I.J.E.M.S., VOL.3(2) 2012: 109 - 114 ISSN 2229-600X

INTERNATIONAL JOURNAL OF ENGINEERING AND MANAGEMENT SCIENCES

QUALITY METRICS IMPLEMENTATION IN COMPONENT BASED SOFTWARE ENGINEERING USING AI BACK PROPAGATION ALGORITHM SOFTWARE COMPONENT

Sidhu Pravneet
SPCET, Mohali, Punjab, India 140 507

ABSTRACT
Quality in the Component Based Software Engineering has always been an area to be explored upon. Component Based Software Engineering is concerned with the rapid assembly of systems from components where components and component frameworks have certified properties and those serve as a purpose for predicting the properties of the system. CBSE generally, consists of modular design and development of applications based on software components, developed independently, and suitably combined to compose the final application. It deals with COTS where there software components are readily available for use without the code being written. The Quality in Component based software engineering is defined as the totality of characteristics of the solution that bear on its ability to meet the user's stated or implied needs. A software component is of high quality if it uses standard language features, it contains no machine dependencies and it implements a single, well-defined, encapsulated and precisely specified function whose computations is all fully adjustable and uses no global variables or side-effects. The quality attributes of a component are most often not a constant property. In this journal certain quality metrics have been used to indicate the exact quality of an Artificial Intelligence component which is the AI Back propagation Algorithm both at compile time and run time.

KEYWORDS: CBSE, COTS, AI Back propagation Algorithm, quality

INTRODUCTION

The Quality in Component Based Software Engineering is defined as the totality of characteristics of the solution that bear on its ability to meet the user's stated or implied needs. Quality of the components can also be increased in component based software engineering. In an ideal setting, a software component that is developed for reuse would be verified to be correct & would make no defects. In reality, formal verification is not carried out routinely, and defects can and do occur. However, with each reuse, defects are found and eliminated, and a component's quality improves as a result. Quality factors are visualized as by user's perspective view, certifier's perspective view and also by developer's perspective view. All the three have to be satisfied in order to have a high quality component. As the source code is not available along with the component so modifications & any sort of manipulations become difficult .In order to achieve a high quality component a prior modifications have to be made so that the component is checked in all sort of environments and is both reliable & fault tolerant.

Literature survey

Two early models[2] were described by McCall (1977) and Boehm (1978). In these models, the model-builders focus on the final product and identify the key attributes of quality from the user's point of view. The key attributes, also called quality factors, are normally external attributes, for example "maintainability" and "reliability" [1]. They may also include internal attributes though, such as "efficiency". Both the McCall model and the Boehm model assume that these quality factors are at too high a level to be meaningful or ever measurable. Because of this, the quality factors are further decomposed into lower level attributes, called quality criteria.

In 2001, Stafford and Wallnau developed a model for component marketplaces that support prediction of system properties prior to component selection. The model is concerned with the question of verifying functional and quality related values associated with the components. This work introduced notable changes in this area, since it presents a Component based development process with support for component certification according to credentials, provided by the component developer. Such credentials are associated with arbitrary properties and property values with components, using a specific notation such as <property, value, credibility>. Through credentials the developer chooses the best components to use in the application development based on the "credibility" level. Stafford and Wallnau also introduced the notion of active component dossier, in which the component developer packs a component along with everything needed for the component to be used in an assembly. A dossier is an abstract component that define certain credentials, and provides quality mechanisms that, given component, will fill in the values of these credentials. Stafford and Wallnau finalized their work with some open questions, such as: how to certify measurement techniques? What level of trust is required under different circumstances? Are there other mechanisms that might be used to support quality?

ISO International Standard 8402 has defined Quality Model as:-

"The set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality".

ISO 9126 model had 5 characteristics and did not take into account the Portability characteristic. Sub-characteristics fault tolerance, stability, analyzability were not considered at all. This model is shown in table 1.

