

International Journal of Business Continuity and Risk Management

ISSN online: 1758-2172 - ISSN print: 1758-2164
<https://www.inderscience.com/ijbcrm>

Enhancing software design through dynamic metrics and entropy: a case study in cloud security for healthcare systems

Subhasish Mohapatra, Abhishek Roy, Bijay Kumar Paikaray

DOI: [10.1504/IJBCRM.2024.10065480](https://doi.org/10.1504/IJBCRM.2024.10065480)

Article History:

Received:	09 May 2024
Last revised:	26 June 2024
Accepted:	01 July 2024
Published online:	13 March 2025

Enhancing software design through dynamic metrics and entropy: a case study in cloud security for healthcare systems

Subhasish Mohapatra and Abhishek Roy

Department of Computer Science and Engineering,
Adamas University,
Kolkata, India
Email: mohapatra.subhasish@gmail.com
Email: dr.aroy@yahoo.com

Bijay Kumar Paikaray*

Department of Computer Science and Engineering,
Centre for Data Science,
Siksha 'O' Anusandhan (Deemed to be) University,
Odisha, India
Email: bijaypaikaray87@gmail.com
*Corresponding author

Abstract: The computational analysis through dynamic metrics and entropy potentially makes the system more robust before its real-time application. It can further bridge the missing link between component slicing and package restructuring. It ensures reliability in design and developers gain a methodical blueprint before coding. The result section elaborates on the cohesive analysis of each class. For a good software design, we need high cohesion and low coupling, so this adaptive analysis is achieved for this cloud-integrated healthcare model at the conceptual level of modelling. This analysis proves the flexibility of our proposed model. It systematically controls the ripple effect of changes in a model by coupling measures. The above concept further demonstrates how the structure of a unified modelling language (UML) class module within a model will explore the future scope of research work to provide an ultimate integrated cloud-based healthcare model to the end user.

Keywords: cloud-healthcare; cohesion; coupling; modularisation; healthcare model; entropy.

Reference to this paper should be made as follows: Mohapatra, S., Roy, A. and Paikaray, B.K. (2025) 'Enhancing software design through dynamic metrics and entropy: a case study in cloud security for healthcare systems', *Int. J. Business Continuity and Risk Management*, Vol. 15, No. 1, pp.1–17.

Biographical notes: Subhasish Mohapatra is currently pursuing PhD at Adamas University, Kolkata. He received his MTech in Computer Science Engineering from BPUT, Odisha, India in 2013. He has qualified for UGC-NET and presently working as an Assistant Professor in the Department CSE in Adamas University, Kolkata. He has an overall ten years of teaching experience. His research areas include AI, cloud simulation, and ML. He has 14+ research international publications in both journals and conferences.

Abhishek Roy is working as a Professor in the Department of Computer Science and Engineering, Adamas University, Kolkata, India. He has a total 60+ publications in international reputed conferences and journals. His research interests include cyber security, cloud computing, and artificial intelligence.

Bijay Kumar Paikaray currently serves as an Associate Professor at the Centre for Data Science, Department of Computer Science and Engineering, Siksha 'O' Anusandhan (Deemed to be) University, Odisha, India. He earned his PhD and MTech in Computer Science and Engineering from Centurion University, Odisha, India. With 15 years of robust academic experience, he is deeply committed to both teaching and research. In addition to teaching, he has authored more than 50 research publications, holds patents, and has authored books. His research interests include information security, machine learning, data visualisation, and IoT.

1 Introduction

All software-based application models should have sufficient level of conceptual and structural flexibility within itself to optimise its quality and performance during its real-time implementation. To design a service friendly software-based application model, it should be capable of accommodating all the customer requirements to perform well in the real-world scenario thereby tackling the real-world parameters and its anomalies (Tiwari and Rathore, 2018). This approach becomes very much essential for practical implementation of any type of electronic service delivery model where service seeker and service provider are connected to each other through virtual medium (i.e., internet). However, as an open access, cost effective and fastest electronic message communication medium, internet is also susceptible to illicit attempts of adversaries, which are usually mounted over any electronic message communication to jeopardise its integrity among its stakeholders (Amara and Ben Arfa Rabai, 2022). These risk factors become fatal during transmission of sensitive information of users (i.e., both service seeker and service provider) through the open communication channel like internet. On the other hand, in this current situation of global pandemic where people are forced to maintain the rules of social distancing to break the chain of coronavirus (COVID-19) transmission. Within the society, they are left with no other option but to rely mostly upon virtual medium (i.e., internet) for transmission of all types of messages among the connected users (Zhou et al., 2021). Due to massive spread of COVID19 throughout the globe, healthcare system has been adversely affected, where large number of patients and front-line workers (i.e., Doctors, Medical Staffs, Ambulance Drivers, Cremation staff, and others) have scrambled to infection thereby resulting in death due to lack of timely medical attention along with essential infrastructural support like medicine, ICU beds, oxygen supplies, steroids, etc. (Xiang et al., 2019). The rapid spread of COVID-19 has led to shortage of sufficient medical attention (i.e., Doctor, Nurse, Medicine, etc.), which has also adversely affected the medical treatment of non-covid patients. As this unfortunate scenario has been noticed throughout the globe, it has explored an enormous scope of future research work to tackle similar pandemic situations in an integrated and more comprehensive manner (Miholca, 2021).

