Novel design metrics to measure real time environment application design.

Mahmood Ahmed[1], Dr. M. Shoaib[1]

[1]Department of Computer Science & Engineering, University of Engineering & Technology Lahore Pakistan
mahmood@uet.edu.pk, shoaib_uet@hotmail.com

Abstract: In this paper we have defined a set of novel design metrics for measuring the design of specially real time environment applications. The aim of the defined new metrics set is to measure the design before handing over to the implementation team. The errors in the design can cost you money and time. Majority of the methodologies leave the task of managing the issue of task deadlines for software programmers in the implementation phase at the coding language stage. LCSF is measured for various methodologies. A non real time system design is also measured for design metrics. Results are plotted and discussed.
[Mahmood Ahmed, M. Shoaib. **Novel design metrics to measure real time environment application design.** Journal of American Science 2011;7(7):222-226]. (ISSN: 1545-1003). http://www.americanscience.org.

Keywords: Deadlines, Design Metrics, Real Time Systems, Design Measurement.

1 Introduction

Strict time limit on task deadlines is the most desired feature of real time [8] [26] . Because of this strict timing restriction real time system design is a challenge. Normally designers leave this task for the developer to cater, in the implementation phase [20] . Normally the methodologies for real time system do not handle inheritance of deadlines [3] [20] [21] and [23] . There is another property which gaining interest is usability. But the properties that lead an application to user friendliness for novices are often dissimilar from those preferred by expert users or the software designers [10] . Usability is also becoming demanding attribute but it is very not easy to quantify usability, user surveys may be useful in measuring usability [10] . Design quality is very important entity that should be taken into account in any software product [22] . This phase get 5 to 10 % of the entire effort but bulk (up to 80%) of whole effort consumed in correcting flawed design decisions [1] . There are many metrics proposed for capturing the quality of Object Oriented design [18] . Coupling and cohesion are most widely used metrics [11] . Coupling tells about how heavy is the coupling how modules are interdependent on each other is it a very spaghetti coupling or is it a very clean structured decomposition where you have a very low coupling and but there is collaboration through low coupling and very highly cohesive modules [7] . So each module is together but different modules are not very tightly coupled to each other. So this is what you would like to achieve high cohesion and low coupling [3] Class cohesion is associated to quality attributes of a software system as discovered in the empirical studies in [13] [14] [9] So independence or separation of concern is a very important principle of software engineering [6] [15] [5] There also many other design metrics that measure the design, but to measure real time system design no metrics have been defined up to best of our knowledge. In this paper we have studied various design methodologies [12] [16] and [28] . All of these methodologies are studied from the point of view of real time environment.

2 Main Contribution of this research

The main contribution of this research paper is the definition of eight new design metrics that have been related to the measurement of design of real time environment applications.

2.1 Verification using Proposed Design Metrics

We have defined the following eight new metrics for measuring a real time system design. Purpose is to measure the design before implementation.

2.2 Soft Deadline Class Factor

SDCF is defined as the ratio of the total no. of classes with soft deadlines to the overall sum of no. of classes with overridden, hard & soft deadlines.

$$SDCF = \sum_{i}^{n} \sum_{j}^{m} \frac{C_{s_{ij}}}{C_{s_{ij}} + C_{h_{ij}} + C_{o_{ij}}}$$

Where

n = Total no. modules constraint by timing restriction

m = Total no. classes per module constraint by timing restriction

$C_{s_{ij}}$ = ith Class in the jth module with soft deadlines.

$C_{h_{ij}}$ = ith Class in the jth module with hard deadlines.

$C_{o_{ij}}$ = ith Class in the jth module with overriden deadlines.

This factor concerned about the type of the real time system. If the value of this factor is high this means the system modules may be given less attention, as the error tolerance level for meeting timing requirements is more.

2.3 Hard Deadline Class Factor

HDCF is defined as the ratio of the total no. of classes with hard deadlines to the overall sum of no. of classes with overridden, hard & soft deadlines.

$$HDCF = \sum_{i}^{n} \sum_{j}^{m} \frac{C_{h_{ij}}}{C_{s_{ij}} + C_{h_{ij}} + C_{o_{ij}}}$$

Where

n = Total no. modules constraint by timing restriction

m = Total no. classes per module constraint by timing restriction

$C_{s_{ij}}$ = ith Class in the jth module with soft deadlines.

