A SURVEY OF COUPLING MEASUREMENT IN OBJECT ORIENTED SYSTEMS

V. S. Bidve [1] and Akhil Khare [2]
[1]Information Technology Department, M.Tech. (II), BVCOE, Pune, India
[2]Assistant Professor, Information Technology Department, BVCOE, Pune, India

ABSTRACT

Coupling measurement is a focus of study for many of the software professionals from last few years. Object-oriented programming is an efficient programming technique for programmers because of its features like reusability, data abstraction etc. Coupling is a very impotent factor in object-oriented programming for software quality measurement and used as predictors of software quality attributes such as fault proneness, impact analysis, ripple effects of changes, changeability etc. Many researchers have worked on coupling measurement and found various dimensions of coupling. Researchers have also worked on various aspects of coupling like static coupling measurement, dynamic coupling measurement, class level coupling, object level coupling etc. But still there is no standardization in the field of coupling measurement which is accepted worldwide. As a result of this it is very difficult to select any existing measure which obtain clear picture of state-of-art of coupling measurement for object-oriented systems. This paper analyses some terminologies of coupling measurement proposed earlier and discusses usefulness of each.

KEYWORDS: Coupling, measurement, object-oriented, dynamic, static

I. INTRODUCTION

Object oriented technology gaining significant importance in the field software development. To evaluate and maintain quality of object-oriented software there is a need to assess and analyse its design and implementation using appropriate measurement metrics [5]. A quality metrics should relate to external quality attributes of a design. External quality attributes include maintainability, reusability, error proneness, and understandability.

Based on observations and empirical studies, coupling has shown a direct impact on software quality [5]. In general, one of the goals of software designers is to keep the coupling in object-oriented system as low as possible. Classes of the system that are strongly coupled are most likely to be affected by changes and bugs from other classes. Such classes have more architectural importance; coupling measures helps in such happenings [2].

It is commonly observed in an object-oriented programming technique, inheritance and polymorphism is used more frequently. Static coupling measurement attributes are not sufficient to measure coupling due to inheritance and polymorphism.

As a result we focus on coupling measurement. We discuss various proposed dynamic coupling measurement metrics and their correlation quality attributes. We compare all measurement aspects and discuss in order to design uniform and standardized framework for coupling measurement.

The following section outlines the related work for object-oriented coupling metrics. Section 3 a detailed survey of existing coupling measures is carried out. In section 4 we provide comparative study of all the frameworks. Section 5 concludes the paper.

II. MOTIVATION

Object-oriented measurement has become popular area. There are large numbers of measures proposed for object oriented attributes such as coupling, inheritance, coherence. Also, there are several negative aspects regarding the manner in which the measures are developed and being developed. Coupling is a more complex software attribute in object oriented systems but our understanding about coupling measurement factors is poor. There is no standardization for expressing coupling measures; many measures are not operationally defined i.e. there is some ambiguity in their definitions. As a result, it is difficult to understand how different measures relate to one other and what their potential use is. All above aspects ultimately shapes to a need of detailed study of coupling measurement in object-oriented systems.

III. RELATED WORK

In this section we perform a detailed survey of existing coupling measurement frameworks in object-oriented systems.

3.1. Framework by Erik Arisholm [4]

The framework described by Erik considered following points regarding coupling measurement.

- Dynamic behavior of software can be precisely inferred from run time information.
- Static coupling measures may sometimes be inadequate when attempting to explain differences in changeability for object oriented design.

The author derived three dimensions of coupling.

1. Mapping : object or class
2. Direction: import or export
3. Strength: number of dynamic messages, distinct methods, or distinct classes.

The empirical evaluation of the proposed dynamic coupling measure consists of two parts.

- First to assess fundamental properties of the measure Second part evaluates whether the dynamic coupling measures can explain the change proneness of class.
- Erik used the concept of role-models for dynamic coupling measurement.
- Scenario: a specific sequence of interactions between the objects.
- Role: abstract representation of the functional responsibility of each object in a given scenario.

Object can have many roles because it may participate in many scenarios. The role-model reflects the dynamic coupling between the roles along three orthogonal dimensions: *direction*, *mapping* and *strength*.