Meanwhile, as in case of electronic message communication users are connected to each other through virtual mode thereby having limited scope for physical mode of identity verification, it is also essential to validate the identity of transacting parties, so as to ensure privacy, integrity, non-repudiation and authentication (PINA) of electronic transaction (Kalouptsoglou et al., 2022). It would be a better approach to provide all types of electronic services to patients in an integrated manner through a single window interface, which will play a vital role to uniquely identify the valid user (i.e., patient). The Entropy based empirical analysis can provide a complexity specific outcome for any object-oriented model. That can potentially more stable for future software development. Dynamic metric with addition to Entropy can bridge the missing link between software design and component slicing (Pattnaik et al., 2024). It makes the design behaviour more precise. To achieve this objective, authors have proposed an integrated cloud-based citizen centric healthcare model and analysed its robustness using coupling and cohesion analysis to ensure its highest level of quality and effectiveness with reference to SERVICE REQUEST of Patient (i.e., citizen) and corresponding SERVICE RESPONSE from medical service provider thereby considering the real-world parameters. To achieve this objective, authors have proposed a multipurpose electronic card (MEC) based cloud healthcare system which will help patients to avail medical facilities and bear its expenses through virtual medium (Hellhake et al., 2022). To ensure the highest level of efficiency of our proposed cloud-based healthcare model during SERVICE REQUEST and SERVICE RESPONSE, in this paper authors have performed its Coupling and Cohesion based analysis to simulate its operation in the real-world scenario. Recently authors have pointed source monitor tool that accumulate information such as number of calls and methods/class that exhibit reader something related to coupling and cohesion. It reflects case visibility of the software model by removing erosion of class, object and method interactions within itself (Himayat and Ahmad, 2023). Adhering to this concept, researchers perform UML component based design to assess cohesion and coupling analysis of any application design to simulate its real-world orientation and robustness.

In this research, Section – 2 states the background of this research work. Section – 3 states the proposed cloud healthcare system. Section – 4 explains the cohesion, coupling and entropy analysis of our proposed model. Section – 5 draws the conclusion and explores future scope of research work.

2 Background and research work

Software liableness is one of the most consequential attributes of software quality. It is often forecast using various application metrics that quantify interior quality characteristic such as cohesion and complexity. Predicting software reliability requires a constant attention to recommended software metrics. In this contexture, a set of four entropy-based metrics is proposed to watch this property. The various metrics that make up this set are calculated manually and have only been theoretically validated. Therefore, we head to provide an empirical approach to validate them as actionable pointer of software reliability. Consequently, we start by evaluating these metrics utilising a set of agenda extracted from real application projects. As re-modularisation of any conceptual model helps to achieve its futuristic application design (Gouda et al., 2024). To find the dynamic nature of any model cohesion and coupling play an important role in finding the

strength of the model. It captures the efficacy of the model at a conceptual stage before its real-time implementation.

A new delineation of weighted entropy is proffered to ameliorate assemblage account for small and different datasets. First, the weighted intra-class and inter-class entropies for the absolute and numerical contingent (on or upon) attributes are evolved utilising the precise definition of entropy, respectively. Second, weighted entropy is accustomed to compute cluster weights for mixed contingent (on or upon) attribution.

Modularity has been considered one of the most consequential characteristics of flourishing software design. It has a eloquent concussion on many external quality attributes such as reusability, maintainability, and understand ability. Therefore, bounce metrics to measure software modularity can be very useful. Howbeit several measures have been bounce to describe certain properties related to modularity; they do not describe the modularity of software in general. Complex networks use network models to summarise the internal building of complex systems, thereby giving a general way to analyses complex systems as a whole. In this document, we acquaint complex network theory to software engineering and use modularity, a widely believed used metric in the field of neighbourhood detection in complex network research, to measure measures the overall modularity of the software. First, the concrete software is depicted by a software network, a feature concatenation network (FCN), where methods and properties are nodes, joints between methods and properties are edges and weights. On the edges represent the coupling force. ‘High cohesion and low coupling’ is one of the most earth-shaking elements in object-oriented (OO) design (Mohapatra et al., 2021b). Cohesion is a clue of coupling within a module, while ‘coupling’ is a clue of coupling between modules.

When designing software, we often look for high cohesion (a coherent module) and low coupling (coupling between modules should be less), which facilitates the arrangement of a modular building in application. Modularity has been considered one of the biggest characteristics of software design, having an eloquent collision on much extraneous quality attributes such as reusability, maintainability, and concludes. When we talk about modularity, we usually mean the concept of interdependence inside modules and self-dependence between modules. Endangered auguring is a defense mechanism that grease the identification parade (and thus mitigation) of endangered beforehand enough in the software development life cycle. The scientific creationism neighbourhood has freshly been interested in developing Deep Learning reproductions that use text mining approach to predict the actuality of vulnerabilities in software components. Even so, there are also delving examining whether using statically extracted software benchmark can lead to acceptable vulnerability prediction models. In this paper, vulnerability prediction models based on software metrics and text mining are built and compared. The admixture of software metrics and primer tokens using a deep learning model is also tested to determine whether the combined model can lead to more correct vulnerability prediction (Mohanty et al., 2021a).

This paper has extrapolated various studies on cohesion and coupling analysis of software-based application models. Gradually it builds sufficient measures to obtain static and dynamic features of any application model. Our research work mainly focuses on cohesion and coupling analysis for object-oriented modelling (OOM) using unified modelling language (UML) diagrams. It emphasises on coupling measurement among the classes through OOM and its method invocations (Razafimahatratra et al., 2018). The authors applied an interdependence, i.e., an information theoretic approach broad- based

on the concept of UML class design system in evaluating coupling and cohesion early in the design phase.

This type of analysis will help to perform the reliability check of any software system in a broader spectrum, i.e., real-world parameters and its anomalies. The schematic diagram shown in Figure 1 elaborates how patients can initiate electronic transactions (i.e., SERVICE REQUEST) through a MEC, which acts as the single window interface to avail all types of electronic services through the virtual medium (i.e., internet). To validate this conceptual model authors have meticulously analysed its class diagrams which are shown through Figures 2, 3, 4, and 5 respectively (Mohapatra et al., 2021b; Roy and Karforma, 2013).

To trace the behavioural specification of any software-based application model, we have to focus on its cohesion and coupling metric analysis. The attributes and methods of each class define specific interactions within itself and other classes of the application model. As per object oriented (OO) concept, cohesion and coupling analysis encompasses the developmental taxonomy of any software model to control its ambiguities, so that it can perform efficiently in a real-world environment. It can be stated that cohesion helps to estimate the degree of strength within a class, which explains how an attribute of the same class is accessed by the member methods of that same class (Rahman et al., 2018).