$C_{h_{ij}}$ = ith Class in the jth module with hard deadlines.

$C_{o_{ij}}$ = ith Class in the jth module with overriiden deadlines.

This factor is also concerned about the type of the real time system. If the value of this factor is high this means the system modules may be given more attention, as the error tolerance level for meeting timing requirements is low.

2.4 Overridden Deadline Class Factor

It is defined as the ratio of the classes having overridden deadlines to the total no. of classes having soft, hard, overridden deadlines.

$$ODCF = \sum_{i}^{n} \sum_{j}^{m} \frac{C_{o_{ij}}}{C_{s_{ij}} + C_{h_{ij}} + C_{o_{ij}}}$$

Where

n = Total no. modules constraint by timing restriction

m = Total no. classes per module constraint by timing restriction

$C_{s_{ij}}$ = ith Class in the jth module with soft deadlines.

$C_{h_{ij}}$ = ith Class in the jth module with hard deadlines.

$C_{o_{ij}}$ = ith Class in the jth module with overriiden deadlines.

This factor is the very vital as it reveals about timing related complexities lying in the modules with soaring ODCF value. These complexities are due to the inheritance of task timing requirements or task deadlines. The largest part of the attention must be centered on those modules with very high ODCF.

2.5 Soft Overriding Factor

SOF factor is defined as the ratio of the Overridden Deadline Class Factor (ODFC) to the Hard Deadline Class Factor (ODFC).

$$SOF = \frac{\sum_{o=1}^{n_o} C_o}{\sum_{h=1}^{n_h} C_h} = \frac{ODCF}{HDCF}$$

SOF gives information about the overall inclination the module. Either it is tilted in the direction of hard or the soft real time approach. If the value of SOF is less than one it means timing restriction on the task have to be met at any cost.

2.6 Message Exchange Factor

No. of exchanged messages considered per second between project partitions.

$$MEF = \frac{\sum_{i=1}^{n_s} m_i}{T}$$

If the value of this factor is high this means that more the underlying component must be critically analyzed. It also tells about how heavy is the coupling how modules are interdependent on each other is it a very spaghetti coupling or is it a very clean structured decomposition where you have a very low coupling. It is also desired that there is collaboration through low coupling and very highly cohesive modules. So each module is together but different modules are not very tightly coupled to each other. So this is what you would like to achieve high cohesion and low coupling [3] and at the same time you want to achieve separation of concerns and also collaboration.

2.7 Early Decomposition Factor.

The Early Decomposition Factor (EDF)

$$EDF = \frac{(\text{No. of partitions of Project})}{(\text{Project Stage No})} \times (\text{Message Exchange Factor})$$

Mathematically it can be symbolized as

$$EDF = \frac{N_p}{S_n} \times \frac{\sum_{i=1}^{n_s} m_i}{T}$$

It is also kept in mind that this metric has not as much of importance when only object oriented systems are under consideration. This early partition decision will have a serious impact on the system resources. If there are too much messages exchange between the various partitions of the projects then it

might be possible that resource consumption goes out of limits.

2.8 Deadline based Predictability Factor

The DPF Factor is defined as ratio of the total no of classes with soft deadline to the total no. of sub classes and added effect of total no. of multithreaded objects.

In Mathematical language it is symbolized as

$$DPF = \frac{\sum_{a=1}^{n_a} C_a}{\sum_{b=1}^{n_b} C_b} + \sum_{1}^{n} Obj_{mt}$$

Since multithreading enhances the predictability therefore in ideal situation the 1st factor must be than one and preferably should be close to zero and 2nd factor must be greater than one [27] .

2.9 Life Cycle Support Factor

Life Cycle Support Factor is defined as the ratio of number of phases having support for timing constraints/deadlines to the total no. of phases in the life cycle plus one.

$$LCSF = \frac{\text{No. of phases having deadline support}}{(\text{Total No. phases in the life cycle}) + 1}$$

Every methodology has support for timing constraints/deadlines Software life cycle in a number of phases. Ideally this factor should be equal to 1, this imply that the methodology bear support for timing constraints/deadlines in the entire life cycle further than the code release stage and into the code maintenance phase.

