RESORT™ 자바 품질 메트릭스 솔루션

이헌기
㈜소프트4소프트

Solutions of RESORT™ Java Quality Metrics

Lee, Heon-Ki
Soft4Soft Co., Ltd.
E-mail: lee@icu.ac.kr

요 약

RESORT™ 품질 솔루션은 Java 언어로 작성된 소프트웨어로부터 품질을 측정하고 평가하기 위한 자바 품질 메트릭스 자동화 도구로서 RESORT™-Java 제품군 중 하나이다. 본 논문에서는 System/Package/Class 단위로 Java Code의 Product Metrics등을 측정하여 소프트웨어 품질을 평가하는데 사용되는 도구들을 기술한다. 이 도구들은 5 종류의 소프트웨어 메트릭스 솔루션을 제공한다: OO Metrics, Package Metrics, Halstead Metrics, Quality Metrics, System Level Metrics. 소프트웨어 메트릭스는 전체 개발 비용의 60% 이상을 차지하는 유지보수의 비용을 줄이고, 고품질의 소프트웨어를 개발하기 위해서 반드시 필요하다. 또한, 소프트웨어의 생산성을 높일 수 있을 뿐 아니라 신뢰성 향상, 그리고 유지보수에 대한 효율성을 향상시킬 수 있다.

1. 서론

소프트웨어 개발은 요구분석, 설계, 구현, 테스트, 및 유지보수의 작업들이 순차적, 순환적, 또는 병행하여 수행되는 생명주기를 거친다. 소프트웨어의 개발 시, 소프트웨어 개발 생명주기의 각 단계에서 소요되는 비용에 대한 통계적인 상대 분포는 소프트웨어 개발 전체 비용의 60% 이상이 소프트웨어 제품의 성능/품질을 향상시키기 위한 유지보수 작업에 소요되고 있는 실정이다.

소프트웨어 관리 기술 중 하나인 품질관리 기술은 흔히 소프트웨어가 소스코드로 구현이 완료된 후 최종 제품에 대한 작업으로 여겨지고 있으나, 실제로 소프트웨어 품질은 코드 크기(Code Size), 코드 스타일(Code Style), 구조(Structure), 복잡도(Complexity), 설계(Design), 테스팅 커버리지(Testing Coverage), 품질(Quality) 등의 측정/평가 작업들을 포함하고 있다. 따라서 소프트웨어 품질관리는 소프트웨어 관리 기술에 국한되기보다는 소프트웨어 품질과 관련한 작업 요소들의 성격에 따라 소프트웨어 개발 생명주기 전반에 걸쳐서 수행되어야 한다.

본 논문에서는 객체지향 Java 언어를 이용하여 소프트웨어를 개발하거나 또는 개발된 기존의 Java 소프트웨어를 관리 할 때, Java 소프트웨어용 역공학, 재공학, 테스팅, 품질 관리 등을 통해 Java 소프트웨어의 성능 및 품질을 향상시키고,

재사용과 유지보수성을 높여주기 위한 "RESORT™-Java 통합 도구(Java 소프트웨어용 역공학, 재공학, 테스팅, 품질 측정/평가 통합 도구)" 중에서 RESORT™ 품질 솔루션 자동화 도구에 관한 특징을 기술한다.

RESORT™ 품질 솔루션 자동화 도구는 System/Package/Class 단위로 Java Code의 Product Metrics등을 측정하여 소프트웨어 품질을 평가하는데 사용되는 도구이며, 5 종류의 소프트웨어 메트릭스 솔루션(Metrics Solutions)을 지원한다: OO Metrics, Package Metrics, Halstead Metrics, Quality Metrics, System Level Metrics.

본 논문에서 제 2장은 객체지향 메트릭스들에 대한 관련 연구 동향, 제 3장은 당사가 개발한 소프트웨어 개발 및 관리 도구인 RESORT™-Java 제품군 소개, 제 4장은 OO Metrics, 제 5장은 Package Metrics, 제 6장은 Halstead Metrics, 제 7장은 Quality Metrics, 제 8장은 System Level Metrics 도구들을 살펴보고, 마지막으로 제 9장에서 결론을 맺는다.