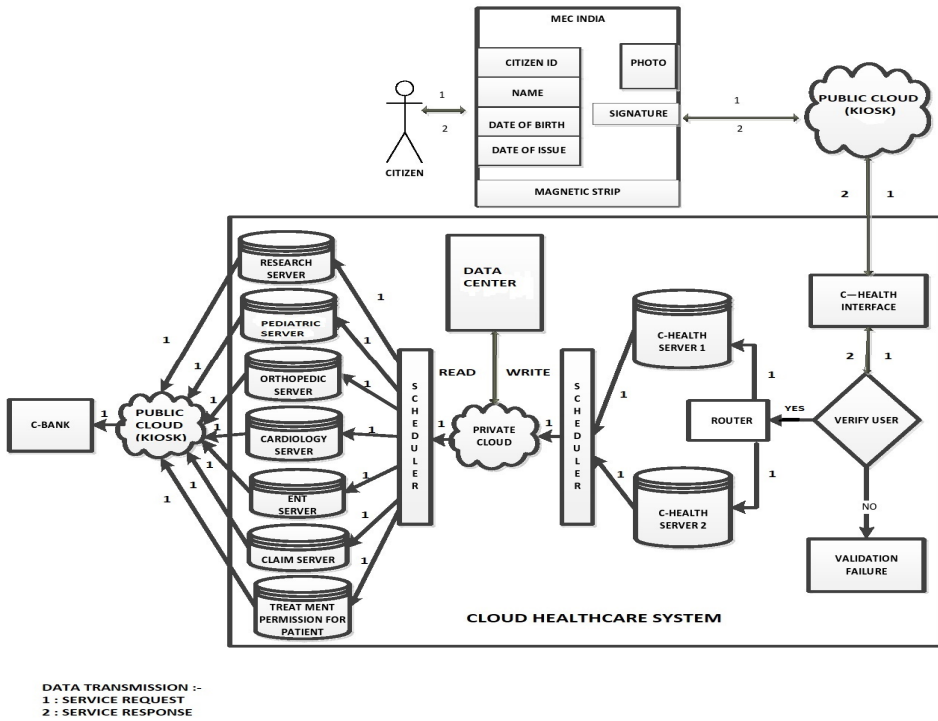
To have a well-defined class along with its data members (i.e., parameters or attributes) and member methods (i.e., behaviours), it should exhibit high cohesion within itself. If an attribute of a class is not counted thoroughly with the method of that class, it represents its weak functionality and exhibits low cohesion metrics, which is undesirable for any efficient software-based application model. In this paper, authors have performed cohesion and coupling analysis, whose formulas are mentioned below:

- 1 degree of cohesion (DCH) = number of attribute used (NAU)/total number of attribute (TNA)
- 2 Degree of coupling (DC) = message received coupling (MRC)/message passed coupling (MPC).

The dynamic nature of any application model should show high cohesion and low coupling to inculcate robustness within itself. Generally, clients adopt those application models that show maximum responsiveness towards elevated cohesion and stubby coupling. In the case of software engineering, the DC quantifies the grade of interdependence between its classes, where efficiency is measured with a low coupling metric. In the case of real-time adaptability, low coupling provides an intensive run time evaluation which is essential for its further enhancement (Iyyappan et al., 2023). Message interpretation of an application model can be measured using the number of messages transmitted and received from that class. High coupling degrades the performance or complexity of a model, because if one module is dependent on too many modules for the transmission of messages, then it may lead to cascading effects of changes within the system. Encapsulation can also be measured a simple priority for creating highly consistent classes and modules. By making data and code coexist, by definition you are creating more cohesive code. The advisement of certain 'behavioural' patterns in class design is clearly delineated as reduced coupling. To develop an efficient, robust, and realistic software-based application model, it should exhibit high cohesion and low

coupling metrics, which have been explained within our proposed citizen-centric single window-based cloud healthcare system (Mohapatra et al., 2021c).

Figure 1 Conceptual diagram of cloud healthcare system



3 Proposed cloud healthcare system

In this research paper Figure 1 shows the conceptual diagram of our proposed cloud healthcare system where patient (i.e., citizen) uses a single window based smart card interface namely, MEC to send SERVICE REQUEST and receive corresponding SERVICE RESPONSE from the Medical Service Provider through the public cloud (i.e., public kiosk). Path – 1 of Figure 1 represents the SERVICE REQUEST of patient (i.e., citizen) and Path – 2 of Figure 1 represents the SERVICE RESPONSE for patient (i.e., citizen), both of which are transmitted to the end user through public kiosk (i.e., public cloud). As our proposed cloud healthcare model requires frequent interaction between all of its stakeholders, it needs accurate integration of all of its modules. Figure 2 shows the modular integration of our proposed cloud healthcare system through various medical units like ENT, cardiology, orthopaedic, paediatric, etc. To perform OOM of any software system, UML diagrams like class diagram, inheritance diagram, use case diagram, etc. are used to represent their structure and assess their quality depending upon the real-world parameters and their capabilities to handle the run-time anomalies. In this paper, we have used a class diagram to show the static structure of primary stakeholders of our proposed cloud healthcare system (Mohapatra et al., 2022). The class diagram of

the patient, kiosk, cloud healthcare server, scheduler, bank, and insurance company for medical policy claim and ENT are shown in Figures 3, 4, 5, 6, 7, 8, and 9 respectively. Furthermore, the cohesion and coupling metrics obtained from these classes are stated in Table 1 and Table 2 (Candela et al., 2016).