Manuel F. Bertoa [2] modified the quality model proposed by ISO to define a quality model for COTS components.Xavier Franch et al. proposed a six steps methodology aimed at defining a quality model for a given software domain using ISO/IEC quality standard as framework in 2003.They adapted the methodology for its application in COTS based system domain- "ERP System" in 2004.

COTS quality model also had 5 characteristics and did not take into account the Portability characteristic. Sub-characteristics were divided into run time sub-characteristics and sub-characteristics at life cycle. Compatibility and complexity sub-characteristic were added in this COTS model. The COTS model is shown in table 2

The quality attributes of a component are most often not a constant property. Much more, the quality of a component heavily depends on the specific usage context. Therefore, we present a specification method for contractually specified components which does not specify quality attributes as constants but as functions to be evaluated at deployment. The "design-by-contract"-principle is applied to components. Parameterised contracts are used to compute the reliability of software components [3]. COTS quality model also had 5 characteristics and did not take into account the Portability characteristic. Sub-characteristics were divided into run time sub-characteristics and sub-characteristics at life cycle. Compatibility and complexity sub-characteristic were added in this COTS model. Refined COTS model is shown in table 3.

MATERIALS AND METHODS

Quality metrics: *Presence, IValues and Ratio*

A quality metric is the defined measurement method & the measurement scale and the measure is the number or category assigned to an attribute.Quality metrics would be used to evaluate the quality[14] level of components before the making the purchase or develop decisions. The possible metrics that can be used in component based software development would be [6]:-

- Management metrics (cost, time to market, software engineering environment and system resource utilization),
- Requirements metrics (requirements conformance, and requirements stability),
- Quality oriented metrics (adaptability, complexity of interfaces and integration, integration test coverage, end-to-end test coverage, fault profiles, reliability, and customer satisfaction metrics).

The metrics that will be used for measuring the attributes are the following [6]

Presence*:*

This metric identifies whether an attribute is present in a component or not. It consists of a Boolean value and a string. The Boolean value is used to indicates whether the attribute is present and, if so, the string describes how the attribute is implemented by the component;

***IValues*:**

This metric is used to indicate exact values of the component information's. It is described by an integer variable and a string to indicates the unit (e.g. kb, mb, khtz, etc.); and

Ratio*:*

This metric is used to describe the ratios.

One of the most compelling reasons for adopting component-based approaches to software development is the premise of reuse. The idea is to build software from existing components primarily by assembling and replacing interoperable parts. The implications for reduced development time and improved product quality make this approach very attractive. In this way, we aim to propose consistent and well-defined quality characteristics, quality attributes and related metrics for the component evaluation. A preliminary evaluation to analyze the results is also presented.

The AI Back Propagation component

The AI back propagation component is able to implement the back propagation algorithm which is used for training multilayer artificial neural networks. It can be applied to any number of layers. For convenience, maximum up to 4 layer networks are used. The GUI for AI back propagation network commands, snap shot for working of AI back propagation algorithm and AI Back Propagation component is shown in the Figure 1, 2 and 3 respectively.

Implementation of quality metrics

In implementation of component quality attributes for sub-characteristics during run time, we have shown the component attributes for Sub-characteristics that are observable at run time. In the component, the accuracy i.e the correctness level is present. Hence the value 1 is given as result. Same is the case with error handling. But response time is not known and so the value 0. The memory and the disk usage is known. In here we have taken the different attributes like correctness, error handling, response time, memory usage disk usage. Some metrics are of type ratio, some are presence and some are IV i.e. information values. The implementation is shown in Table 4.

In implementation of component quality attributes for sub-characteristics during life cycle, we have shown the component attributes for Sub-characteristics that are observable at life cycle. In the component, the suitability sub characteristic is present. Hence the value 1 is given as result. Same is the case with attribute document available. But failure removal i.e. the number of bugs fixed is not known. Metrics are of type ratio, presence and some are IV i.e. information values type. The implementation is shown in Table 5.