2. 관련 연구

소프트웨어 메트릭스의 필요성은 다음과 같다.

(1) 모든 소프트웨어 시스템의 상세 설계 및 아키텍처 분석 및 이해
(2) 소프트웨어 설계 시, 근원적인 에러 식별
(3) 소프트웨어 단위 테스팅의 작업 지원
(4) 소프트웨어 품질(생산성과 제품 품질) 측정 및 평가
(5) 소프트웨어 프로젝트의 비용과 개발 기간 예측 및 평가
(6) 재사용 가능한 소프트웨어 컴포넌트 식별
(7) 소프트웨어 외주 관리 지원

몇 십년 동안, 많은 연구가들의 경험을 바탕으로 새로운 메트릭스들이 개발되어 왔으며, 또한 이와 같은 다양한 종류의 메트릭스들은 절차적 및 객체 지향적 프로그래밍과 같은 서로 다른 프로그래밍 패러다임과 함께 개발되어 왔다. 이들 가운데에서, LOC(Lines of Code)는 가장 기본적이고 가장 오래된 메트릭스 중 하나이다.

1990년 초기에 S. Chidamber와 C. Kemerer가 제안한 6가지 새로운 객체지향 메트릭스가 있다 [2]. 이들 메트릭스들은 Class 기반 설계 단계에서 사용될 수 있다. M. Lerenz와 J. Kidd는 소프트웨어 설계 단계에서 정적인 특성을 다루는 Project Metrics와 Design Metrics를 제안 했다 [5]. B. Abreu와 W. Melo는 시스템 레벨 기반 설계 단계에서 객체 지향 패러다임을 측정할 수 있는 MOOD (Metrics for Object Oriented Design) 메트릭스를 제안하였다 [1].

그리고, 1970년 말에 T. McCabe (Cyclomatic Complexity) [7], M. Halstead (Software Science) [3] 등이 제안한 코드 기반 절차적 메트릭스들은 지금까지도 널리 사용되고 있다.

이러한 소프트웨어 메트릭스들을 기반으로 다양한 자동화 품질 CASE(Computer-Aided Software Engineering) 도구들이 다수 존재한다. 대부분 CASE 도구들은 기본적으로 Attribute과 Method들을 포함한 Class 레벨 메트릭스를 지원하고 있지만, 프로젝트 관리자 및 책임자에게 필요한 종합적인 System/Package 관점의 메트릭스 솔루션들이 부재하다는 단점을 갖고 있다.

일반적으로 Java 아키텍처는 System/Package/File/Class/Method별로 다른 개념을 갖고 있기 때문에, 각각의 단위를 기반으로 최종 System을 측정 및 평가할 필요가 있다. 이러한 필요성을 고려해서 RESORT™ 품질 솔루션 자동화 도구에서는 Java 아키텍처에 따라 소프트웨어 메트릭스를 5 종류의 카테고리로 분류하고, 각 카테고리는 정의된 메트릭스의 측정 유형(Measure Type)에 따라 크기(Size), 구조(Structure), 복잡도(Complexity), 품질(Quality) 등의 용어로 Product Metrics 등을 측정하고 평가한다.

3. RESORT™-Java 제품군 소개

본 제품은 객체지향 Java 언어용 소프트웨어 메트릭스 기반 소프트웨어 분석, 테스팅, 품질관리 솔루션 도구이다. 특히, 소프트웨어 메트릭스 관점에서 개발자, 관리자, 책임자 등의 요구 사항을 동시에 만족시켜 주는 다양한 품질 메트릭스 솔루션들을 지원한다.

본 제품군은 [그림 1]과 같이 18 종류의 도구로 구성되어 있으며, 주요 기능은 다음과 같다.