Figure 2 Module interaction diagram

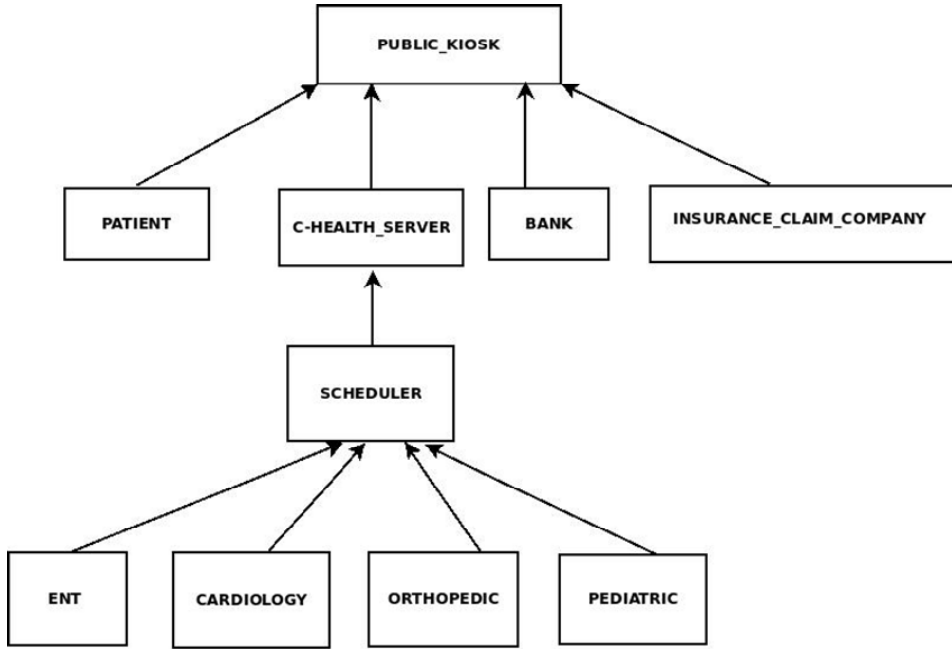
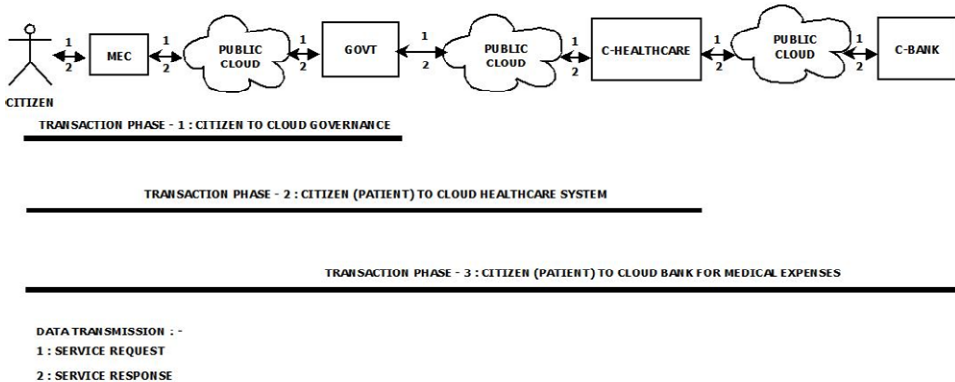


Figure 3 Service flow between class



Here in this section, the authors state an empirical study of the proposed model shown in Figure 1. The classes used in this model are analysed in Table 1 and Table 2. Hence it captures which class is strongly cohesive and evaluates their modular interlinking. It is a novel work performed by authors that moves around all modular classes involved in this model which is shown in Figure 2 (Mohapatra et al., 2021b; Roy and Karforma, 2013;

Mohapatra and Anand, 2021; Mohapatra and Parija, 2021). As a part of future enhancement, coupling and cohesion analysis of OOM provides spectral analysis of class for addition or deletion of new methods within itself. This paper will attempt to explore the impact of cohesion and coupling between the classes of our proposed application model. The relative strength and interdependency of this model are stated in Tables 1 and 2 respectively.

4 Cohesion and coupling analysis

Eventually, cohesion and coupling are two contrasting goals of any software-based application model. The activities of any application software are distributed among its component modules, which communicate within its classes to execute their operations. To develop a real-world oriented efficient application model, it should exhibit good cohesion and coupling analysis during its simulation through OOM. It is very essential to find the complexity of a model (Mallick et al., 2024). There may be various software paradigms but developers must rely upon the design which offers better complexity analysis of class in terms of interlinking among the classes. UML diagrams like class diagrams are considered an appropriate visual tool to perform this type of class analysis (Divakaran et al., 2022). It exhibits syntax visualisation of the class and behaviour study of methods among classes for its qualitative analysis. To perform the cohesion and coupling analysis of our proposed cloud healthcare system, we have used the following classes which are discussed below (Candela et al., 2016; Jin et al., 2022; Rahman et al., 2018; Mohapatra et al., 2023a).

Cohesion and coupling analysis in this section is elaborated in Table 1 and Table 2.

In order to perform cohesion analysis of our proposed cloud healthcare system, the static structure of its primary stakeholders like *PUBLIC_KIOSK*, *PATIENT*, *C_HEALTHSERVER*, *SCHEDULER*, *BANK*, *POLICY_CLAIM* and *ENT* are shown through their class diagrams (i.e., Figures 3, 4, 5, 6, 7, 8 and 9) respectively.

4.1 *PUBLIC_KIOSK CLASS*

This class is the base class which redirects messages to *PATIENT*, *C_HEALTHSERVER*, *BANK*, *POLICY_CLAIM*. DC for this class is zero. All six methods in *PUBLIC_KIOSK* class redirects to respective child class, at the same time it accumulates no message request from any of the class. DCH analysis for this class is 0.6 from the TNA three attribute, i.e., OTP, Service_request and Flag has been used within class.

Public attributes of *PUBLIC_KIOSK* class are:

- 1 KIOSK_ID
- 2 OTP
- 3 KIOSK_ADDRESS
- 4 FLAG.

Protected attribute of *PUBLIC_KIOSK* class is SERVICE_REQUEST

The public methods of *PUBLIC_KIOSK* class are:

```

1  send_mecno()
2  send_otp()
3  send_username( )
4  send_password()
5  receive_ack( )
6  send_service_request( ).

