Katia Isabelle Palmar Duarte

BSc

Limitations in the Support to Modularity in MATLAB: a Survey-based Empirical Study

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Miguel Pessoa Monteiro, Assistant Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-adviser: Fernando Brito e Abreu, Associate Professor, ISCTE

Examination Committee

Chairpersons:
Raporteurs:
Members:

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

march, 2017

Limitations in the Support to Modularity in MATLAB: a Survey-based Empirical Study

This document was created unsing the (pdf)LaTeX processor, based in the "unlthesis" template[1], developed at the Dep. Informática of FCT-NOVA [2]. [1] https://github.com/joaomlourenco/unlthesis [2] http://www.di.fct.unl.pt

To my mom and dad

Acknowledgements

I would first of all like to thank my advisers, Professor Miguel Monteiro of the Faculty of Science and Technology of the New University of Lisbon and Professor Fernando Brito and Abreu of ISCTE. Without their support and help provided during the creation of the research questions, the pilot test and the data analysis. They were always ready to help whenever I ran into a trouble spot or had a question about my research or writing.

I would also like to thank the experts who were involved in the pilot test phase for this research project: Professor Miguel Monteiro, Professor Manuel Ortigueira, Professor Paulo Gil, Professor Luís Palma, Professor Arnaldo Batista, Professor Francisco Monteiro, Professor Glauco Carneiro and Professor João Cardoso. Without their participation and feedback, the survey validation could not have been successfully conducted.

I would also like to acknowledge the help given towards the advertising of the survey in the communities by their administrators, and to thank all of the participants.

Finally, I must express my profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Abstract

This research compares modularity mechanisms provided in MATLAB with those of a mainstream programming language (Java) and describes the results of a survey-based study on the opinion of the MATLAB community about the limitations to the support of modularity provided by that language and its consequences in the light of Software Engineering theory. Social networks and email were used to broadcast the questionnaire to potential participants.

Four research questions were set at the start of the study, which originated eight hypotheses. Topics covered include: MATLAB developers' relationship with code tangling, code duplication avoidance, modularity, code maintainability and code structure. We analyzed if the opinions on those issues were different on industry and academia.

Main results are as follows. Developers are aware of code tangling in MATLAB software and care about it. They tend not to use the sole available tool to help them visualize code structure, but would like to have one. Developers from both industry and academia feel the same way about MATLAB code maintainability and code duplication avoidance.

Internal validity of the used research instrument was guaranteed, doubtful responses were discarded and results are statistically significant. However, external validity (generalization) of results cannot be claimed due to the sample size and its questionable representativeness.

Keywords: MATLAB; Software Engineering; Modularity; Unmodularized Concerns; Code Tangling; Surveys; Empirical Study.

Resumo

Esta pesquisa compara os mecanismos de modularidade fornecidos pelo MATLAB com os de uma linguagem de programação mainstream (Java) e descreve os resultados de um estudo baseado em pesquisas sobre a opinião da comunidade de programadores MATLAB acerca das limitações ao suporte da modularidade e suas consequências. As redes sociais e o e-mail foram os mecanismos usados para partilhar o questionário com potenciais participantes.

Quatro questões de pesquisa foram estabelecidas no início do estudo, que depois vieram a originar oito hipóteses. Os tópicos abordados incluem: relacionamento dos programadores de MATLAB com código emaranhado, evitar código duplicado, modularidade, manutenção de código e estrutura de código. Analisamos se as opiniões sobre essas questões eram diferentes na indústria e no meio acadêmico.

Os principais resultados obtidos são os seguintes. Os programadores estão cientes e preocupam-se com a existência de código emaranhado em código MATLAB. Os programadores tendem a não usar a única ferramenta disponível para ajudá-los a visualizar a estrutura de código, mas gostariam de ter uma. Os programadores que usam MATLAB, tanto na indústria como no meio académico, sentem a mesma coisa sobre a manutenção de código em MATLAB e evitar código duplicado.

A validade interna do instrumento de pesquisa utilizado foi garantida, as respostas duvidosas foram descartadas e os resultados são estatisticamente significativos. No entanto, a validade externa (generalização) dos resultados não pode ser reivindicada devido ao tamanho da amostra e sua representatividade questionável.

Palavras-chave: MATLAB; Engenharia de Software; Modularidade; Facetas não modularizadas; Código Emaranhado; Questionários; Estudo Empírico.