(1) 역공학 도구

프로그램의 Control Flow Graph, Data Flow Graph, Source Code Browser 등의 절차적 프로그램 정보와 Class Diagram, Sequence Diagram, Statechart Diagram, Package Diagram 등의 UML 1.4 다이아그램을 자동 생성하여 소프트웨어의 구조, 기능 , 행위 분석을 지원하고, 분석된 Software Metrics을 관리, 평가, 모니터링할 수 있는 도구이다.

(2) 재공학 도구

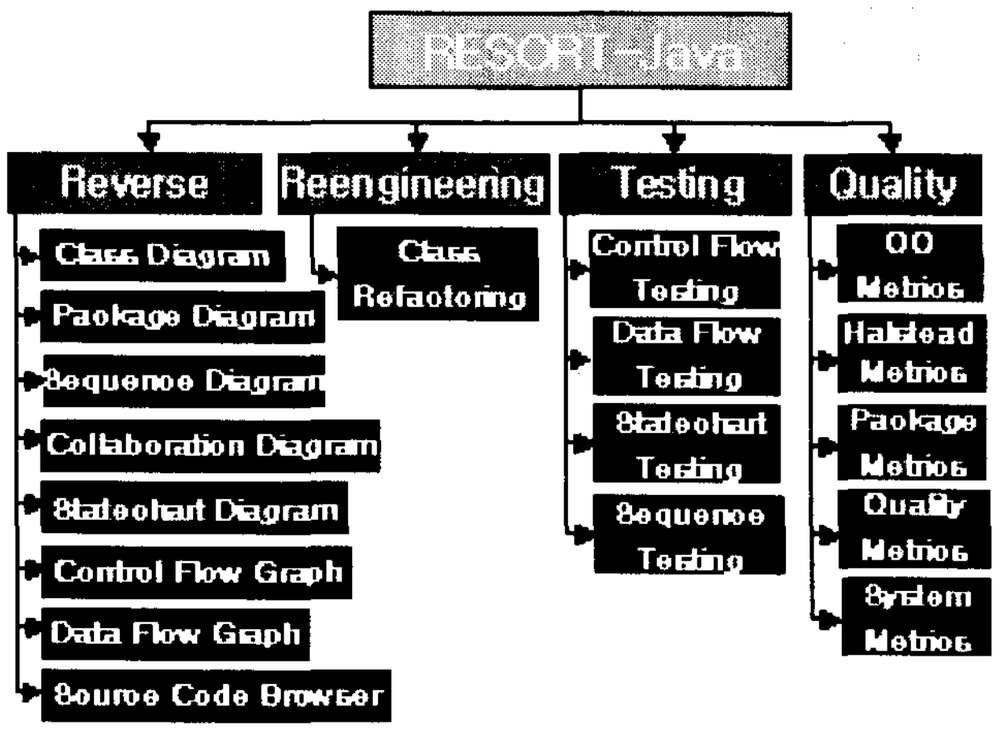
Class Diagram을 바탕으로 새로운 요구사항을 삽입/변경/삭제하여 재사용, 재개발 등을 할 수 있는 일종의 리팩토링(Refactoring) 도구이다.

(3) 테스팅 도구

프로그램의 제어 및 자료 흐름 테스팅과 그리고 UML의 Sequence와 Statechart 테스팅을 지원하고, 다양한 Coverage 등을 분석, 모니터링할 수 있는 도구이다.

(4) 품질관리 도구

소프트웨어 개발 및 조직 환경에 따라 각 Class Metrics 기준 값을 설정하며, 각각의 System/Package/Class 레벨 단위로 Software Metrics를 측정하고, 그 분석 결과를 다양한 품질 제어 그래프(Quality Control Graph)와 Check Sheet 등으로 자동 생성하여, 소프트웨어 프로젝트를 관리, 제어, 모니터링, 예측 및 평가할 수 있는 도구이다.



