Measuring Coupling in Microservices Using COSMIC Measurement Method

ROBERTO PEDRAZA-COELLO, FRANCISCO VALDÉS-SOUTO

ROBERTO PEDRAZA-COELLO

Master Student in the Computer Science and Engineering Postgradute program of the UNAM. With a bachelor degree in Software Engineering.

FRANCISCO VALDÉS SOUTO

COSMIC President. Researcher-professor at the Sciences Faculty of the UNAM. With a Doctorate in Software Engineering specializing in Software Measurement and Estimation, Computer Engineer from the Engineering Faculty of the UNAM.

More than 20 years of experience in critical software development, Founder of SPINGERE, the first Mexican company specialized in software measurement, estimation and evaluation, Founder of the Mexican Association of Software Metrics (AMMS).

Microservices Architectural Style (MAS)

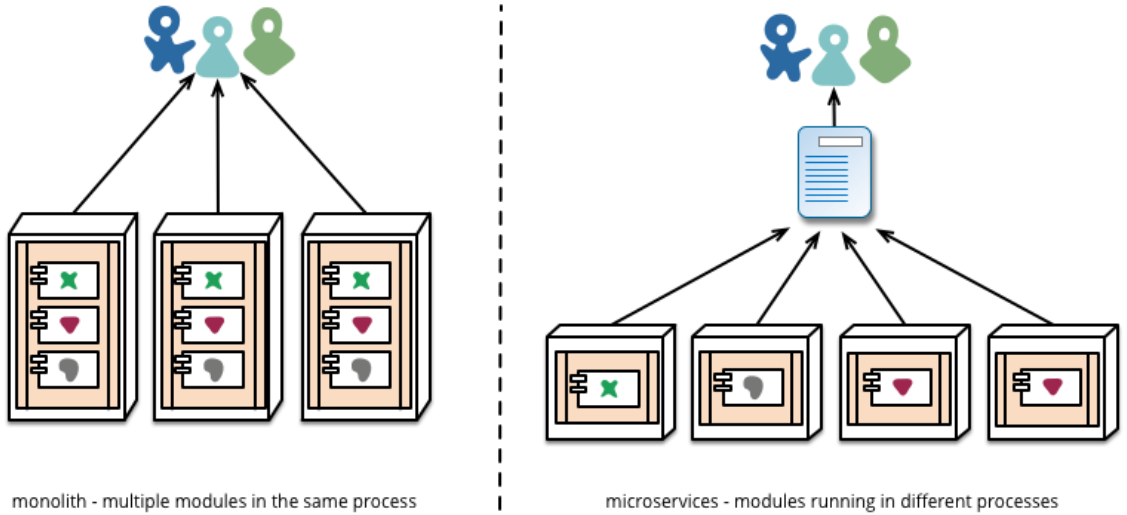
Is one of the latest trends in software development companies. Its main idea is to develop an application as a set of small services. Each one of these services is called a microservice [1].

Benefits/Characteristics

Each microservice is implemented and operated as a small an independent system [1].