- Direction of Coupling (Import and Export coupling): Dynamic import coupling counts the messages sent from a role, whereas dynamic export coupling counts the messages received.
- Mapping: Object-level and Class-level Coupling: *Object-level* coupling quantifies the extent to which messages are sent and received between the objects in the system. Dynamic, *class-level* coupling quantifies the extent of method dependencies between the classes implementing the methods of the caller object and the receiver object.
- Strength of Coupling: The strength of coupling quantifies the amount of association between the roles. It is of three types.
  1. *Number of dynamic messages.* Within a run-time session, to count the total number of times each message is sent from one role to another to implement a certain functional scenario.
  2. *Number of distinct method invocations.* To count the number of distinct method invocations between two roles.
  3. *Number of distinct classes.* To count the number of distinct classes.

Dynamic coupling is compared with static coupling and three important differences are Scope of Measurement, Dead code, Polymorphism. In all three measures dynamic coupling is considered more suitable than the static coupling. The relationship between dynamic coupling measures and the change proneness of the classes is explored by Erik and concluded that changes may prone to error.

3.2. Framework by R. Harrison, S. Counsell, R. Nithi [6]

This framework involves following points regarding coupling measurement.

1. Coupling between Object (CBO): Is a count of the number of classes to which a class is coupled. It counts each usage as a separate occurrence of coupling. This includes coupling via inheritance.
2. Number of Associations (NAS): is defined as the number of associations of each class. Counted from design documents. Counts repeated invocations as a single occurrence of coupling. This also includes coupling from inheritance.

Author considered that CBO is greater than NAS.
Three hypotheses related to coupling are investigated by the authors:

1. H1: As inter-class coupling increases, the understandability of a class decreases. This hypothesis is rejected by authors.
2. H2: As inter-class coupling increases, the total number of errors found in a class increases. This hypothesis is rejected by authors.
3. H3: As inter-class coupling increases, the error density of a class increases. This hypothesis is supported by authors.

To investigate these hypotheses author studied dependent variables such as

- Software Understandability (SU)
- Number of known errors(KE)
- Error per thousand non-comment source lines(KE/KNCSL)

Also, Coupling due to object as a parameter of methods and return type for a method is considered by authors.

3.3. Framework by Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson [5]

The authors have referred many papers and conclude following points regarding coupling measurement.
Two design metrics are considered by authors.

1. Static: can only calculate design time properties of an application.
2. Dynamic: used to calculate actual runtime properties of an application.

Two types of coupling is considered by authors

1. Class level coupling (CLC): Only invocations from one method to another are considered.
2. Object level coupling (OLC): The invocations from one method to another and frequency of invocations at run time is also considered.

Authors also considered that, there is correlation between the number of faults and complexity of system. Therefore, static complexity is used to assess the quality of software. To measure dynamic complexity metrics authors used ROOM design charts. Cyclomatic complexity, operation complexity is calculated from ROOM charts.
Authors explained export and import object coupling with context, description, formula and impact on design quality attributes.

1. Export object coupling (EOC): Measure is a percentage of number of messages sent from object A to object B with respect to total number of messages exchanged during the execution of some scenario.
2. Import Object coupling (IOC): Measure is a percentage of the number of messages received by object A *and* was sent by object B *with* respect to the total number of messages exchanged during the execution of some scenario.

3.4. Framework by Erik Arisholm, Lionel C. Briand, and Audun Føyen [1]

The authors described many significant dynamic coupling measures and highlights way in which they differ from static measures. Authors collected the measures using UML diagrams and accounted precisely for inheritance, polymorphism and dynamic binding.
Classification of dynamic coupling measures

1. Entity of measurement: The entity of measurement may be a class or an object.
2. Granularity: The granularity can be class level or object level.
3. Scope: The objects and classes are to be accounted for measurement.

The authors captured situations for import and export coupling.

1. Dynamic messages: Total number of distinct messages sent from one object to other objects and vice versa, within the scope considered.
2. Distinct method invocations: Number of distinct methods invoked by each method in each object.
3. Distinct classes: Number of distinct classes that a method in a given object uses.

