석은 프로그램 실행 없이 행해진다. 정적인 분석은 분석할 산출물의 텍스트를 조사한다. 정확성, deadlock 검사, safety/liveness 같은 속성을 체크한다. 이 결과는 모든 가능한 실행을 반영한다. Modeling checking은 개별적인 실행/상태를 평가하나, 모든 실행/상태에 적용된다. 그러나 symbolic execution은 소스 텍스트를 조사하나, 개별적인 실행/상태를 요약하는 것이다.

동적 분석은 테스팅이라고도 하며, 개별적인 프로그램의 실행을 조사한다. 결과는 조사하여 실행한 것만을 반영한다. 동적 분석은 실행 가능한 코드를 필요로 하기 때문에 개발의 초기 단계에서는 적용할 수 없다. 또한 테스팅으로는 시간에 의존적인 오류 같은 것은 찾기가 힘들다. 따라서 정적 분석과 동적 분석은 상호 보완적이다.

품질목표가 검증 가능하지 않으면 우리는 그것을 만족시켰는지 말할 수 없다. 따라서 각 품질속성마다 분석할 수 있는 방법들이 필요하다. 프로그램의 정확성은 증명할 수 없는 대신 단순화된 모델의 간단한 속성은 증명할 수 있다.

보통 개발의 앞 단계일수록 수학적 개념적 방법을 사용하고 뒤 단계일수록 실험적인 방법을 사용한다. 비용절감을 위해 개발의 앞 단계에서 분석이 이루어질 수 있게 하여야 하는데 솔루션 영역의 첫 단계인 구조단계로 끌어올리는 것이 바람직하다.

표 3.3은 산출물에 따른 각 속성의 분석 방법을 나타낸다. 비어있는 셀은 어떤 산출물로 어떤 속성을 분석하기에 적절한 방법이 없거나 아직 제시되지 않음을 의미한다. 요구사항 명세에서부터 코드의 분석은 정적 분석이고 실행시스템의 분석은 동적 분석이다. 각 속성마다 분석의 방법이 다르고, 또한 개발의 생명주기에서 적용될 수 있는 단계가 다르다.

또한 산출물로부터 직접적으로 분석하는 경우와, 직접 분석하는 것이 적절치 않아 모델을 만들어 내고 그 모델을 분석하는 경우가 있다. 예를 들어 Z명세를 이용할 경우 산출물에서 바로 기능요구사항을 만족시키는지 분석할 수 있지만, 신뢰성을 분석할 경우 명세를 이용하여 모델(Markov model)을 끄집어 낸 뒤 분석 가능하다[6]. 분석한 결과는 정제되어 분석의 대상이 된 산출물의 도출에 반영된다.

4. 결론

이 논문에서 소프트웨어 분석의 정의를 기초로 하여 산출물에 따른 각 속성을 분류하고 이러한 분류에 기초하여 과거 어떤 분석방법의 연구가 있었는가 알아보았다. 이러한 분류체계는 분석대상 방법에 대한 깊은 이해를 주고 과거의 분석방법을 새로운 대상에 활용될 수 있는 방향을 제시한다. 본 연구의 분석체계는 분석대상의 속성에 대하여 체계적인 분류를 제시하지만 분석방법에 대하여는 아직까지 체계적인 분류를 제시하고 있지 못하다. 향후 본 연구를 분석방법에 대한 체계적인 분류로 확장시킴과 동시에 새로운 분석방법에 대한 연구를 수행할 예정이다.

참고문헌

[1] Maurice H. Halstead. Elements of Software Science. Elsevier, Amsterdam, 1977.

[2] K. Torii, "Analysis in software engineering," Proceedings of The 1994 Asia-Pacific Software Engineering Conference , 1994.

[3] James F. Peters, Witold Pedrycz, *Software Engineering: an Engineering Approach*, Wiley, 1999.

[4] Bass, Clements, Kazman, *Software architecture in practice*, pp 75~91, 2002.

[5] A.Avizienis, J.C.Laprie , B.Randell, "Fundamental concepts of dependability", 3rd Information Survivability Workshop (ISW'2000), Boston (USA), pp.7-12, October 2000.

[6] K. Goseva - Popstojanova and K. S. Trivedi, "Architecture Based Software Reliability", Proc. of ASSM 2000 Int. Conf on Appplied Stochastic System Modeling, March 2000

[7] Bruce Lewis, "Software portability gains realized with MetaH and Ada95", Proceedings of the 11th international workshop on Real-time Ada workshop, 2002

표 2.3 산출물에 따른 각 속성의 분석 방법

Property	Artifact	Requirement specification	Architecture	Detail design	Code	Executable
기능	Functionality			*		
	Security		MetaH	Access Control model, Information Flow Model		
	Timing		MetaH	*		
성능	Performance		Queueing network			Profiler
품질	Maintainability		*		Maintainability Index	
	Portability		MetaH [7]			
	Reusability		*			
	Reliability	Markov model	State based Path based models			