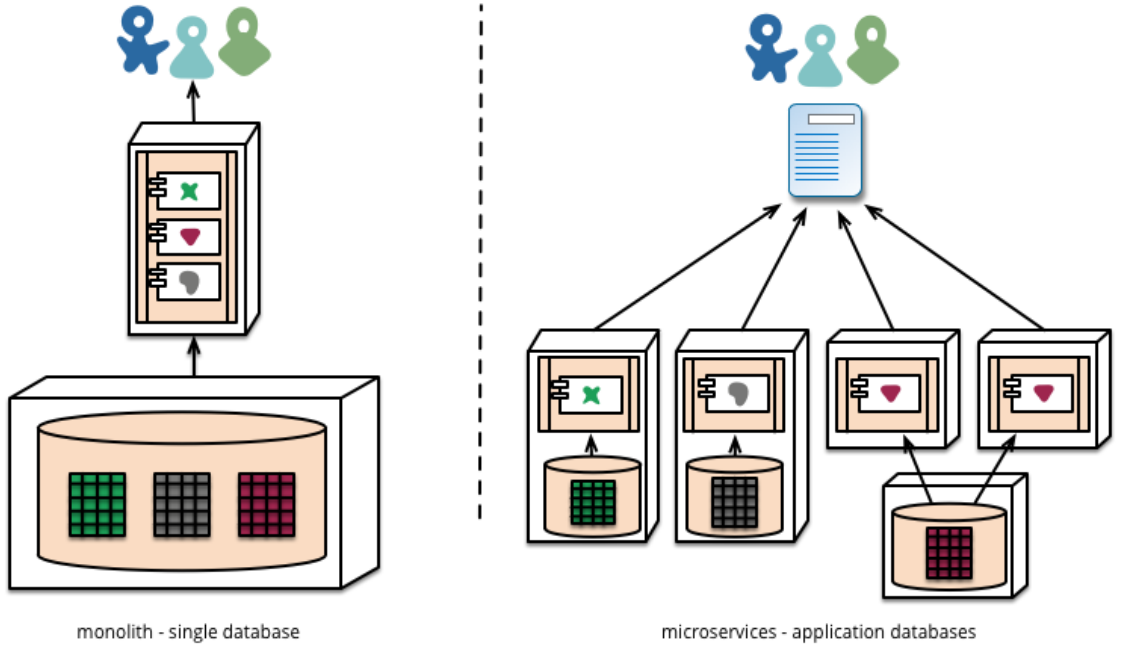
Offers access to its internal functionality and data through a well-defined network interface [1].

Increases the software development process agility because each microservice is an independent unit of development, deployment, operations, versioning, and scaling [1].