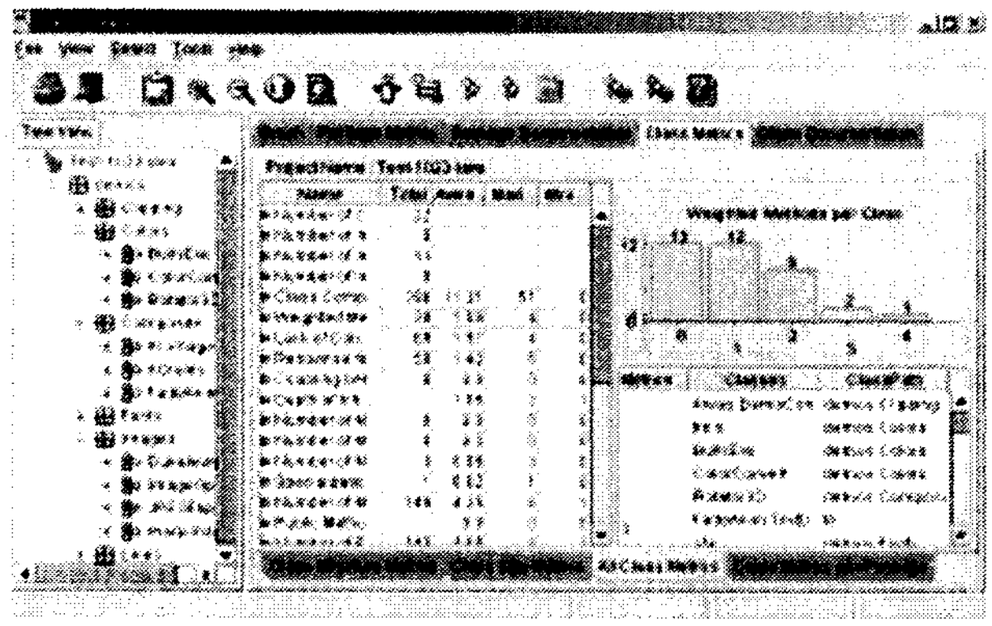
[그림 1] RESORT-Java 제품군

4. OO Metrics

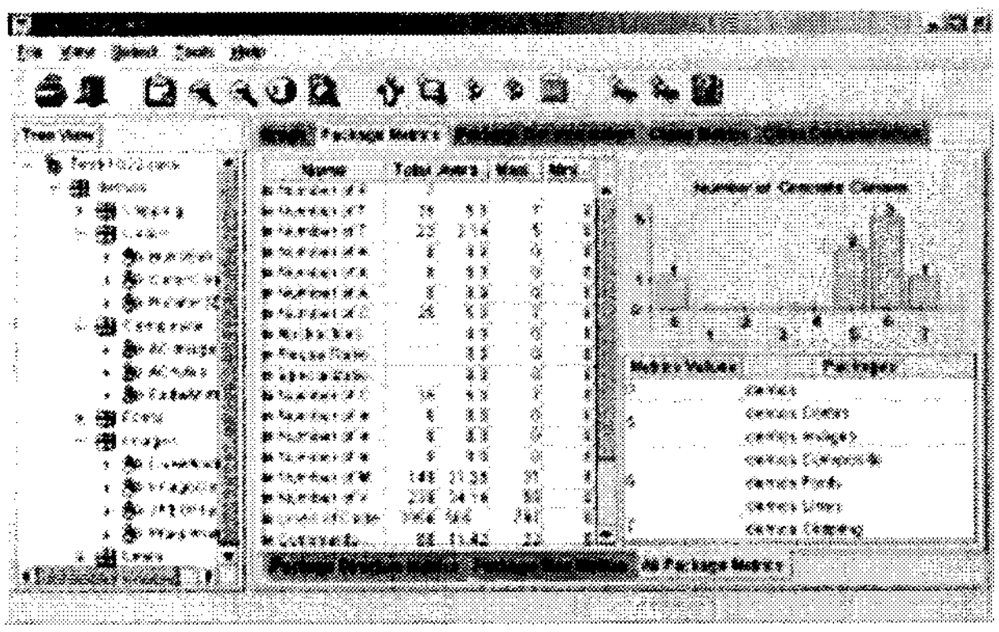
OO Metrics는 [표 1]과 같이 Attribute과 Method들을 포함한 Class의 다양한 내.외적 속성들을 측정하여 [2, 5, 8], 각 Class의 Metrics 값 또는 Grouping 값에 의해 System/Package/Class 레벨 단위로 소프트웨어 품질 평가를 지원한다. [그림 2]와 같이 측정된 품질을 관리, 평가 및 모니터링할 수 있는 Bar Graph & Check Sheet, Kiviat Graph, Metrics Table을 제공한다.