The measures described by authors are summarized into table 1.

Table 1. Heading and text fonts.

Direction	Entity	Strength
Import coupling	Class	Dynamic messages
		Distinct Methods
		Distinct classes
	Object	Dynamic messages
		Distinct Methods
		Distinct classes
Export coupling	Class	Dynamic messages
		Distinct Methods
		Distinct classes
	Object	Dynamic messages
		Distinct Methods
		Distinct classes

The authors described much more regarding polymorphism and dynamic binding using the above measures than the others and concluded that coupling is one of the factors which affect change proneness.

3.5. Framework by Lionel C. Briand, John W. Daly, and Jurgen Wust [3]

The authors identified six criteria of dynamic coupling measures.

1. The type of connection: What items (attribute, method or class) constitutes coupling.
2. The locus of impact: To decide whether to count import or export coupling.
3. Granularity of the measure: The level of detail at which information is gathered, i.e. what components are to be measured and how exactly the connections are counted.
4. Stability of server: There are two categories of class stability defined. The first is unstable classes which are subject to modification or development (user defined) and stable classes which are not subject o to change (library).
5. Direct or indirect coupling: To decide whether to count direct or indirect coupling. For example, if a method *m1* invokes a method *m2*, which in turn invokes a method *m3*, we can say that *m1* indirectly invokes *m3*. Methods *m1* and *m3* are indirectly connected.
6. Inheritance: inheritance-based vs. non-inheritance-based coupling, and how to account for polymorphism, and how to assign attributes and methods to classes.

IV. COMPARISON AND DISCUSSION OF EXISTING FRAMEWORK

A comparison shows that there are differences in the manner in which coupling is addressed. There are differences due to different points of focus by different authors. It is observers that there is no uniformity in the measurement. The significant differences are discussed in the following subsection.

4.1 Type of coupling

In most of the frameworks the entity of measurement is a class or an object. But, the mechanisms that constitute coupling are different. Erik uses the concept of role-model that constitutes dynamic coupling. Harrison considers coupling due to any means between the classes including parameter passing to a method, return type of a method. Sherif and his team consider invocation from one method to another as a coupling. Lionel and his team consider connection due attributes, classes, and

method as a coupling. There are differences in the mechanism that constitute coupling with respect each framework.

4.2 Strength of coupling

It depends on type of connection and frequency of connection between two classes. Different types of coupling have different strengths. Erik counts the strength in terms of number of dynamic messages, number of distinct method invocations, and number of distinct classes involved in coupling. Harrison counts it in term of number of classes to which a class is coupled and counted each invocation separately. Sherif and his team counts strength in terms of method invocations and frequency of invocation. Lionel and his team have considered granularity instead of strength. Measure for strength of coupling is not uniform it also varies as per author.

4.3 Direction of coupling

The framework by Erik distinguishes import and export coupling. Import coupling counts messages sent from a role, whereas export coupling counts the messages received. Harrison has not discussed anything regarding direction of coupling. Sherif and his team have explained import and export coupling with respect to total number of messages exchanged during scenario. Lionel and his team explained it as a locus of impact in which import coupling is analyzed as client whereas export coupling as server in their roles. Definition of import and export coupling is also ambiguous. There is need to clearly define the concept of client and server class.

4.4 Direct and indirect coupling

Only Lionel and his team have discussed the concept of direct and indirect coupling. The observation is that many measures stated have used the direct coupling but some of measures have used indirect coupling also. Consideration of direct or indirect measure is again a matter of discussion. Many authors have not defined direct and indirect coupling. There is a need to clearly define these terms and to show measures under the terms.

4.5 Stability of server class

This point is unique to the framework by Lionel and his team. Using a stable class is better than using an unstable class, because modifications which could ripple through the system are less likely to occur. The remaining frameworks have not discussed this point; this is again an important point. How stability of server class is important that is also not discussed by many authors.