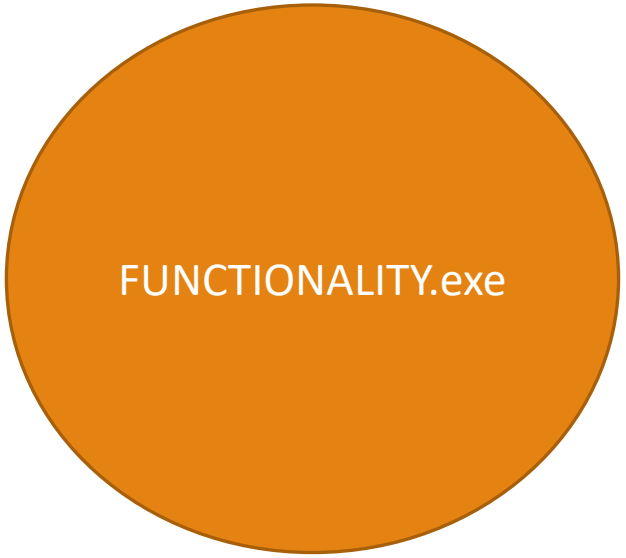


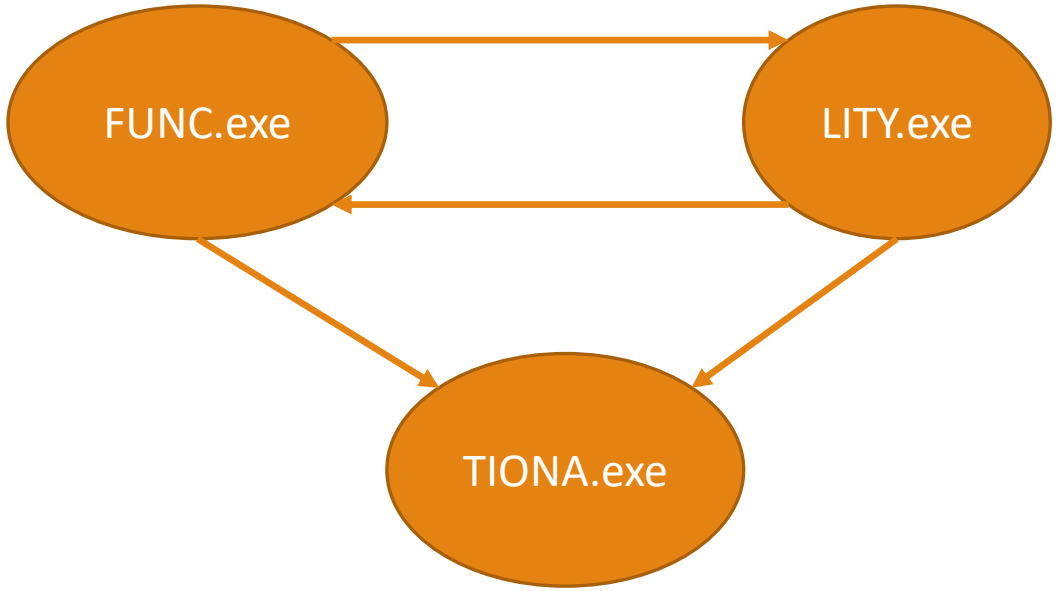
MICROSERVICES

"The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies. [2]"



MONOLITHIC VS MICROSERVICES





COUPLING

In the OOP, low coupling is achieved when each object depends on little or nothing of other objects.

If the microservices depend a lot on each other, then the coupling is high. If the microservices depend little or none on each other, the coupling is low.

It's recognized that if the coupling is high, then technological and management problems can arise.

Coupling Measurements:

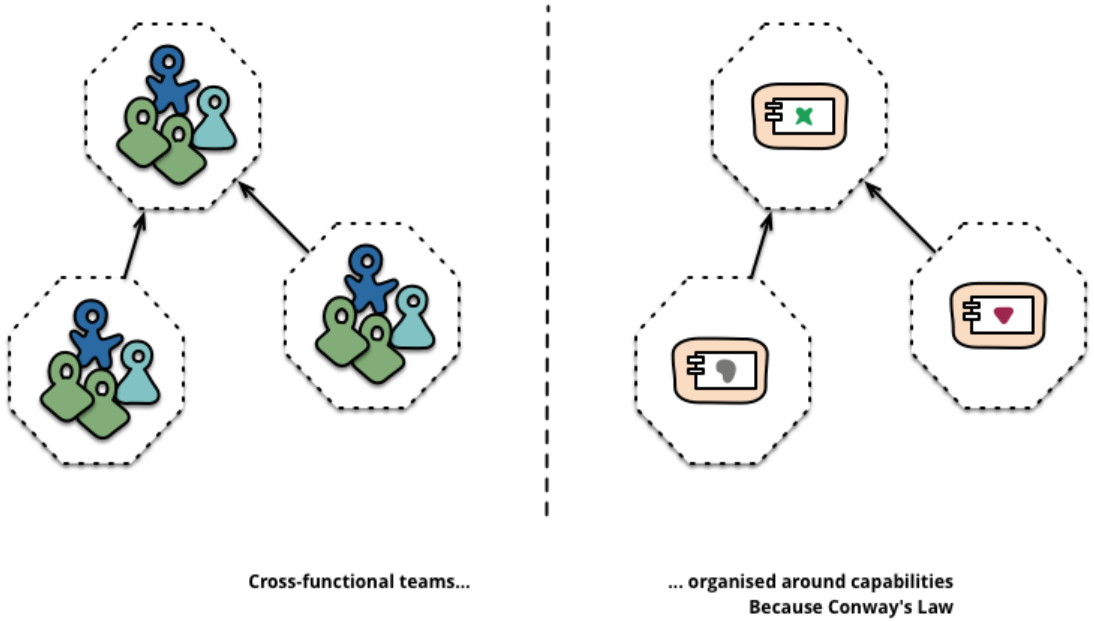
Coupling Between Object classes (CBO) by Chidamber and Kemerer [3].

Measuring Coupling and Cohesion by Allen and Khoshgoftaar [4].