[표 1] Class Metrics

Measure	Name	Description
Size	LOC	Lines of Code
	STMT	Number of Statements
	CD	Comments Density
Structure	NOM	Number of Methods
	PMR	Public Methods Ratio
	NIM	Number of Instance Methods
	NSM	Number of Static Methods
	NOV	Number of Variables
	NIV	Number of Instance Variables
	NSV	Number of Static Variables
Complexity	WMC	Weighted Methods per class
Cohesion	LCOM	Lack of Cohesion of Methods
Coupling	RFC	Response for a Class
	CBO	Coupling between Objects
Inheritance	DIT	Depth of Inheritance Tree
	NCC	Number of Children Classes
	SIX	Specialization Index



[그림 2] Class Metrics



[그림 3] Package Metrics

5. Package Metrics

Package Metrics는 [표 2]와 같이 Package의 의존성, 재사용성 등을 측정하여 [2, 6, 8], 각 Package의 Metrics 값 또는 Grouping 값에 의해 System/Package 레벨 단위로 소프트웨어 품질 평가를 지원한다. [그림 3]과 같이 측정된 품질을 관리, 제어, 평가 및 모니터링할 수 있는 Bar Graph & Check Sheet, Kiviat Graph, Metrics Table을 제공한다.

[표 2] Package Metrics

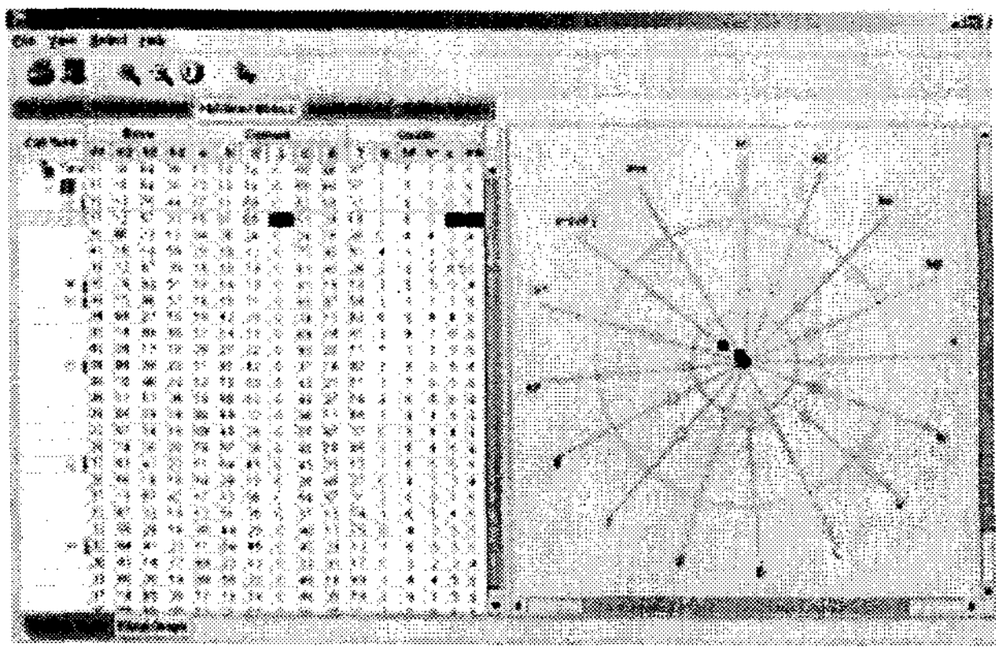
Measure	Name	Description
Size	LOC	Lines of Code
	STMT	Number of Statements
	CD	Comments Density
Structure	NOC	Number of Classes
	NIC	Number of Inner Classes
	NOI	Number of Interfaces
	NII	Number of Inner Interfaces
	PMR	Public Methods Ratio
Complexity	NOT	Number of Type
	TOPC	Number of Top Level Classes
	NOAT	Number of Abstract Classes
	NOAI	Number of Abstract Interfaces
	NOCC	Number of Concrete Classes
Reuse	RR	Reuse Ratio
	SR	Specialization Ratio
Dependency	AC	Afferent Couplings
	EC	Efferent Couplings
	ABS	Abstractness
	INST	Instability
	DIST	Distance