```

4.2 PATIENT CLASS

As shown in Figure 3, the attributes and methods of PATIENT class are mentioned as below

In PATIENT class there are seven private attributes, which are:

```

1  MEC_NO
2  USER NAME
3  PASSWORD
4  DOB v. AGE
5  ADDRESS vii.PHONE.

```

The protected attribute is SERVICE_REQUEST. The public methods of this class are mentioned as:

```

1  enter_mec_no( )
2  enter_otp( )
3  enter_user_name( )
4  enter_password( )
5  enter_service_request( ).

```

Based on the above-mentioned components of the PATIENT class, its DC has been calculated as 0 (i.e., zero). The patient class has received messages through two method calls from PUBLIC_KIOSK class, i.e., send_user_name() and send_otp(). The patient class has not passed any message to any other class. The DCH for this class is generated as 0.62, The Number of the attributes used for this class is five (i.e., MECNO, USERNAME, PASSWORD, OTP, SERVICE REQUEST). As shown in Figure 4, the Patient class has a total eight attributes. (3) C_HEALTHSERVER CLASS

As shown in Figure 5, (i.e., C_HEALTHSERVER) has two private attributes namely, SERVERIP and SERVERDNS. The public attributes for C_HEALTH SERVER are mentioned as:

```

1  receive_mecno( )
2  validate_mecno( )
3  send_username( )

```

```

4  send_password( )
5  receive_ack( )
6  send_service_request( )

```

4.3 *C_HEALTHSERVER*

This class accumulates three service requests from PUBLIC_KIOSK class, i.e., send_username (), send_service_request(), receive_ack(). Class C_HEALTH SERVER communicates all six methods with the child Class SCHEDULER. The DC for this class is 0.5. DCH shows 0.5 for this class as the TNAs (i.e., here) and NAU (i.e., 1 here which is SERVERDNS) (Priyalakshmi and Latha, 2018; Mohapatra et al., 2021c, 2023b).

4.4 *SCHEDULER CLASS*

There are three protected attributes in SCHEDULER class, which are mentioned below:

```

1  DEPARTMENTID
2  SCHEDULE
3  SERVICEREQUEST

```

SCHEDULER class has two public methods, which are:

```

1  publish_schedule ( )
2  forward_service_request ( )

```

The role of the SCHEDULER class streamlines the activities from ENT, Cardiology, Orthopaedic, and Paediatric. The DC for this class is 0.5. There are two methods refer to Figure 6 from this one method forward_service_request () is received from C_HEALTHSERVER Class and two methods are passed to the respective child class. Cohesive analysis for this class is found as 0.66. The NAU is two (SCHEDULE, SERVICEREQUEST), TNA is three.

4.5 *BANK CLASS*

This class has four private attributes, which are:

```

1  CLAIM_CITIZEN_MEC_NO
2  NON_CLAIM_CITIZEN_MEC_NO
3  MEC_NO_CITIZEN_EPAYMENT
4  MEC_NO_CITIZEN_OFFLINE

```

It receives transactions from the PUBLIC_KIOSK class (receive_claim_citizen_mecno ()). This class passes seven methods to the POLICY_CLAIM class. The DC for this class is 0.125. This is one of the highly cohesive classes in this model. The DCH for this class is one. The model is given in Figure 1. In Section 1 the description of the above-mentioned system initiates successful class module descriptions for Figure 2.

4.6 POLICY_CLAIM CLASS

This class has four protected attributes, which are:

- 1 CLAIM_CITIZEN_MEC_NO
- 2 POLICY_COMPANY
- 3 STATUS_POLICY
- 4 AMOUNT.

This class validates receiving and passing message from BANK and PATIENT class. The DC for this class is 0.5 as message received for this class is one, i.e., forward_service_request () from PUBLIC_KIOSK Class. It is one of the highly cohesive classes as four attributes are connected within the class.

4.7 ENT CLASS

Just like ENT Class there may be several other similar classes like Cardiology, Orthopaedic, Paediatric, etc. These classes are taking responses from SCHEDULER Class. DC for this class is one MRC for this class is shared with SCHEDULER class two methods publish_report (), execute_service_request (). MPC for this class is with patient using two methods, i.e., send_ack () and ack_dept_id ()

5 Result and discussion

To validate the proposed model authors created Table 3 which represents the comprehensive metric value of individual UML classes. There are seven UML classes for our comprehensive Cloud healthcare model which are shown in each row of Table 3. For precision analysis, the metric value for each class is constructed column-wise from M1 to M6 (Mohapatra et al., 2021b; Roy and Karforma, 2013; Mohapatra and Anand, 2021).

M1 (dependency), M2 (association between class), M3 (aggregation), M4 (generalisation), M5 (class method), M6 (class attributes) respectively. In Table 4 experimental result shows the normalised value that is calculated as per formula

$$R_{ij} = a_{ij} / \text{Max}(a_{ij}) \quad i \in N, j \in M \quad (1)$$

To get column normalisation one standardised approach is taken as below

$$R_{ij} = R_{ij} / \sum_{i=1}^n R_{ij} \quad i \in N, j \in M \quad (2)$$

For future modelling entropy plays an important aspect in complexity analysis. As entropy is defining model behaviour for probabilistic bench marking in terms of consistency. In general term entropy measures the randomness of any system. In this research work author develop six metrics in addition to cohesion and coupling for seven UML classes that represents rows and column value in Table 3. The formula for entropy is given as below

$$E_j = -1 / \ln N \left(\sum_{i=1}^n R_{ij} \ln R_{ij} \right) \quad i \in N, j \in M \quad (3)$$

To generalise the dependency, association, aggregation, generalisation, class method, and attributes of classes for model-1 authors have articulated the class diagram and inheritance diagram which are shown in Figure 2 to Figure 9. The authors have given a detailed outline of metric association in Table 3. PATIENT class shows dependency with C-HEALTHCARE and BANK class and at the same time exhibits normal association with POLICY CLAIM class. C-HEALTHCARE class shows dependency with the SCHEDULER class. BANK class exhibits one dependency and one normal association with PATIENT and POLICY_CLAIM class. POLICY_CLAIM class shows normal association and aggregation with PATIENT and PUBLIC_KIOSK class. The attribute and method of each class module is depicted in each UML class diagram for further reference (Mohapatra et al., 2021b; Mohapatra and Anand, 2021; Priyalakshmi and Latha, 2018).