3 Case Study

Now as a case study we consider the following eleven different methodologies, as listed in the table no. 1, and computed the LCSF metric for them is also listed. We studied each methodology and searched which phases carry support for the task timing limitations/deadlines and computed the LCSF factor. To make things easier we considered the following specific no. of phases for Timing constrains/deadline support reflection.

1 specification
2 design
3 implementation
4 Testing & verification
5 deployment
6 maintenance

We come to conclusion that no methodology has full life cycle support. A three dimensional bar chart of the LCFS is shown in figure 7. It is evident that HRT-HOOD [19] [27] ROOM [25] & OCTOPUS [17] , ARTS [27] , have good support for the timing constrains/deadlines in different phases of the software development life cycle. JSD [2] [24] has support only in one phase. We searched many design documents for real time systems but unfortunately we were not able to get our desired real time system design examples that have considered the deadlines/timing constraints in the entire life cycle. So we are unable to compute the remaining metrics.

To compute design metrics we used the tool SDMetrics [4] which is a software design metrics measurement tool for especially for UML diagrams. UML is these days a preferred software design tool preferred by a good number of designers. SDMetrics itself is analyzed through their own tool.

Two important metrics DIT and WMC [11] have been measured. DIT tells about how deep is your inheritance tree? If it's too deep then it is considered to be complex in terms of say the behavior of different polymorphism possible or the behavior or the tracing required to understand that which method is going to be invoked through the inheritance tree.

DIT is plotted as a histogram in Figure 8. A high value of DIT means it is difficult to understand those classes inherit from a lot of classes. It is also found out that, classes having high value of DIT may not be correct specializations of every predecessor class **Error! Reference source not found.**.

For the module NumOp (Number of operations) in Figure 9 we have plotted a histogram for the metric WMC (Weighted method complexity) [11] , **Error! Reference source not found.**.

Table 1: Life Cycle Support Factor for different methodologies.

Methodology	Phases having Support	LCSF
JSD	1	0.142857
ATRS	3	0.428571
COBRA	1	0.142857
HOOD/PNO	1	0.142857
HRT-HOOD	4	0.571429
OCTOPUS	2	0.285714
OMTs	1	0.142857
OPNets	1	0.142857
ROOM	3	0.428571
RTO	1	0.142857
Transnet	1	0.142857

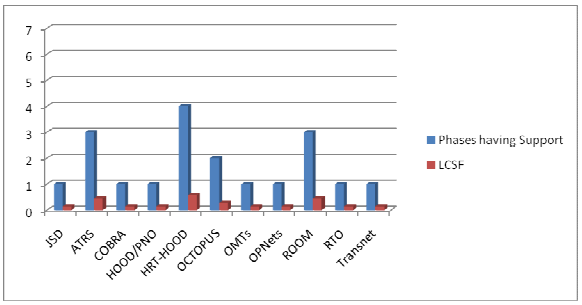
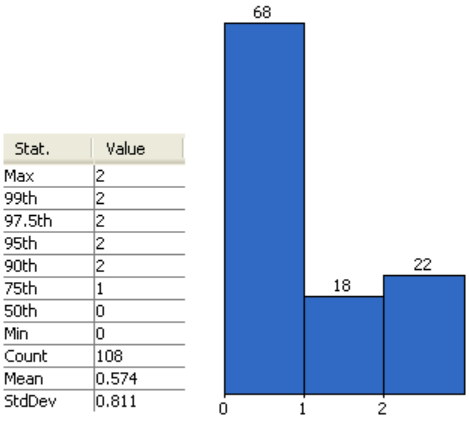
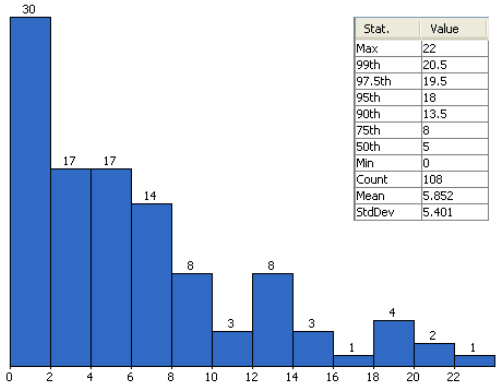