6. Halstead Metrics

Halstead Metrics는 [표 3]과 같이 기존 메트릭스와는 달리 소스 코드의 오퍼레이터(Operator)와 오퍼랜드(Operand) 계산에 의해 혼합 및 품질 측정을 통해 소프트웨어 생산성 및 품질 예측 등을 측정하여 [3], 각 Halstead의 Metrics 값 또는 Grouping 값에 의해 System/Package/ Class 레벨 단위로 소프트웨어 품질 평가를 지원한다. [그림 4]과 같이 측정된 품질을 관리, 평가, 예측 및 모니터링할 수 있는 Bar Graph & Check Sheet, Kiviat Graph, Metrics Table을 제공한다.

[표 3] Halstead Metrics

Measure	Name	Description
Basic	n1	Number of Unique Operators
	n2	Number of Unique Operands
	N1	Total Number of Operators
	N2	Total Number of Operands
Hybrid	n	Program Vocabulary
	N	Program Length
	V	Program Volume
	D	Program Difficulty
	E	Program Effort
	L	Program Level
Quality	T	Estimated Time
	B	Estimated Errors
	N^	Estimated Length
	V^	Potential Volume
	n^(VF)	Vocabulary Frequency
	PR	Purity Ratio



[그림 4] Halstead Metrics

7. Quality Metrics

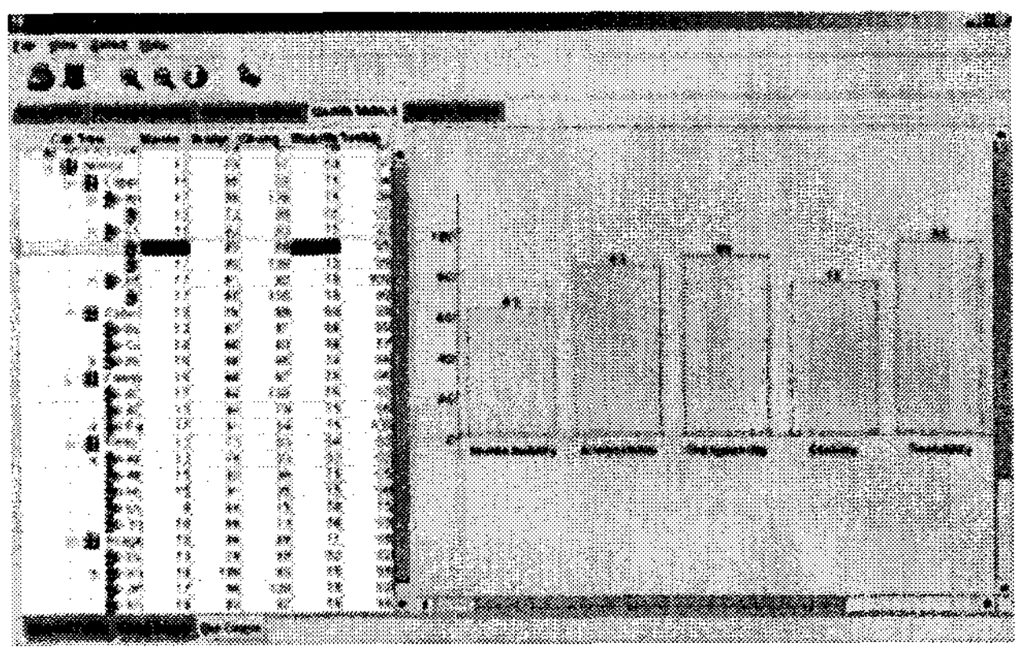
Quality Metrics는 Product의 Quality를 평가하기 위해서 ISO/IEC 9126 특성들 중 [4, 8], 변경 및 오류 사항의 교정에 대한 최소화 노력의 품질 목표인 유지보수성(Maintainability) 특성과 관련된 4 개의 부 특성을 측정한 후 [표 4], 그 부 특성의 측정 값과 가중치에 의해 유지보수성 특성을 측정한다 [표5]. 그리고, 각 Characteristic/Sub-Characteristic의 Metrics 값 또는 Grouping 값에 의해 System/Package/Class 레벨 단위로 소프트웨어 유지보수성의 품질 평가를 지원한다. [그림 5]와 같이 측정된 품질을 관리, 평가 및 모니터링 할 수 있는 Bar Graph& Check Sheet, Kiviat Graph, Metrics Table을 제공한다.