RESULTS AND DISCUSSION

In here, we have found certain quality metrics for the AI back propagation component. Both the quality metrics and their types have been asserted in the above tables. Quality metrics and quality attributes both at run time and at life cycle have been evaluated for the AI back propagation component. For clarity during run time of the component four sub-characteristics, namely accuracy, recoverability, time behavior and resource behavior are taken into account. For these sub-characteristics certain attributes are specified such as correctness, error handling, response time memory usage and disk usage. During life cycle sub-characteristic like suitability, maturity and understandability are visualized. These include attributes such as pre and post conditions, failure removal and document available.

CONCLUSION

Quality by product issue is especially critical in emergence of the software component market. By producing components satisfying the quality requirements, software producers become more enthusiastic to consume the market-oriented components. In return, a more matured market would become available which further encourages the component consumption and producing the software in component-based manner. An objective method has been described here to calculate the quality of the software component by using component quality metrics like presence, Ivalues and ratios.

FUTURE WORK

The software quality problem cannot be solved without changing the existing software development style, and establish a new development culture based on continuous process improvement. This requires that, the software development organizations must define formal processes and regularly collect data to assess and improve the development receiving a better organizational maturity. Other aspects of the quality can also be taken into consideration for a detailed structure analysis of the Software components like maintainability, availability and usability.

REFERENCES

B. Meyer, M. Jezequel, "Design by Contract: The Lessons of Ariane", *IEEE Computer*, Vol. 30, No. 02, 1997, pp. 129–130.

B. Boehm, C. Abts, and E. Bailey, "COCOTS Software Integral Cost Model: an Overview," *In Proceedings of the California Software Symposium,* 1998

ISO/IEC JTC1/SC7, Information Technology - Software product quality: Quality model,. ISO/IEC, 9126, 1999.

Kam-Fai Wong, Michael R. Lyu, Xia Cai-Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes;In Proceedings of the Seventh Asia-Pacific Software Engineering Conference(APSEC.00), IEEE 2000.

Raje R., .UMM: Unified Meta-object Model for Open Distributed Systems, Proceedings of the fourth IEEE International Conference on Algorithms and Architecture for Parallel Processing (ICA3PP'2000).

Brahnmath G., Raje R., Olson A., Sun C., .Quality of Service Catalog for Software Components., Technical Report (TR-CIS-0219-01), Department of Computer and Information Science, Indiana University Purdue University Indianapolis, 2001.

Table 1: ISO 9126 Quality Model

CHARACTERISTICS	SUB-CHARACTERISTICS
Functionality	Suitability
	Accuracy
	Interoperability
	Compliance
	Security
Reliability	Maturity
	Recoverability
Usability	Learnability
	Understandability
	Operability
Efficiency	Time behaviour
	Resource behaviour
Maintainability	Changeability
	Testability
	Installability
	Conformance
	Replacability
	Adaptability

Table 2: COTS Quality Model

CHARACTERISTICS	SUB-CHARACTERISTICS (RUN TIME)	SUB-CHARACTERISTICS (LIFE CYCLE)
Functionality	Accuracy Security	Suitability Interoperability Compliance Compatibility
Reliability Usability	Recoverability	Maturity Learnability Understandability Operability Complexity
Efficiency	Time behaviour Resource behaviour	
Maintainability		Changeability Testability

Table 3: COTS Quality Model

CHARACTERISTICS	SUB-CHARACTERISTICS (RUN TIME)	SUB-CHARACTERISTICS (LIFE CYCLE)
Functionality	Accuracy Securitysss	Suitability Interoperability Compliance Compatibility
Reliability	Recoverability	Maturity
Usability		Learnability Understandability Operability Complexity
Efficiency	Time behaviour Resource behaviour	
Maintainability		Changeability Testability

Table 4: Component Quality Attributes for Sub-characteristics at Runtime

SUB-CHARCTERISTICS	ATTRIBUTES	METRICS	KIND OF METRICS	RESULT
Accuracy	Correctness	Test results/Precision	R	1
Recoverability	Error handling	Mechanism implemented	P	1
Time behavior	Response time	Time taken between a set of invocations	IV	0
Resource behavior	1)Memory usage 2)Disk usage	1)Memory used 2)Disk used	IV IV	1 1