4.6 Inheritance

Inheritance is very important aspect of dynamic coupling. It is observed that there is a need to consider inheritance based coupling in measurement. Erik has considered polymorphism as part of dynamic coupling but not discussed inheritance. Harrison has considered coupling due to inheritance but not given any measures of inheritance and non-inheritance based coupling. Sherif also has used ROOM charts which shows coupling due to inheritance but explicitly it is not discussed. Erik and his team accounted inheritance, polymorphism and dynamic binding using various levels of granularity. Lionel and his team have differentiated various measures under the category of inheritance based and non-inheritance based coupling. There is no clear picture of how to use inheritance in coupling. Every author has different idea regarding inherence for coupling measurement.

4.7 Granularity

The granularity of the measure is the level of detail at which information is gathered. This is also an important point but not discussed by all the authors. Erik and Lionel have discussed the point but both have given a different explanation of the same. Very few authors have discussed this point. There is no clear understanding regarding granularity when we consider multilevel inheritance.

V. CONCLUSIONS

In this paper, we have studied five frameworks of dynamic coupling measurement for object-oriented systems. The motivation is to point out lack standardization and uniformity in the dynamic coupling

measurement. We have made comparison of all five frameworks with respect to total seven aspects. It is found that all frameworks differ in the definitions of measure, depth of measure, scope of measure and inclusion of points for coupling measurement. Many measures are ambiguous for e.g. type of coupling is an aspect in which cases which constitutes to a coupling are clearly not defined. Similarly, it is found with inheritance, strength of coupling and all other aspects of dynamic coupling measurement. Finally we come to the conclusion with following points.

- There is need of standardization in the field of dynamic coupling measurement
- Clear definition of every aspect of a measurement is needed
- Scope of measurement is needed to define for each measure
- Every measure must be empirically supported

These are problems we faced in the study of the various frameworks emerged as ideas to design a new framework model for dynamic coupling measurement.

REFERENCES

[1] Erik Arisholm, Lionel C. Briand, and Audun Føyen, "Dynamic Coupling Measurement for Object-Oriented Software," IEEE Transactions on Software Engineering, 30(8), 2004.
[2] Denys Poshyvanyk, Andrian Marcus, "The Conceptual Coupling Metrics for Object-Oriented Systems," ICSM '06 Proceedings of the 22nd IEEE International Conference on Software Maintenance, 2006.
[3] Lionel C. Briand, John W. Daly, and Jürgen Wüst, "A unified framework for coupling measurement in object-oriented systems," IEEE Transactions on Software Engineering, 25(1), 91-121, 2002.
[4] Erik Arisholm, "Dynamic Coupling Measures for Object-Oriented Software," IEEE Symposium on Software Metrics in *Proc.8*, 33-42, 2002.
[5] Sherif M. Yacoub, Hany H. Ammar, and Tom Robinson, "Dynamic Metrics for Object Oriented Designs," Software Metrics Symposium, Proceedings. 6, 50-61, 2002.
[6] R. Harrison, S. Counsell, R. Nithi, "Dynamic Metrics for Object Oriented Designs," Software Metrics Symposium, Proceedings. 5, 150-157, 2002.
[7] S.R. Chidamber, C.F. Kemerer, "Towards a Metrics Suite for Object Oriented design", in A. Paepcke, (ed.) *Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications*(OOPSLA'91), October 1991. Published in SIGPLAN Notices, 26 (11), 197-211, 1991.
[8] S.R. Chidamber, C.F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, 20 (6), 476-493, 1994.
[9] V. R. Basili, L. C. Briand, and W. L. Melo. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22(10):751{761, 1996.
[10] F. Abreu, M. Goulão, R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems", 5th International Conference on Software Quality, Austin, Texas, USA, October 1995.
[11] V. Basili, L. Briand, W. Melo, "Measuring the Impact of Reuse on Quality and Productivity in Object-Oriented systems", Technical Report, University of Maryland, Department of Computer Science, CSTR-3395, January 1995.
[12] V.R. Basili, L.C. Briand, W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators",IEEE Transactions on Software Engineering, 22 (10), 751-761, 1996.
[13] L. Briand, P. Devanbu, W. Melo, "An Investigation into Coupling Measures for C++", *Technical ReportISERN 96-08, IEEE ICSE '97*, Boston, USA, (to be published) May 1997.
[14] L. Briand, K. El Emam, S. Morasca, "Theoretical and Empirical Validation of Software Product Measures",Technical Report, Centre de Recherche Informatique de Montréal, 1995.
[15] L. Briand, S. Morasca, V. Basili, "Measuring and Assessing Maintainability at the End of High-Level Design", *IEEE Conference on Software Maintenance*, Montreal, Canada, September 1993.
[16] L. Briand, S. Morasca, V. Basili, "Defining and Validating High-Level Design Metrics", T*echnical Report,*University of Maryland, CS-TR 3301, 1994.
[17] L. Briand, S. Morasca, V. Basili, "Property-Based Software Engineering Measurement", *IEEE Transactions*of Software Engineering, 22 (1), 68-86, 1996.
[18] S.R. Chidamber, C.F. Kemerer, "Towards a Metrics Suite for Object Oriented design", in A. Paepcke,(ed.) Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'91), October 1991. Published in SIGPLAN Notices, 26 (11), 197-211, 1991.
[19] S.R. Chidamber, C.F. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions onSoftware Engineering*, 20 (6), 476-493, 1994.