[표 4] Quality Metrics

Measure	Name	Description
Characteristic	Maintainability	Characteristics used to assess the effort required to make given modifications
Sub-Characteristic	Analyzability	Attribute of a class characterizing the effort necessary to diagnose failures or failure causes or to identify the parts of the source code to be modified
	Changeability	Attribute of a class characterizing the effort necessary to modify the class or remedy defects
	Stability	Attribute of a class characterizing the risk of unexpected consequences of modifications
	Testability	Attribute of a class characterizing the test effort necessary to validate the studied class.

[표 5] 측정된 값과 평가된 레벨

Scale Category	Rated Level
Satisfactory	Excellent
	Good
	Fair
Unsatisfactory	Poor



[그림 5] Quality Metrics

8. System Level Metrics

System Level Metrics는 [표 6]과 같이 All Package, All Class, All Method, All Attribute의 Grouping 값에 의해 동적인 소프트웨어 품질 측정을 지원한다 [5]. [그림 6]과 같이 측정된 품질을 평가, 예측 및 모니터링할 수 있는 Circle Graph와 Metrics Table을 제공한다.

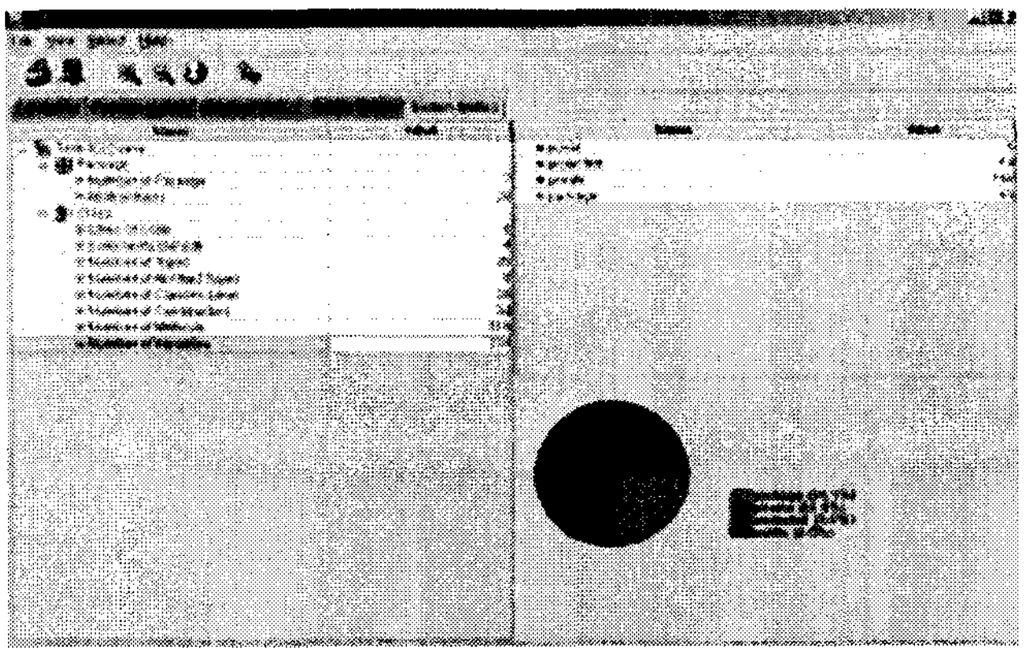
이러한 System Level Metrics는 소프트웨어 개발 생명주기상에서 전체 시스템 요구 사항을 평가하기 위한 프로젝트 예측 방법으로 사용될 수 있다.