Contents

List of Figures

List of Tables

Listings

Glossary

.mltbx Packaged custom toolbox.

CCC Concerns that are directly responsible for code tangling and scattering, resulting in loss of modularity.

concern A concern is considered to be, in software engineering, any concept, feature, requirement of the problem or set of responsibilities that we would like to localize on its own module.

convenience sampling The nearest and most convenient persons are selected as subjects.

data logging The process of using a computer to collect data through sensors, analyze the data and save and output the results of the collection and analysis. Data logging also implies the control of how the computer collects and analyzes the data.

dynamic language Dynamic programming language is a language that doesn't force the check type-safety during compile-time. In other words, executes most of its logic during at runtime. Some of these languages can be, also considered, scripting languages.

module A module in MATLAB, can be expressed as a function or m-file.

MOF A set of standard interfaces that can be used to define and manipulate a set of interoperable meta-models and their corresponding models.

positivism Positivism is a philosophy where only the knowledge gained through observations can be considered trustworthy. In empirical studies that means, that the researcher and its research are limited to the data collection and its interpretation through an objective approach, to concluded anything..

quasi-experiment In this research, quasi-experiment means that the participants for our questionnaire were not chosen at random, but the participants themselves choose if they wanted to participate or not.

Acronyms

AOP Aspect-Oriented Programming.

CCC Crosscutting Concerns.

CCM Cyclomatic Complexity Metric.

CNC Coefficient of Network Complexity.

FEUP Faculdade de Engenharia da Universidade do Porto.

IP Internet Protocol.

ISCTE-IUL ISCTE-Instituto Universitário de Lisboa.

MATLAB MATrix LABoratory.

MOF Meta-Object Facility.

OOP object-oriented paradigm.

PIMETA Paradigm Independent Meta-model.

PP procedural paradigm.

Chapter 1

Introduction

MATrix LABoratory (MATLAB) is a programming language used by the scientific, engineering and research communities. There is currently about 1500 books based on the language (and its software companions), and it's translated in 27 different languages. MATLAB is used in different areas like scientific computing, control systems, signal processing, image processing, simulation, computational finance [34], among other fields. The different uses given to programs created with MATLAB means, that the developers may have different levels of programming knowledge (and academic backgrounds), which results in a huge diversity of developers with different skills. These differences between MATLAB developers, will probably affect the importance that each of them give towards modularity concerns/aspects and the limitations in the support to modularity given by the language.

Another factor that may influence the said importance, is the amount of legacy code that a developer uses when developing their solutions, or when maintaining solutions created by others. According to Joost Visser in its book 'Building Maintainable Software' [54], sometimes there are *newly built systems for which the maintainability was so low that it was no longer possible to effectively modify them—even before the systems went into production. Modifications introduced more bugs than they solved.* Although the quote and the book are related to code written in Java, the same concept can be applied to any other programming language (in our case MATLAB) and that may create a new problem. If solutions written in MATLAB have these issues regarding their level of maintainability

(and readability) from the beginning, then the developers may not even get to see or feel the limitations in the support to modularity that language suffers.

In this dissertation, we focus on the symptoms and consequences caused by the limitations in the support to modularity and how MATLAB developers feel regarding this situation. More about the limitations can be read in Section 1.1 and 3.4.

1.1 The Problem: Limitations in the Support to Modularity in MATLAB

Figure 1.1 captures one of the symptoms that the limitations in the support to modularity in MATLAB causes, *code tangling*. The figure represents two different implementations of the "Discrete Fourier Transform" function, which is often used in signal processing.

On the left, we have a clean version of the function with a single concern. On the right, there is a more complex version of the function that, besides taking care of the original concern, it also cares about data type specialization. The data type specialization concern can be detected through the eight occurrences, in twenty-two lines of code, of the 'quantize' (marked in green) and 'quantizer' (marked in yellow) functions. Therefore, making the function on the right, an extreme case of code tangling.

```
function [y] = dft(x)
  y=zeros(size(x));
  N=length(x);
  t=(0:N-1)/N;
  for k=1:N
     y(k) = sum(x.*exp(-j*2*pi*(k-1)*t));
  end
```