Table 5: Component Quality Attributes for Sub-characteristics at Life cycle

SUB-CHARACTERISTICS	ATTRIBUTES	METRICS	KIND OF METRICS	RESULT
Suitability	Pre & post conditions	Verification of pre & post conditions	P	1
Maturity	Failure removal	Number of bugs fixed	IV	0
Understandability	Document available	Document analysis	P	1

FIGURES

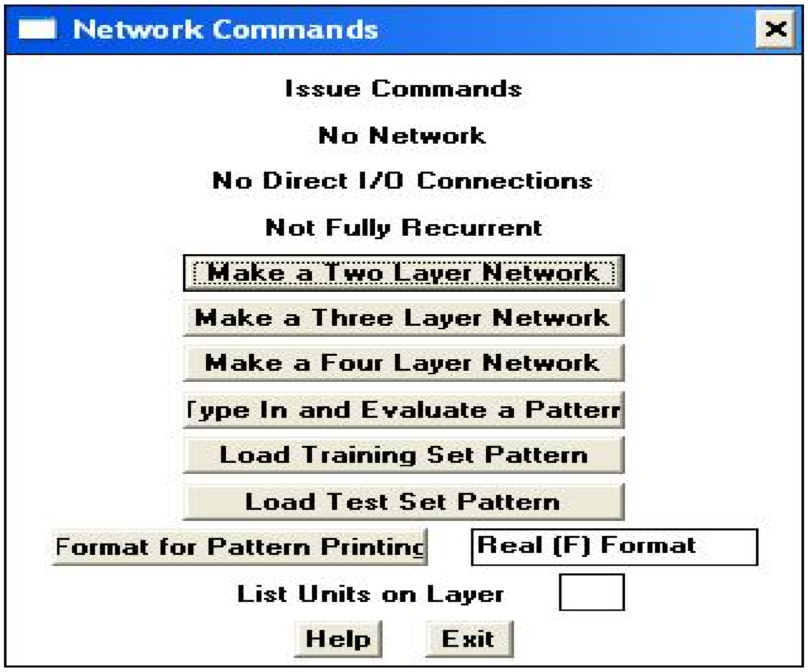
Network Commands
Issue Commands
No Network
No Direct I/O Connections
Not Fully Recurrent
Make a Two Layer Network
Make a Three Layer Network
Make a Four Layer Network
Type In and Evaluate a Pattern
Load Training Set Pattern
Load Test Set Pattern
Format for Pattern Printing
Real (F) Format
List Units on Layer
Help
Exit