그러나, Class Level Metrics들과 달리 System Level Metrics는 소프트웨어 개발 과정에서 소프트웨어의 품질을 측정하지 않는다. 즉, 전체적인

프로젝트 시각 차원에서 소프트웨어 개발 목표의 진행 과정을 관리, 제어, 예측 및 모니터링 하는 데 사용된다.

[표 6] System Metrics

Measure	Name	Description
All Package	NOP	Number of Package
	ABS	Abstractness
	RR	Reuse Ratio
	SR	Specialization Ratio
	LOC	Lines of Code
	CD	Comments Density
All Class	NOT	Number of Types
	NOAT	Number of Abstract Types
	NOCL	Number of Classes Level
	CS	Class Size
All Method	NOM	Number of Methods
	PMR	Public Methods Ratio
All Attribute	NOV	Number of Attributes



[그림 6] System Metrics

9. 결론

소프트웨어의 개발 및 관리에 관련된 소프트웨어 품질을 향상시키기 위한 유지보수 비용이 급증하고, 또한, 고품질의 소프트웨어 개발 중요성이 높아지고 있으며, 향후 지속적인 증가 추세를 보일 것으로 예상되고 있다. 따라서 소프트웨어 메트릭스 솔루션 자동화 도구는 유지보수의 고비용을 줄이고, 고품질의 소프트웨어 개발과 대규모 소프트웨어 외주에 대한 품질 관리를 하기 위해서 반드시 필요하다.

이러한 소프트웨어 품질관리는 소프트웨어 관리 기술에 국한되기보다는 소프트웨어 품질과 관련한 작업 요소들의 성격에 따라 소프트웨어 개발 생명주기의 전반에 수행함으로써 고품질의 소프트웨어 개발을 추진할 수 있도록 하는 것이 바람직하다.

[참고문헌]

[1] B. Abreu and W. Melo. Evaluating the Impact of Object-Oriented Design on Software Quality, *Proceedings of 3rd International Metric Symposium,* 1996.

[2] S. Chidamber and C. Kemerer. A metrics suite for object oriented design, *IEEE Transaction on Software Engineering,* 20(6): 476-493, 1994

[3] M. Halstead. *Elements of Software Science,* Elsevier North Holland 1977

[4] ISO/IEC 9126(1991), Quality Characteristics and Guidelines for Their Use

[5] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics,* Prentice Hall, 1994

[6] R. Martin. OO Design Quality Metrics, *OOPSLA ' 94, Workshop on Metrics,* 1994.

[7] T. McCabe. A complexity measure, *IEEE Transaction on Software Engineering,* 2(4): 308-320, 1994

[8] ㈜소프트4소프트. *재공학 소프트웨어 기술개발,* 정보통신부 2001