Table 1 D  g  e de coh  sion

CLASS NAME	TNA	DCH = NAU/TNA
PUBLIC_KIOSK	05	03/05 = 0.6
PATIENT	08	05/08 = 0.62
C_HEALTHSERVER	02	01/02 = 0.5
SCHEDULER	03	02/03 = 0.66
BANK	04	04/04 = 1
POLICY_CLAIM	04	04/04 = 1
ENT	02	02/02 = 1

Table 2 Degree of coupling

CLASS	MRC/MPC	DC
PUBLIC_KIOSK	0/6	0
PATIENT	2/0	0
C_HEALTHSERVER	3/6	0.5
SCHEDULER	1/2	0.5
BANK	1/8	0.125
POLICY_CLAIM	1/2	0.5
ENT	2/2	1

The empirical analysis of each entropy value lies between 0 to 1. The nature of entropy significantly impacts upon the consistency and completeness of our model. The calculated entropy of each metric is given in Figure 12. The empirical study of entropy decides the chaos in the conceptual model as per Figure 11 only E5 (entropy -5) = 0.94. So, this entropy value needs future functional decomposition. For this model five out of six fall below 0.8 which is taken as the threshold of entropy for this model. Less entropy decides less chaos and more consistency. So the model achieves 83% consistency which is a good sign for future implementation

$$5(\text{Stable entropy value}) / 6(\text{Total entropy}) * (100) = 83 \quad (4)$$

Table 3 Metric value of each class

CLASS NAME	U1	U2	U3	U4	U5	U6
PUBLIC_KIOSK	00	04	00	00	06	05
PATIENT	02	01	00	01	05	08
C_HEALTHSERVER	01	00	01	01	06	02
SCHEDULER	00	00	01	01	02	03
BANK	01	01	01	00	08	04
POLICY_CLAIM	00	01	01	00	02	04
ENT	00	00	01	00	04	02

This section highlights behaviour analysis of each class in terms of cohesion and coupling. It has included more features for better interoperability. It allows scalability support for hosting dynamic module of any realistic application model. Table 1 include the DCH of each class that show case specific strength of each module. formula for DCH = NAU/TNA

Table 4 Normalised metric value of each class

CLASS NAME	U1	U2	U3	U4	U5	U6
PUBLIC_KIOSK	0	1	0	0	0.75	0.62
PATIENT	1	0.25	0	1	0.62	1
C_HEALTHSERVER	0.5	0	1	1	0.75	0.25
SCHEDULER	0	0	1	1	0.25	0.375
BANK	0.5	0.25	1	0	1	0.5
POLICY_CLAIM	0	0.25	1	0	0.25	0.5
ENT	0	0	1	0	0.5	0.25

Table 5 Column normalised metric value of each class

CLASS NAME	U1	U2	U3	U4	U5	U6
PUBLIC_KIOSK	0	0.57	0	0	0.18	0.17
PATIENT	0.5	0.14	0	0.33	0.15	0.02
C_HEALTHSERVER	0.75	0	0.20	0.33	0.18	0.07
SCHEDULER	0	0	0.20	0.33	0.06	0.10
BANK	0.25	0.14	0.20	0	0.24	0.14
POLICY_CLAIM	0	0.14	0.20	0	0.06	0.14
ENT	0	0	0.20	0	0.12	0.07

Based on the model shown in Figure 1, the coupling of classes is stated in Table 2. formula for DC = MRC/MPC (Mohapatra and Parija, 2021; Candela et al., 2016; Jin et al., 2022).

A highly cohesive class is less dependent on another class, whereas a strongly coupled class depends on another class and hence shifting of new method or deletion of an existing method becomes difficult concerning the interlinked classes. The experimental study is elaborated on below. From the table researcher has analysed that the DC is 85%. Out of a total of seven classes, six classes have achieved low coupling.

As it is a pretty good analysis for further real-time implementation. This model has set 0.6 for the cohesion and coupling threshold. If any class has achieved a DCH greater than this, then this class is considered a highly cohesive class. As for coupling analysis DC is set to 0.6 if any class has achieved a DC greater than this then it is declared as a highly coupled class. rather author emphasises on low coupling class which is useful for better software design. The effective analysis of each class based on cohesion and coupling is given in Figure 10 and Figure 11.

Figure 10 Degree of cohesion of class (see online version for colours)

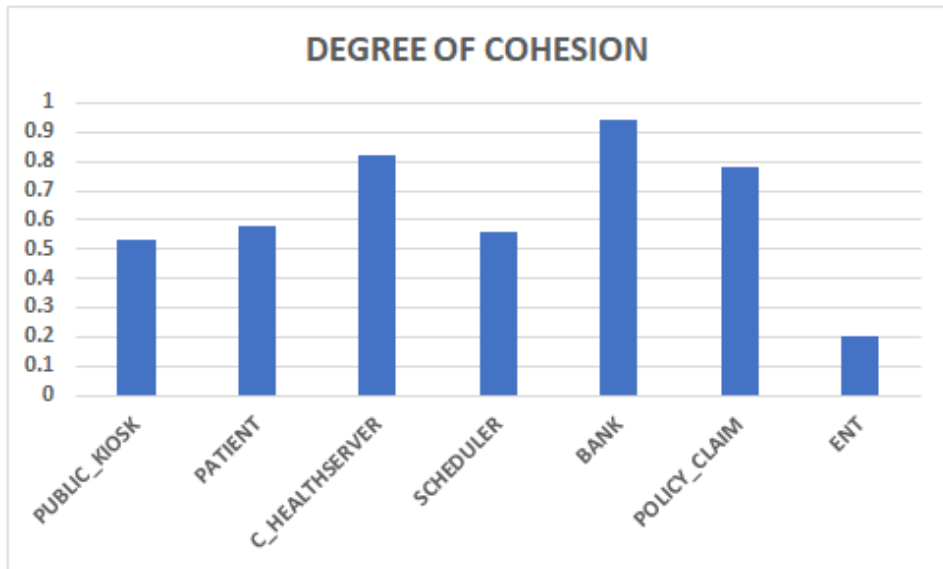


Figure 11 Degree of coupling of classes (see online version for colours)