Figure 1: 3-D Plot of LCSF Factor



Stat.	Value
Max	2
99th	2
97.5th	2
95th	2
90th	2
75th	1
50th	0
Min	0
Count	108
Mean	0.574
StdDev	0.811

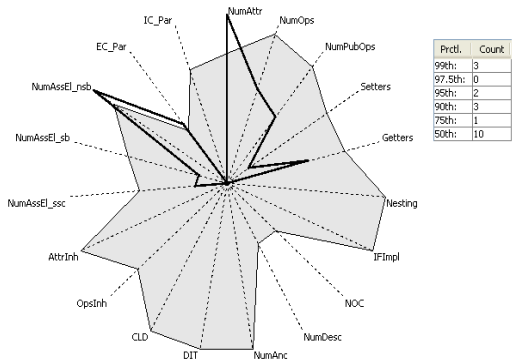
Figure 2: Histogram plot for DIT [4] .



Stat.	Value
Max	22
99th	20.5
97.5th	19.5
95th	18
90th	13.5
75th	8
50th	5
Min	0
Count	108
Mean	5.852
StdDev	5.401

Figure 3: Histogram plot of the NumOp (WMC) [4] .

The graph shown in figure 10 is a Kiviat diagram [4] , showing the values of all metrics for the module ExpressionNode. Each axis (or ray) of the Kiviat diagram represents one metric, as labeled in the diagram. The range of the metric is the scale of all axes: the lowest value is to be found in the center, the highest value at where the axis ends. The scaling of all the axes is linearly done. The table on the right side of the figure 9 shows, different percentiles values against the no. of metrics whose values go beyond the percentile for the selected module. Metrics with higher values point to inferior quality, a module design must be considered critical for which most of metric values for the module are in the higher percentiles (e.g., 90th, 95th).



Prctl.	Count
99th:	3
97.5th:	0
95th:	2
90th:	3
75th:	1
50th:	10

Figure 4: Kiviat Plot for the rule ExpressionNode module [4] .

Figure 11 is a discovery of the design rule violation by the SDMetrics tool at the package level design. Although the severity of the design rule violation is of medium and low level, still these discovered violations may help the designers to revisit their design and try to correct those errors early and before the implementation.

Name	Rule	Value	Category	Severity	Description
SDMetrics.com.sdmetrics.app	SDP2	SDMetrics.com.sdmetrics.metrics	Style	2-med	Package violates the Stable-Dependencies Principle (SDP).
SDMetrics.com.sdmetrics.app	SDP2	SDMetrics.com.sdmetrics.model	Style	2-med	Package violates the Stable-Dependencies Principle (SDP).
SDMetrics.com.sdmetrics.metrics	SDP2	SDMetrics.com.sdmetrics.model	Style	2-med	Package violates the Stable-Dependencies Principle (SDP).
SDMetrics.com.sdmetrics.metrics	SDP2	SDMetrics.com.sdmetrics.math	Style	2-med	Package violates the Stable-Dependencies Principle (SDP).
SDMetrics	Capitalization	SDMetrics	Naming	3-low	Package name has upper case letters.

Figure 5: Design rule violation at package level [4] .

4 Conclusion

In this paper we have defined a new metrics suite for the measurement of a real time environment application design. The metrics that have been defined are targeted to especially measure the real time system design, to identify the modules needing more detailed concentration. Those designs of those modules are revised again for the purpose of quality.

References

[1] S.R. Ragab, H.H. Ammar, "Object Oriented design Metrics and tools a Survey",7th International Conference on Informatics and Systems (INFOS), IEEE, Page(s) 1-7, 2010.

[2] Michael A. Jackson and John Cameron, "Jackson system development", http://jackson-system-development.co.tv/, last visited 2010.

[3] Sukainah Husein, Alan Oxley, "A Coupling and Cohesion Metrics Suite for Object-Oriented Software", International Conference on Computer Technology and Development, ICCTD '09, 2009, Page(s): 421 - 425.