Number of messages, number of distinct method invocations, and distinct classes, by Arisholm et al [5].

Dynamic method invocations, dynamic messages, and distinct classes by Lavazza et al [6].

WHY TO MEASURE COUPLING BETWEEN MICROSERVICES?



Measuring the coupling between microservices in the early stages of the software development cycle could help to quantify the interdependency that exists between different microservices. This could improve the software architect's decision making in terms of avoiding, as far as possible:

- High interdependency between teams
- Latency and network traffic.

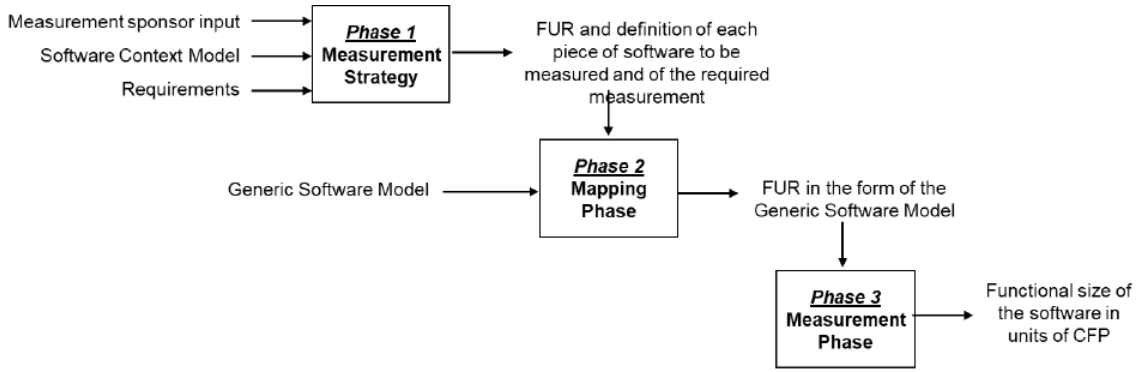
COSMIC – FUNCTIONAL SIZE MEASUREMENT METHOD (ISO/IEC 19761)

Currently, the only type of software measurement with international standards adopted by the ISO is the measurement of functional size [7].

The COSMIC Functional Size Measurement method complies with the metrology requirements [7].

The COSMIC sizing methodology is applicable to any type of software (https://cosmic-sizing.org)

A course to learn to measure software with COSMIC takes from 24 hours to 40 hours



COUPLING METRIC BASED IN COSMIC

Use the COSMIC concepts to measure the coupling between microservices to ensure that the measurement is consistent, repeatable and comparable.

The COSMIC measurement manual explains how to measure software by counting the data movements in each of the functional processes. There are certain rules of when a certain data movement is considered for the measurement and when not. The proposed coupling metric uses the same rules that COSMIC uses.