Before: ***Discrete Fourier Transform***

```
function [y] = dft_specialized(x)
  y=zeros(size(x));
  N=length(x);
  t=(0:N-1)/N;
  quant1=quantizer('fixed','floor','wrap', [18 16]);
  t=quantize(quant1, t);
  quant2=quantizer('fixed','floor','wrap', [23 20]);
  pi_fix = quantize(quant2, pi);
  quant3=quantizer('fixed', 'floor', 'wrap', [20 8]);
  quant4=quantizer('fixed','floor', 'wrap', [23 10]);
  quant5=quantizer('fixed','floor', 'wrap', [24 10]);
  quant6=quantizer('fixed','floor', 'wrap', [26 12]);
  quant7=quantizer('fixed','floor', 'wrap', [28 14]);
  quant8=quantizer('fixed','floor', 'wrap', [32 16]);
  for k=1:N
     v1 = quantize(quant3, (k-1)*t);
     v2 = quantize(quant4, pi_fix*v1);
     v3 = quantize(quant5, -j*2*v2);
     v4 = quantize(quant6, exp(v3));
     v5 = quantize(quant7, x.*v4);
     y(k) = quantize(quant8, sum(v5));
  end
```

After: ***Discrete Fourier Transform + Data type specialization***

Figure 1.1: Code tangling [5]

Code tangling and other symptoms, in this study, are just the visible consequences

caused by MATLAB limitations in the support to modularity, something that we already know. What we are going to research is whether MATLAB developers feel and/or acknowledge these limitations, and the consequences that they bring to their code.

1.2 The Approach: Survey-based Empirical Study

The approach taken is to gather the required data needed to establish, that MATLAB developers do care about the limitations and consequences presented in the previous section. To do that, we choose to create a *survey-based empirical study* to tackle the problem. The use of this approach and its overall process, help us verifying and understand what is the opinion from the MATLAB developers' community, about the impact of these limitations in their code and programming habits.

The study, starts with a description about what are Crosscutting Concerns (CCC), since they are the visible result from the limitations provided in the code. It's important to note that, although CCC is related to the concept of Aspect-Oriented Programming (AOP), this dissertation doesn't cover issues specifically pertaining AOP. Specifically, MATLAB extensions, that help the language support this programming paradigm. [1]

After that learning curve, we start our empirical study by creating our hypothesis and from there, our research questions. This knowledge helps us build a more appropriate survey, so the answers can be more focused in the problem we are studying. Finally, we choose the best way to sample the population of MATLAB developers, to make the process of delivering the questionnaire easier and faster.

1.3 Research Objectives

The main objective for this dissertation is assessing whether MATLAB developers recognize the limitations to modularity in MATLAB, and that it creates some difficulties while programming. That proof will enable us to validate (scientifically), past and current works related to this problem, and motivate further developments. Those developments will occur in a more focused and clear way, since it will have concrete knowledge about what these developers (in both, the industrial and academic world) feel towards these limitations.

Our second objective is to proceed to the categorization of MATLAB developers. That categorization is done using questions to gather background information, and depending on the response given by the subject about the problem. With that information,

[1] To learn more about this theme, see papers [4] and [5].

we can make some correlations so we can associate each type of answers to a specific use.

1.4 Research Questions

At the start of our empirical study, we defined what were the questions for which we set out to obtain answers. In other words, we described and refined our objectives. This helped setting a more concrete context to our study by detailing and adding focus to our problem[56]. The following list is comprised with our four main research questions.

1. Do MATLAB developers suffer from software legacy problems?
2. Which are the modularity traits in MATLAB?
3. Which factors influence the modularity practices of a MATLAB developer?
4. Can structure visualization features help MATLAB developers in understanding legacy software?

Table 1.1: Main Research Questions

After defining our main research questions, we proceeded by creating a group of specific questions for each of them.

1.1 Do MATLAB developers deal with large programs produced by other developers (legacy software)?
1.2 Are the code in those legacy software hard to understand?
1.3 Do MATLAB developers experience difficulties in understanding the structure of legacy software?
2.1 Is cohesion / tangling a concern for MATLAB developers?
2.2 Is coupling / scattering a concern for MATLAB developers?
3.1 The application domain influences the modularity practices of MATLAB developers?
3.2 Developers' background influences their modularity practices in MATLAB?