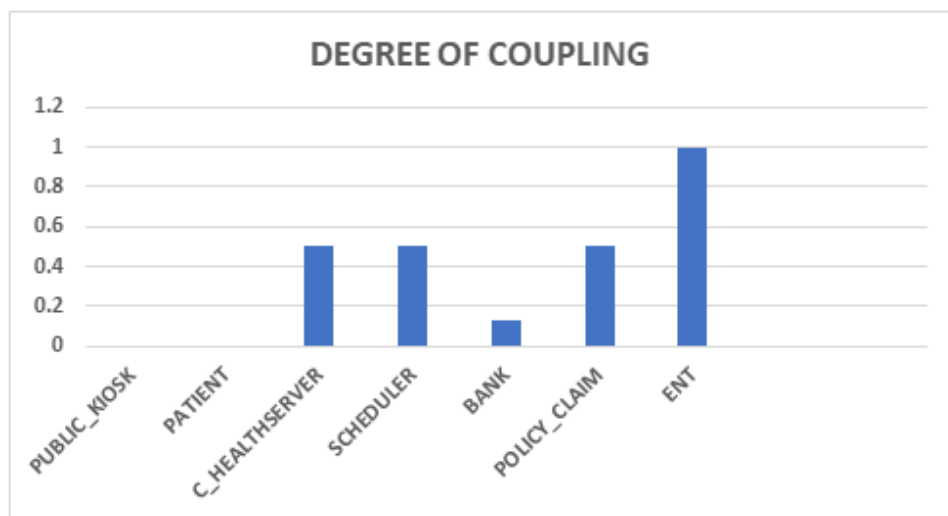
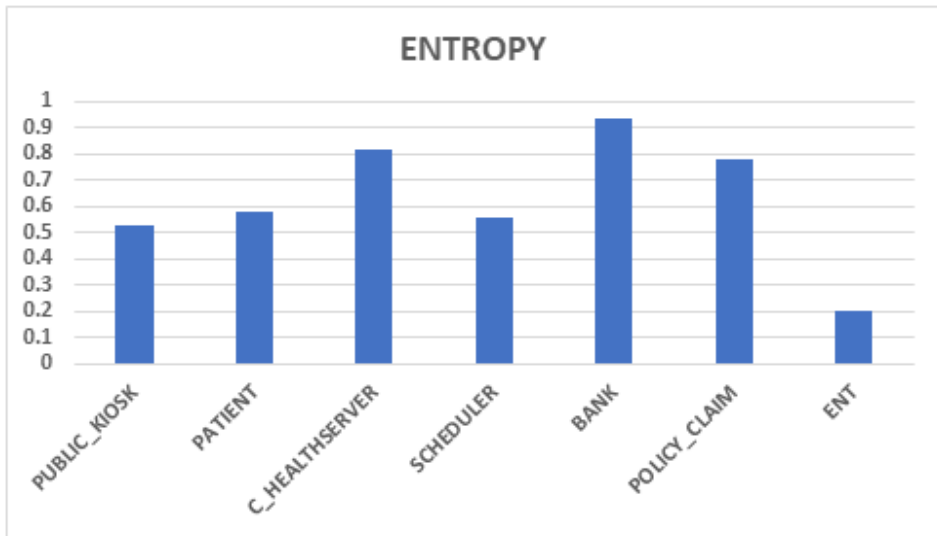


Figure 12 Entropy of metric (see online version for colours)

6 Conclusions

The concept of the proposed research work facilitates the quick identification of strongly cohesive and weakly coupled classes within our citizen-centric cloud healthcare system. It stimulates the refactoring of class functionality to achieve robustness. Consequently, cohesion and coupling can promote better software design. Preliminary identification through this can capture design flaws for any conceptual model before implementation. This approach substantiates momentum with the help of UML class attributes and methods. With the help of this analysis, the researcher can regulate a quantitative summary of cohesion and coupling. In account of modularisation of software systems, this paper focuses on the subjectivity of class cohesion and the objective of coupling that implicitly provides a proper balance between classes of the software system. Enhancing model behaviour author not only considers cohesion and coupling but also at the same time entropy-based metric calculation of each model component is carried out to achieve consistency modelling here in this research outcome authors have analysed the coupling and cohesion of each class by using the message interaction method it adheres to intra and inter-class. At the same time, it can assist any information technology expert in deciding the complexity measurement of an object-oriented model. Entropy control envisages a pivotal role in polishing up precision in OOM technique. Conversely, meticulous analysis of entropy can improve software quality and potent.