[20] N.I. Churcher, M.J. Shepperd, "Comments on 'A Metrics Suite for Object-Oriented Design'", *IEEE*Transactions on Software Engineering, 21 (3), 263-265, 1995.
[21] N.I. Churcher, M.J. Shepperd, "Towards a Conceptual Framework for Object Oriented Software Metrics",Software Engineering Notes, 20 (2), 69-76, 1995.
[22] P. Coad, E. Yourdon, "Object-Oriented Analysis", *Prentice Hall*, second edition, 1991.
[23] P. Coad, E. Yourdon, "Object-Oriented Design", *Prentice Hall*, first edition, 1991.
[24] J. Eder, G. Kappel, M. Schrefl, "Coupling and Cohesion in Object-Oriented Systems", *Technical Report*,University of Klagenfurt, 1994.
[25] N. Fenton, "Software Metrics: A Rigorous Approach", *Chapman and Hall*, 1991.
[26] M. Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented systems", in *Proc. Int.*Symposium on Applied Corporate Computing, Monterrey, Mexico, October 1995.
[27] M. Hitz, B. Montazeri, "Chidamber & Kemerer's Metrics Suite: A Measurement Theory Perspective",IEEE Transactions on Software Engineering, 22 (4), 276-270, 1996.
[28] I. Jacobson, M. Christerson, P. Jonsson, G. Overgaard, "Object-Oriented Software Engineering: A Use Case Driven Approach", *ACM Press/Addison-Wesley*, Reading, MA, 1992.
[29] E. Arisholm, "Empirical Assessment of Changeability in Object-Oriented Software," PhD Thesis, Dept. of Informatics, Univ. of
[30] Oslo, ISSN 1510-7710, 2001.
[31] E. Arisholm, "Dynamic Coupling Measures for Object-Oriented Software," Proc. Eighth IEEE Symp. Software Metrics (METRICS '02),pp. 33-42, 2002.
[32] E. Arisholm, D.I.K. Sjøberg, and M. Jørgensen, "Assessing the Changeability of Two Object-Oriented Design Alternatives—
[33] A Controlled Experiment," Empirical Software Eng., vol. 6, no. 3,pp. 231-277, 2001.
[34] E. Arisholm, L.C. Briand, and A. Føyen, "Dynamic Coupling Measurement for Object-Oriented Software," Technical Report
[35] 2003-05, Simula Research Laboratory, http://www.simula.no/~erika, 2003.
[36] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language Users Guide. Addison-Wesley, 1998.
[37] L. Bratthall, E. Arisholm, and M. Jørgensen, "Program Understanding Behaviour During Estimation of Enhancement Effort on
Small Java Programs," Proc. Third Int'l Conf. Product Focused Software Process Improvement (PROFES 2001), 2001.
[38] L.C. Briand and J. Wuest, "Empirical Studies of Quality Models in Object-Oriented Systems," Advances in Computers, vol. 59, pp. 97-166, 2002.
[39] L.C. Briand and Y. Labiche, "A UML-Based Approach to System Testing," Software and Systems Modeling, vol. 1, no. 1, pp. 10-42,2002.
[40] L.C. Briand, J. Daly, and J. Wust, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems," Empirical Software Eng., vol. 3, no. 1, pp. 65-117, 1998.
[41] L.C. Briand, J.W. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems," IEEE Trans.
Software Eng., vol. 25, no. 1, pp. 91-121, 1999.
[42] L.C. Briand, J. Wust, and H. Lounis, "Using Coupling Measurement for Impact Analysis in Object-Oriented Systems," Proc. Int'l Conf. Software Maintenance (ICSM '99), pp. 475-482, 1999.
[43] F. BritoeAbreu, "The MOOD Metrics Set," Proc. ECOOP '95 Workshop Metrics, 1995.
[44] M. Cartwright and M. Shepperd, "An Empirical Investigation of an Object-Oriented Software System," IEEE Trans. Software
[45] Systems, vol. 26, no. 8, pp. 786-796, 2000.
[46] M.A. Chaumun, H. Kabaili, R.K. Keller, F. Lustman, and G. Saint-Denis, "Design Properties and Object-Oriented Software Changeability,"Proc. Fourth Euromicro Working Conf. Software Maintenance and Reeng., pp. 45-54, 2000.
[47] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476-493, 1994.
[48] S.R. Chidamber, D.P. Darcy, and C.F. Kemerer, "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," IEEE Trans. Software Eng., vol. 24, no. 8, pp. 629-637, 1998.
[49] I.S. Deligiannis, M. Shepperd, S. Webster, and M. Roumeliotis, "A Review of Experimental Investigations into Object-Oriented
[50] Technology," Empirical Software Eng., vol. 7, no. 3, pp. 193-232,2002.
[51] G. Dunteman, Principal Component Analysis. SAGE, 1989.
[52] K. El Emam, S. Benlarbi, N. Goel, and S.N. Rai, "The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics,"