Table 1.2: Sub-questions for each Research Question

1.5 Hypothesis Formulation

According to Eric M. Rogers, hypothesis is a *"... single tentative guesses–good hunches–assumed for use in devising theory or planning experiment, intended to be given a direct*

experimental test when possible." [47]

When it comes to formulate a hypothesis, we need to make it clear and formal. We also need to present the necessary background to understand where the hypotheses are derived from [21]. The background to our problem is given in Section 1.1 and 3.4. Considering the effects of these limitations in the code and the impact that it can have in the developers works, the following formal null hypotheses were created:

Hypothesis 1. MATLAB developers don't find or care about tangling of concerns.

Hypothesis 2. While developing, be either creating or maintaining a system, MATLAB developers never felt the necessity of using a tool to help visualize the code structure.

Hypothesis 3. There are no evidences that the limitations in the support to modularity in MATLAB, affect the maintainability (and readability) of the code.

Hypothesis 4. The academic and industrial professional background of a developer doesn't influence the importance given to the limitations (and its consequences) in the support to modularity in MATLAB.

Hypothesis 5. The academic and industrial professional background of a developer doesn't influence the importance given occurrences of code tangling in MATLAB code.

Hypothesis 6. The academic and industrial professional background of a developer doesn't influence the importance given to MATLAB code maintenance.

Hypothesis 7. The academic and industrial professional background of a developer doesn't influence the importance given to code duplication avoidance while developing software in MATLAB.

Hypothesis 8. The academic and industrial professional background of a MATLAB developer doesn't influence the importance given to code structure.

The *statistical significance* level (α) for testing the *null hypothesis* is 5% (0,05). α is the probability of a *Type I error*. In other words, we are calculating the probability of incorrectly rejecting the null hypothesis [27]. A lower level would be viable given a large

enough sample size, which will not be the case here due to limited time and number of participants.

1.6 Document Structure

The dissertation is organized, excluding the current chapter, as follows:

- **Chapter 2: MATLAB programming language** gives the necessary background information about MATLAB. This also includes a short comparison between MATLAB and GNU Octave and the languages syntax. This chapter is part of the related work/literature review done for this research.
- **Chapter 3: MATLAB Modularity Study** is the result about a study regarding modularity in MATLAB. It contains sections about a comparison between modularity in oriented-object and the procedural paradigm, and about the limitations to the support in MATLAB. This chapter is part of the related work/literature review done for this research.
- **Chapter 4: Study Design** describes the outcome of the survey planning phase, including a study about the research paradigms, a description of the types of empirical studies and comparison between the conditions for each of them. It also includes an explanation about the survey design, variables, planning of the survey research, a research about the target population, methods to sample the participants, instrument, data collection procedure, analysis procedure and validity evaluation.
- **Chapter 5: Execution** describes each step taken in the execution of the study. It includes sampling, description about the pilot test step, execution of the questionnaire, data collection performed and threats to the study validity.
- **Chapter 6: Analysis** resumes the data collected and the treatment given to it. That analysis includes a descriptive, multivariate, covariance and inferential analysis, as well as hypothesis testing.
- **Chapter 7: Conclusions and Future Work** presents a summary of the study, including results, contributions, research limitations found during the study and future work.

Chapter 2

MATLAB Programming Language

2.1 Introduction

The following chapter can be considered the first part of our related work/literature review. The chapter revolves around the necessity of making an overview of how the MATLAB programming language works, considering that this is part of the topic behind our survey-based empirical study. This aspect, makes this chapter more of a programming language review, instead of a normal literature review.

The overview of MATLAB starts with a small description of its uses and history, the basic syntax that can be used to program, how we can structure our code in m-files and a brief comparison of MATLAB with GNU Octave, one of the most similar clone languages available [50].

2.2 History

MATLAB is an interpreted, procedural, imperative (like C and Fortran), proprietary and dynamic language, with an interactive environment. The language was initially used by the applied mathematics community because of its ability to allow matrix data manipulations [6]. Later, it expanded to other fields like scientific computing, control systems, signal processing, image processing, simulation, computational finance among other fields [34].

MATLAB was created in the Computer Science Department at the University of New Mexico, in the late 1970s by Cleve Moler[1]. After being acquired by MathWorks in 1984, it underwent a complete redesign (and continuous updates) by starting to use toolboxes. One of those is Simulink (only available in MATLAB), that is used to do simulation and control system design.

2.3 Basic Syntax

Like any other programming language, MATLAB has a basic syntax. That syntax is composed by variables, arrays, operators, statements, functions and toolboxes. In this section, we will describe the syntax available in MATLAB.