OUR DEFINITION OF MICROSERVICES COUPLING

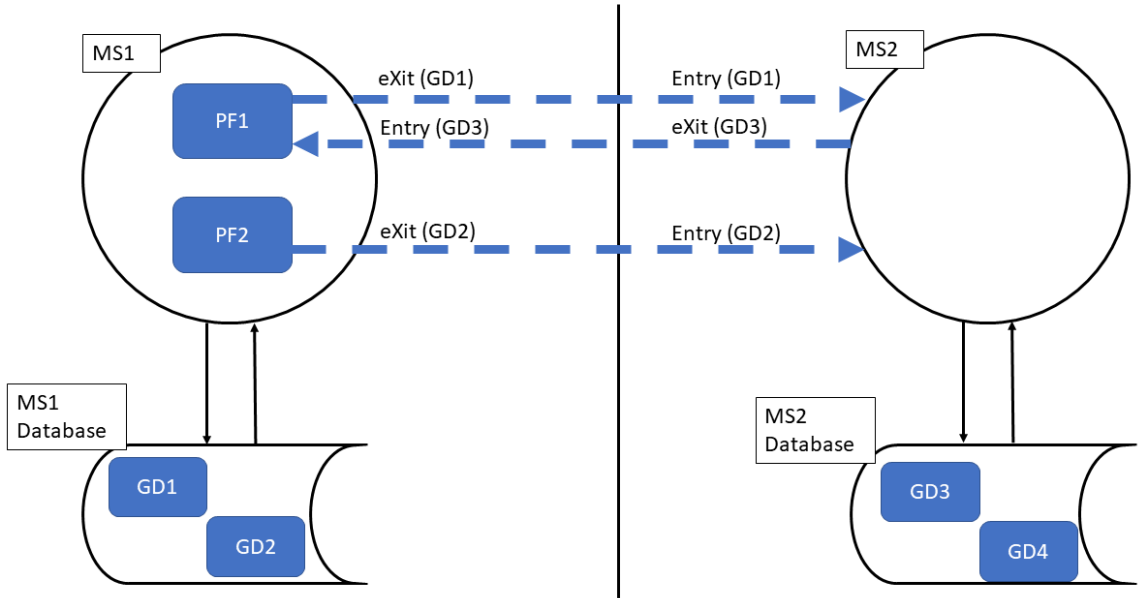
When said that MS1 is coupled to MS2, it is meant that MS1 depends on MS2 to complete a certain portion of its own functionality. However, it doesn't necessarily mean the same in inverse mode

It can be said that MS1 is coupled to MS2 when MS1 starts a request/response communication with MS2. A good analogy is to imagine a client-server architecture where MS1 is the client and MS2 is the server, the client depends on the server, not the other way around.

In this sense, it is understood that there is a unilateral or bilateral coupling. A relationship between two microservices can be hierarchical (exclusively one service uses the services of another), or bidirectional (both services use services of the other service).

BASIC EXAMPLE

To measure how coupled is MS1 to MS2 we need to count, for each functional process of MS1, the number of data movements that are exchanged between MS1 and MS2, for all the cases where MS1 start the communication with MS2.



PF1: 1 (eXit) + 1 (Entry) = 2 CFP

PF2: 1 (eXit) = 1 CFP

How coupled is MS1 to MS2?

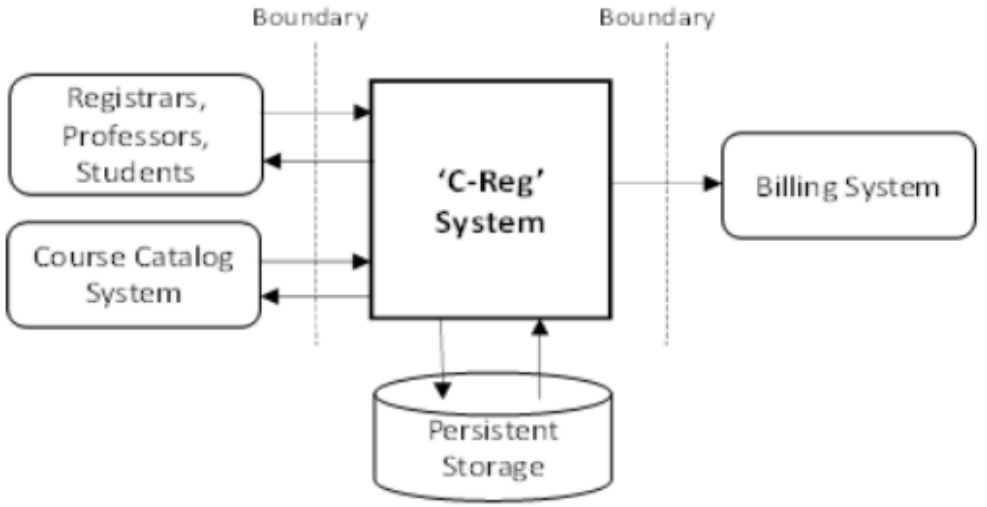
1 CFP + 2 CFP = 3 CFP

C-REG Case Study

The C-Reg case study [8] can be found in the COSMIC web application. It shows the whole process of measuring functional size.