Figure 1: GUI for AI back propagation network commands

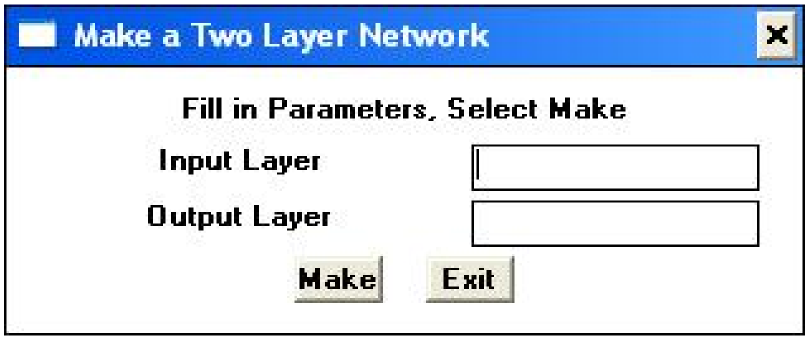

Figure 2: Snap shot for working of AI back propagation algorithm

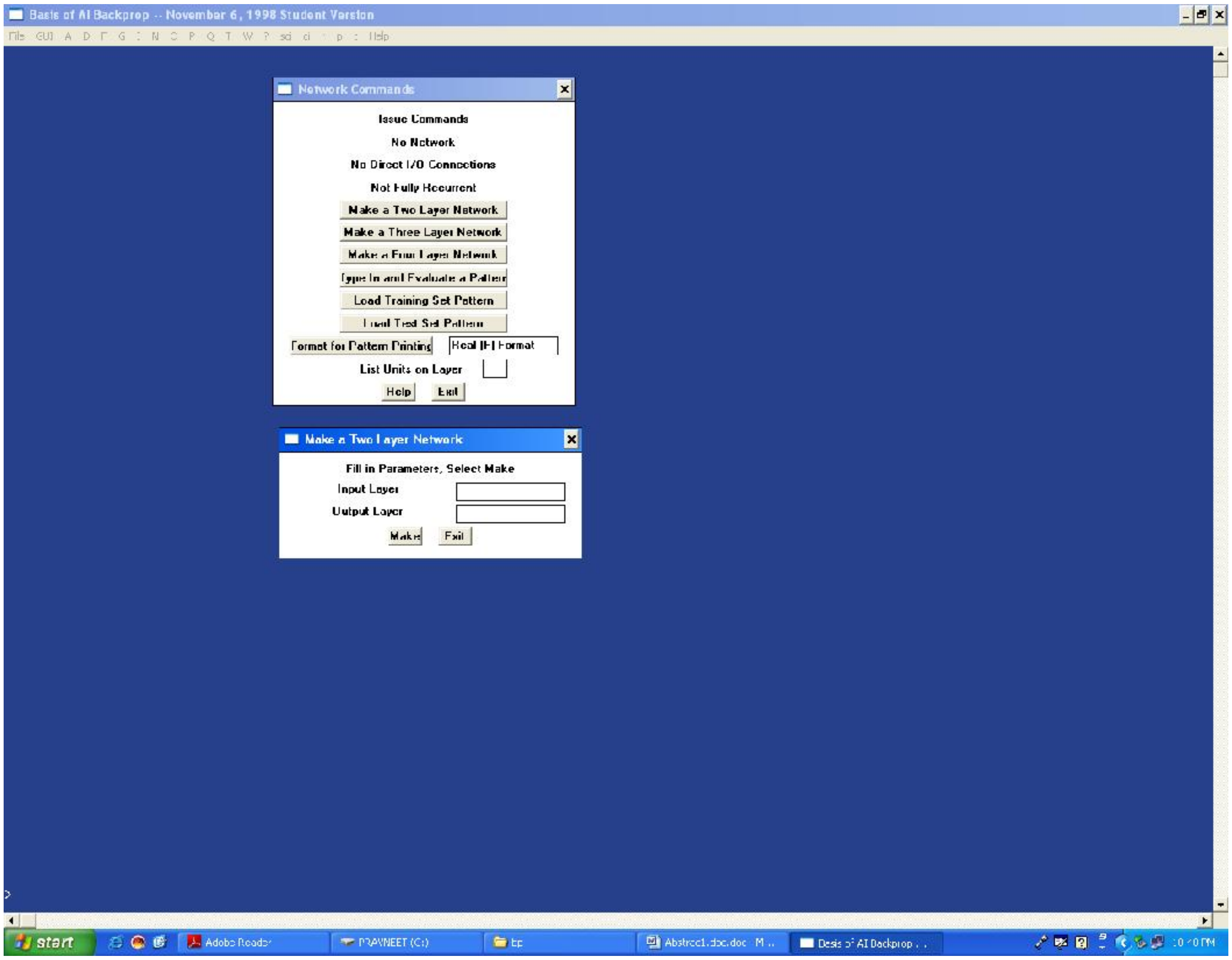
Basis of AI Backprop -- November 6, 1998 Student Version
Network Commands
Issue Commands
No Network
No Direct I/O Connections
Not Fully Recurrent
Make a Two Layer Network
Make a Three Layer Network
Make a Four Layer Network
Type In and Evaluate a Pattern
Load Training Set Pattern
Load Test Set Pattern
Format for Pattern Printing
Real (F) Format
List Units on Layer
Help
Exit
Make a Two Layer Network
Fill in Parameters, Select Make
Input Layer
Output Layer
Make
Exit

Figure 3: AI Back Propagation component

.