2.3.1 Variables

As it was described in the beginning of this chapter, MATLAB focus is matrix data manipulations making the variables, by default, matrices and arrays. Considering that, it's an interpreted language there is no need to declare if the variables are of type integer, string or combinations of the two, before assigning a value [10]. Integer variables include numeric values such as negative infinite, positive negative, and zero value. See some examples of variables in Algorithm 2.1.

```
% Numeric Value
value = 4;

% String value
string = 'Hello';

% Array with values from 1 to 10
array = [1:10];

% Matrix 3-by-3
matrix = [3 6 9 54; 10 34 79];

% Variable with function assigned to it
function = myfun();
```

Listing 2.1: Examples of Variables Assignment

[1] To learn more about the creator of MATLAB, see http://www.mathworks.com/company/aboutus/founders/clevemoler.html.

Variables can either be global or local. They are usually local variables, which means that they can only be accessed in the function where they are declared. In the case of global variables, those variables are declared once and all the functions that call it, share a single copy of the variable [12].

2.3.2 Arrays

Array or cell array is a data structure where each cell is indexed and can have any type of data [8]. This data can be empty, a list of text string, numeric values or a combination of the last two.

There are different ways to declare an array, see examples in Algorithm 2.2.

```
% Array with values from 1 to 10
a = [1 2 3 4];

% Matrix 3-by-3
m = [1 2 3; 4 5 6; 7 8 10]

% Matrix 3-by-3 with 1
o = ones(3,3)

% Matrix 5-by-1 with 0
z = zeros(5,1)
```

Listing 2.2: Examples of Declaring Arrays

2.3.3 Operators

Operators are symbols that tell the compiler to perform a certain mathematical or logical operation. Although, MATLAB is designed to operate (primarily) with matrices and arrays, these operators can work with scalar and non-scalar values. These operators and elementary operations can be divided into five different groups [40]:

- Arithmetic Operations (Table 2.1);
- Relational Operations (Table 2.2);
- Logical Operations;
- Set Operations;
- Bit-Wise Operations.

When it comes to arithmetic's (see Table 2.1), there are two types of operations: array operations and matrix operations [3]. While array operations execute element-by-element operation, matrix follows the rules from linear algebra.

	Operator for Array	Operator for Matrix	Function for Array	Function For Matrix
Addition	A+B	A+B	plus(A, B)	plus(A, B)
Subtraction	A-B	A-B	minus(A, B)	minus(A, B)
Multiplication	A.*B	A*B	times(A, B)	mtimes(A, B)
Right Division	B./A	B/A	rdivide(A, B)	mrdivide(B, A)
Left Division	A.	A	ldivide(B, A)	mldivide(A, B)
Exponentiation (A^B)	A.^B	A^B	power(A, B)	mpower(A, B)
Transpose	A.'	A'	transpose(A)	ctranspose(A)

Table 2.1: Arithmetic Operations

Relational operators perform element-by-element comparisons between arrays with, either the same size or if one is a scalar (see Table 2.2) [2]. The results given by these operators are a logical array, that shows where the relational is true.

	Operator	Function
Less than	A	lt(A, B)
Greater than	A	gt(A, B)
Less than or equal to	A <= B	le(A, B)
Greater than or equal to	A >= B	ge(A, B)
Equal to	A == B	eq(B, A)
Not equal to	A ~=B	ne(A, B)
Array equality	-	isequal(A, B, ...)
Array equality (treating NaN values)	-	isequaln(A, B, ...)

Table 2.2: Relational Operators

Logical operations return logical values (0 = false and 1 = true) to show, if the condition that is being tested was full field. Set operations are used to perform joins, unions and intersections between two arrays. Bit-wise operators are used to set, shift or compare a specific bit/value in one array.