References

- Amara, D. and Ben Arfa Rabai, L. (2022) ‘Empirical validation of entropy-based redundancy metrics as reliability indicators using fault-proneness attribute and complexity metrics’, in *International Conference on Evaluation of Novel Approaches to Software Engineering*, April, pp.93–115, Springer Nature Switzerland, Cham.
- Candela, I., Bavota, G., Russo, B. and Oliveto, R. (2016) ‘Using cohesion and coupling for software remodularization: is it enough?’, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 25, No. 3, pp.1–28.
- Divakaran, J., Prashanth, S. K., Mohammad, G.B., Shitharth, D., Mohanty, S.N., Arvind, C., Srihari, K., Yasir Abdullah R. and Sundramurthy, V.P. (2022) ‘Improved handover authentication in fifth-generation communication networks using fuzzy evolutionary optimisation with nanocore elements in mobile healthcare applications’, *Journal of Healthcare Engineering*, Vol. 2022, No. 1, pp.2500377.
- Gouda, B.S., Giri, P.K., Das, S., Panigrahi, T. and Paikaray, B.K. (2024) ‘Predicting and diagnosing self-intermittent faults in a dynamic distributed attack on wireless sensor network’, *International Journal of Business Continuity and Risk Management*, Vol. 14, No. 2, pp.182–208.
- Hellhake, D., Bogner, J., Schmid, T. and Wagner, S. (2022) ‘Towards using coupling measures to guide black-box integration testing in component-based systems’, *Software Testing, Verification and Reliability*, Vol. 32, No. 4, p.e1811.
- Himayat, S. and Ahmad, D.J. (2023) ‘Prediction systems for process understand ability and software metrics’, SSRN 4514290.
- Iyyappan, M., Kumar, A., Ahmad, S., Jha, S., Alouffi, B. and Alharbi, A.A. (2023) *Component Selection Framework of Cohesion and Coupling Metrics*, Vol. 44, No. 1.
- Jin, N., Yang, N., Sharif, S.M.F. and Li, R. (2022) ‘Changes in knowledge coupling and innovation performance: the moderation effect of network cohesion’, *Journal of Business and Industrial Marketing*, Vol. 37, No. 11, pp.2380–2395.
- Kalouptoglou, I., Siavvas, M., Kehagias, D., Chatzigeorgiou, A. and Ampatzoglou, A. (2022) ‘Examining the capacity of text mining and software metrics in vulnerability prediction’, *Entropy*, Vol. 24, No. 5, p.651.
- Mallick, C., Giri, P.K. and Paikaray, B.K. (2024) ‘The privacy-preserving high-dimensional synthetic data generation and evaluation in the healthcare domain’, in *Applications of Synthetic High Dimensional Data*, pp.162–178, IGI Global.
- Miholca, D.L. (2021) ‘New conceptual cohesion metrics: assessment for software defect prediction’, in *2021 23rd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, December, pp.163–170, IEEE.
- Mohapatra, S. and Anand, K. (2021) ‘An expert system to implement symptom analysis in healthcare’, *Integration of Cloud Computing with Internet of Things: Foundations, Analytics, and Applications*, pp.57–69.
- Mohapatra, S. and Parija, S. (2021) ‘IoT-based modeling of electronic healthcare system through connected environment’, in *Progress in Advanced Computing and Intelligent Engineering 2021*, pp.423–431, Springer, Singapore.
- Mohanty, S.N., Nalinipriya, G., Jena, O.P. and Sarkar, A. (Eds.) (2021a) *Machine Learning For Healthcare Applications*, John Wiley and Sons, Hoboken, NJ.
- Mohapatra, S., Paul, K. and Roy, A. (2021b) ‘Object-oriented modeling of cloud healthcare system through connected environment’, in *Applications of Internet of Things*, pp.151–164, Springer, Singapore.
- Mohapatra, S., Satpathy, S. and Paul, D. (2021c) ‘Data-driven symptom analysis and location prediction model for clinical health data processing and knowledgebase development for COVID-19’, in *Applications of Artificial Intelligence in COVID-19*, pp.99–117, Springer, Singapore.

- Mohapatra, S., Satpathy, S. and Mohanty, S.N. (2022) 'A comparative knowledge base development for cancerous cell detection based on deep learning and fuzzy computer vision approach', *Multimedia Tools and Applications*, Vol. 81, No. 17, pp.24799–24814.
- Mohapatra, S., Satpathy, S. and Paikaray, B.K. (2023a) 'A machine learning approach to assist prediction of Alzheimer's disease with convolutional neural network', *International Journal of Bioinformatics Research and Applications*, Vol. 19, No. 2, pp.141–150.
- Mohapatra, S., Siddique, M., Paikaray, B.K. and Riyazbanu, S. (2023b) 'Automated invasive cervical cancer disease detection at early stage through deep learning', *International Journal of Bioinformatics Research and Applications*, Vol. 19, No. 4, pp.306–326.
- Pattnaik, L., Satpathy, S., Paikaray, B.K. and Swain, P.K. (2024) 'DDoS analysis using machine learning: survey, issues, and future directions', *International Journal of Business Continuity and Risk Management*, Vol. 14, No. 1, pp.57–76.
- Priyalakshmi, G. and Latha, R. (2018) 'Evaluation of software reusability based on coupling and cohesion', *International Journal of Software Engineering and Knowledge Engineering*, Vol. 28, No. 10, pp.1455–1485.
- Rahman, M.M., Riyadh, R.R., Khaled, S.M., Satter, A. and Rahman, M.R. (2018) 'MMRUC3: A recommendation approach of move method refactoring using coupling, cohesion, and contextual similarity to enhance software design', *Software: Practice and Experience*, Vol. 48, No. 9, pp.1560–1587.
- Razafimahatratra, H., Mahatody, T., Razafimandimby, J.P. and Andriamahazomandimby, T.M.B. (2018) 'Automatic detection of the cohesion type in the UML sequence diagram', in *2018 19th International Carpathian Control Conference (ICCC)*, May, pp.462–467, IEEE.
- Roy, A. and Karforma, S. (2013) 'Coupling and cohesion analysis for implementation of authentication in E-Governance', in *ACEEE Conference Proceedings Series*, Vol. 2, pp.544–554.
- Tiwari, S. and Rathore, S.S. (2018) 'Coupling and cohesion metrics for object-oriented software: a systematic mapping study', in *Proceedings of the 11th Innovations in Software Engineering Conference*, February, pp.1–11.
- Xiang, Y., Pan, W., Jiang, H., Zhu, Y. and Li, H. (2019) 'Measuring software modularity based on software networks', *Entropy*, Vol. 21, No. 4, p.344.
- Zhou, J., Chen, K. and Liu, J. (2021) 'A clustering algorithm based on the weighted entropy of conditional attributes for mixed data', *Concurrency and Computation: Practice and Experience*, Vol. 33, No. 17, p.e6293.