To the best of our knowledge, this case study was not thought as MAS software. However, there is communication between the measured software and other pieces of software.

We assume that the three pieces of software mentioned in the case study were built as MAS. This premise does not affect the COSMIC measurement nor the Coupling measurement.



C-REG Case Study

No	Functional Process
1	Add a Professor
2	Enquire on a Professor
3	Modify a Professor
4	Delete a Professor
5	Enquire on Course Offerings (Professor)
6	Create Course Offering commitments
7	Modify Course Offering commitments
8	Delete Course Offering commitments
9	Add a Student's details
10	Enquire on a Student's details
11	Modify a Student's details
12	Delete a Student's details
13	Enquire on Course Offerings (Student)
14	Create a Student Schedule
15	Modify a Student Schedule
16	Delete a Student Schedule
17	Monitor Course Offering Enrolment progress
18	Monitor Student Schedule Enrolment progress
19	Close Registration

C-Reg app counts with 19 functional processes

- 11 of these do at least one data movement between C-Reg and one or more software functional users.
- The rest of the functional processes only communicate with human functional users.

C-REG CASE STUDY
FUNCTIONAL PROCESS: DELETE PROFESSOR

Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move -ment Type	CFP
Delete a Professor's details	Registrar	Registrar presses the delete command for the Professor whose details are displayed	Professor ID	Professor	E	1
	Course Catalog	C-Reg asks Course Catalog if Professor has any Course Offering commitments	Professor ID	Professor	X	1
	Course Catalog	Course Catalog replies 'yes' or 'no'	Professor's Course Offering commitment status	Course Offering	E	1
		C-Reg prompts the Registrar to confirm the deletion	Prompt Control Command		-	-
		The Registrar confirms or cancels the deletion	Confirmation Control Command		-	-
		C-Reg deletes the Professor	Professor details	Professor	W	1
	Registrar	Display error messages	Error Messages	Errors	X	1

C-REG CASE STUDY
FUNCTIONAL PROCESS: CLOSE REGISTRATION

Process descriptions	Functional user	Sub-process Description	Name of Data Group moved	Object of interest of Data Group moved	Data Move -ment Type	CFP
Close Registration	Registrar	The Registrar selects sub-option "Close registration"	Date closed	Course Offering	E	1
	Course Catalog	C-Reg requests Course Offering data (with number of students enrolled, etc.) from the Course Catalog	Course Offering Data request	Course Offering	X	1
	Course Catalog	Course Offering data received	Course Offering Data	Course Offering	E	1
		C-Reg checks that at least three students enrolled. If <3, it sets Course Offering status to 'cancelled'	(Data manipulation)		-	-
	Course Catalog	C-Reg sends updated statuses of Course Offerings to the Course Catalog	Course Offering statuses	Course Offering	X	1
		C-Reg retrieves the Student Schedule items for the Students enrolled for each Course Offering	Student Schedule item data	Student Schedule item	R	1
	Billing System	C-Reg sends data to the Billing System for each Student enrolled for each Course Offering that has not been cancelled	Student Schedule item data	Student Schedule item	X	1
		C-Reg retrieves Student name and e-mail address	Student details	Student	R	1
	Student	C-Reg sends info on each cancelled Schedule item to each Student in the form of an e-mail	Student e-mail address.	Student	X	1
	Student		Cancelled Student Schedule item message	Student Schedule item	X	1
		C-Reg updates Student Schedule items for cancelled Course Offerings	Student Schedule item data	Student Schedule item	W	1