[53] IEEE Trans. Software Eng., vol. 27, no. 7, pp. 630-650, 2001.
[54] R.J. Freund and W.J. Wilson, Regression Analysis: Statistical Modeling of a Response Variable. Academic Press, 1998.
[55] Jakarta, "The Apache Jakarta Project," http://jakarta.apache.org/,2003.
[56] Java.net, "Java Compiler Compiler (JavaCC)," https://javacc.dev.java.net/, 2003.
[57] H. Kabaili, R. Keller, and F. Lustman, "Cohesion as Changeability Indicator in Object-Oriented Systems," Proc. IEEE Conf. Software Maintenance and Reeng. (CSRM), pp. 39-46, 2001.
[58] A. Lakhotia and J.-C. Deprez, "Restructuring Functions with Low Cohesion," Proc. IEEE Working Conf. Reverse Eng. (WCRE), pp. 36-46, 1999.
[59] G. Myers, Software Reliability: Principles and Practices. Wiley, 1976.
[60] H. Sneed and A. Merey, "Automated Software Quality Assurance,"IEEE Trans. Software Eng., vol. 11, no. 9, pp. 909-916, 1985.
[61] M.M.T. Thwin and T.-S. Quah, "Application of Neural Networks for Software Quality Prediction Using Object-Oriented Metrics,"Proc. IEEE Int'l Conf. Software Maintenance (ICSM), 2003.

Authors

V. S. Bidve: Computer engineering from University of Aurangabad and the M. Tech pursuing from BVUCOE, Pune. He has nine years of teaching experience in Pune and Mumbai. He is now working as a lecturer in the Department of information technology SKNCOE, Pune.

A. R. Khare: Has completed bachelor degree in computer engineering from Bhopal University, India and M. Tech. from same University. Pursuing P. hD. In the field of computer engineering. Working as Assistance Professor in Information Technology Department of BVCOE, Pune. Having 10+ years of teaching experience. Working as a PG coordinator for IT department and guiding number of students for their project work and various academic activities.