[4] SDMetrics "The Software Design Metrics tool for the UML" http://www.sdmetrics.com accessed in May **2011**.
[5] Gui, G., Scott, D., "Measuring Software Component Reusability by Coupling and Cohesion Metrics," Journal of Computers, vol 4, no 9, pp 797-805, **2009**, Academy Publishers.
[6] Marcus, M., Poshyvanyk, D., "Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented System", IEEE Transactions on Software Engineering, Vol. 34, No. 2, **2008**.
[7] Linda Northrop, "Let's Teach Architecting High Quality Software", IEEE 19th Conference on Software Engineering Education & Training, **2006**, page(s) 5.
[8] Rob Williams, "Real-Time Systems Development", Butterworth-Heinemann publications, Elsevier, **2006**.
[9] Zhou, Z., Leung, H., "Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults", IEEE Trans. On Software Engineering, Vol. 32, No. 10, pp 771-789, **2006**.
[10] Phillip A. Laplante, "REAL-TIME SYSTEMS DESIGN AND ANALYSIS", third edition, Wiley IEEE press, **2004**.
[11] Norman E. Fenton, Shari Lawrence Pfleeger: Software Metrics, A Rigorous and Practical Approach, Thomson Learning, **2003**.
[12] Gjalt de long, "A UML-based design methodology for real-time and embedded systems," in Proceedings of Design, Automation and Test in Europe Conference and Exhibition, 2002, page(s) 776-779, **2002**.
[13] Briand L., et al., "Exploring the Relationships between design measures and software Quality in Object Oriented Systems", Journal of Systems and Software, Vol. 51, Issue 3, pp. 245-273, **2000**.
[14] Gyimothy, T., Ferenc, R., and Siket, I., "Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction," IEEE Transactions on Software Enineering, Vol. 31, Issue 10, pp. 897-910, 2005.
[15] Lee, J., Jung, S., Kim, S., Jang, W., and Ham, D: "Component Identification Method with Coupling and Cohesion," in Proceedings Eighth Asia-Pacific Software Eng. Conf., pp. 79-86, Dec. 2001.
[16] Timothy G. Woodcock, "Extensions to Real-Time Object-Oriented Software design Methodologies", PhD Thesis, FAU, **1996**.
[17] Jurgen Ziegler, Maher Awad, Juha Kuusela," Applying Object-Oriented Technology in Real-Time Systems with the OCTOPUS Method," in Proceedings of First IEEE International Conference on Engineering of Complex Computer Systems, November **1995,** page(s)306-309.
[18] Chidamber, S., Kemerer, C., "A Metrics Suite for Object Oriented Design". IEEE Transactions on Software Engineering, 1994, 20 (6), 476-493.
[19] A. Burns and A.J. Wellings, "A Structured Design Method for Hard Real-time Systems, Real time system, Springer, vol 6 no.1 page(s) 73-114, 1994.
[20] Hassan Gomaa, "A BEHAVIORAL ANALYSIS METHOD FOR REAL-TIME CONTROL SYSTEMS", Control Engineering Practice, vol. no. 1, **1993**, page(s) 33-72.
[21] M. Paludetto & S. Raymond, "A Methodology based on Objects and Petri Nets for Development of Real-Time Software", IEEE International Conference on Systems, Man & Cybernetics, **1993**, page(s) 705 to 710.
[22] P. Daponte, and et. Al, "Object Oriented Design of Measurement Systems," IEEE Transactions on Instrumentation and Measurement, December **1992**, vol. 41, no. 6, page(s) 874-880.
[23] K.M Sacha," Transnet Approach to Requirements Specification and Prototyping" in Proceedings of CompEuro '92. 'Computer Systems and Software, IEEE, **1992**, Page(s): 220 – 225.
[24] L. Rollo, "Jackson system development", IEE Colloquium on, Introduction to Software Design Methodologies, **1992,** Page(s): 3/1 – 313.
[25] Bran Selic, "ROOM: An Object-Oriented Methodology for Developing Real-Time Systems" 5th International Workshop on Computer-Aided Software Engineering. IEEE, **1992**, page(s) 230-240.
[26] Rajeev Alur and David L. Dill, "A theory of Timed Automata," in Real Time: Theory in Practice. Proceedings of REX Workshop, **1991,** page(s) 47-73.
[27] C. W. Mercer & H. Tokuda: "The ARTS Real-Time Object Model," 11th IEEE Real Time System Symposium, **1990,** page(s) 2-10.
[28] Michael A. Jackson, "A System Development Method", Tools and notions for program construction: An advanced course; Nice 1981; Cambridge University Press, **1982**, pages 1-25.

6/21/2011