C-REG CASE STUDY MEASUREMENT RESULT

		Number of Data Movements	
No	**Functional Process**	**Course Catalog**	**Billing System**
4	Delete a Professor	2	
5	Enquire on Course Offerings (Professor)	2	
6	Create Course Offering commitments	3	
7	Modify Course Offering commitments	3	
8	Delete Course Offering commitments	1	
13	Enquire on Course Offerings (Student)	2	
14	Create a Student Schedule	1	
15	Modify a Student Schedule	1	
16	Delete a Student Schedule	1	
17	Monitor Course Offering Enrolment progress	2	
19	Close Registration	3	1
	Total	**21**	**1**

COUPLING METRICS ANALYSIS

- International Standard: Is it based on an International Standard?
- Metrology Requirements: Does it comply with the metrology concepts mentioned by Abran [2]?
- Comparable: Is it valid to compare the measurement results of different software?
- Proved on MAS: Is there a case study where the metric (or an adaptation of it) was used to measure coupling between microservices?

Coupling Measurement	International Standard	Metrology Requirements	Comparable	Proved on MAS
Coupling with COSMIC	Yes	Yes	Yes	Yes
Allen and Khoshgoftaar [5]	No	No	No	NF
Arisholm et al. [6]	No	No	SP and EC	NF
Coupling Between Object Classes [7]	No	No	SP	No
Lavazza et al [14]	No	No	SP and EC	NF
Hassoun et al [11]	No	No	No	NF

CONCLUSION

Many companies are developing software based on MAS, because of the multiple benefits that come with it. Developers face multiple challenges when developing software based on MAS.

Some problems could be generated becase of the coupling that exists between microservices when achieving a particular functionality (High interdependency between teams, high network traffic, latency, etc)

We propose a way of measuring coupling between microservices. This metric is based on the COSMIC measurement standard to ensure that the measurement obtained is consistent, repetable and comparable.

		Number of Data Movements	
No	**Functional Process**	**Course Catalog**	**Billing System**
4	Delete a Professor	2	
5	Enquire on Course Offerings (Professor)	2	
6	Create Course Offering commitments	3	
7	Modify Course Offering commitments	3	
8	Delete Course Offering commitments	1	
13	Enquire on Course Offerings (Student)	2	
14	Create a Student Schedule	1	
15	Modify a Student Schedule	1	
16	Delete a Student Schedule	1	
17	Monitor Course Offering Enrolment progress	2	
19	Close Registration	3	1
	Total	**21**	**1**

FUTURE WORK

There can be situations where low-coupled microservices are generating a lot of network traffic (high-usage functionality), and high-coupled microservices are generating little network traffic (low-usage functionality). Study to observe the correlation between coupling and network traffic.

Low coupling is a software quality characteristic in all software, not only on microservices. It would be interesting to find a way to adapt the proposed metric so it takes into account all kinds of software.

Comparison of how reliable are other coupling metrics against ours. Taking into account that a good measurement method is independent of the person measuring and the measurement environment. The measurement results have to be consistent, repeatable, and comparable.

REFERENCES

[1] P. Jamshidi, C. Pahl, N. C. Mendonca, J. Lewis, and S. Tilkov. Microservices: The journey so far and challenges ahead. IEEE Software, 35(3):24–35, 2018.

[2] M. Fowler and J. Lewis. Microservices - A definition of this new architectural term, 2014

[3] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering,20(6):476–493, 1994.

[4] E. B. Allen, T. M. Khoshgoftaar, and Y. Chen. Measuring couplingand cohesion of software modules: An information-theory approach.International Software Metrics Symposium, Proceedings, pages 124–134, 2001.

[5] E. Arisholm, L, C Briand, and A. Føyen. Dynamic coupling measurement for object-oriented software. IEEE Transactions on Soft-ware Engineering, 30(8):491–506, 2004.

[6] L. Lavazza, S. Morasca, D. Taibi, and D. Tosi. On the definition of dynamic software measures. International Symposium on Empirical Software Engineering and Measurement, pages 39–48, 2012.

[7] A. Abran.Software Metrics and Software Metrology. 2010.

[8] A. Lesterius, A. Abran, C. Symmons. Course Registration (‘C - REG ’) System Case Study. (December):1–43, 2015.