2.3.4 Statements

Control statements require that the programmer use one (or more) condition(s) to evaluate the code. Then, controlling the flow of execution of the program itself. To create those conditions, the programmer can use the operators visible in Table 2.1 and 2.2, plus other operators like bit-wise, logical and set ones. Some of those statements are

represented here [9], and in the following algorithms (from Algorithm 2.3 to Algorithm 2.6).

```
a = 5;

%check the boolean condition
if a == 10
    % if condition is true
    fprintf('Value of a is 10\n' );
elseif( a == 15 )
    % if elseif condition is true
    fprintf('Value of a is 20\n' );
else
    fprintf('None of the values are matching\n');
end
```

Listing 2.3: If Statement

```
grade = 'B';

switch(grade)
    case 'A'
        fprintf('Excellent!\n' );
    case 'B'
        fprintf('Well done\n' );
    case 'C'
        fprintf('Well done\n' );
    case 'D'
        fprintf('You passed\n' );
    case 'F'
        fprintf('Better try again\n' );
    otherwise
        fprintf('Invalid grade\n' );
    end
```

Listing 2.4: Switch Statement

```
fib = ones (1, 10);
i = 3;

while (i <= 10)
    fib (i) = fib (i-1) + fib (i-2);
    i++;
end
```

Listing 2.5: While Statement

```
% creates a matrix 1-by-10
fib = ones (1, 10);

% for cycle where i value goes from 3 to 10
for i = 3:10
    fib (i) = fib (i-1) + fib (i-2);
end
```

Listing 2.6: For Statement

These statements are the building blocks to create more complex algorithms in MATLAB.

2.3.5 Functions

A function can be created with the use of operators (Section 2.3.3), statements (Section 2.3.4) and other functions, where it can have as many inputs and output values. When it to comes to the name of the function to be valid, this needs to begin with an alphabetic character[11]. Otherwise, it can contain other letters, number, or underscores. See an example in Algorithm 2.7.

```
% Function that finds the maximum value in the array. It returns
% the value and its position in the array.

% function [output1, output2, ...] = myfunction(input1, ...)
function [value, index] = my_max(array)

    j = length(array);
    % Starts value with negative infinite. So, that in the
    % beginning any number will become the maximum value, even if
    % it is a negative one.
    max_value = -inf;

    for i = 1:j
        if array(i) > value
            value = array(i);
            index = i;
        end
    end
end
```

Listing 2.7: Example of a function

In the case of a function having other functions in it, we call it a nested function. See example in Algorithm 2.8.

```
function show_string
    a = 10;
    fprintf('Old value of a %d\n', a);
    b = 20;
    c = 'Hello';
    output(c);
    fprintf('New value of a %d\n', p);

    function output(string)
        disp(string)
        p = a * b;
    end
end
```

Listing 2.8: Example of a nested function

MATLAB offers a list of available functions[2] to help speed up the process of developing programs. A couple of those functions can be seen in Table 2.1, Table 2.2 and Table 2.3.

To learn more about functions and how do they work inside a m-file, go to Section 2.4.1.

	Description
nargin	Returns the number of input arguments specified for a function.
nargin('fun')	Returns the number of declared inputs for the M-file function fun or -1 if the function has a variable of input arguments.
nargout	Returns the number of output arguments specified for a function.
nargout('fun')	Returns the number of declared outputs for the m-file function fun.

Table 2.3: Input and Output Arguments

2.3.6 Toolboxes

Toolboxes are created by a collection of MATLAB files, and when it's created MATLAB generates a single file using .mltbx for filename extension. [51]. These files can be either

[2]For a complete list of MATLAB functions, see the following website `http://www.mathworks.com/help/matlab/functions.html`.

m-files (see Section 2.4), data, documents, apps, or examples.

2.4 M-files

MATLAB allows the programmer to write two types of m-files, functions files and script files. The names of both kinds of file must end with an extension of .m, so it can always be MATLAB-compatible [48].

2.4.1 Function Files

Except for simple one-shot programs, it is not practical to have to define all the functions you need each time you need them. Instead, you will normally want to save them in a file so that you can easily edit them, and save them for use at a later time [19]. That specific file is called a function file. Functions files can accept inputs and return outputs, while the internal variables are only local to the function. This file, normally starts with the keyword function.

Unlike Java, in MATLAB it's not required that the main function have the same name as the m-file. So, the common name is used for reasons of clarity and good style. When the function and the file name differ, the file name must be used to call the main function (first function in the m-file). All the subsequent functions are called local functions, and are called by the main function and other local functions in the same m-file [30]. Local functions are easy to identify in a m-file since they must end with the *end* keyword.

Another type of functions available are private functions. This type is only accessible to other functions in a specific location, like local functions. They can be found in a subfolder with the name private. Private functions are only available to functions founded in the folder immediately above to the private one. They are used to separate code into different files, or to share it between multiple, related functions [45].

2.4.2 Script Files

Script files are m-files containing MATLAB statements, variables, operators and function calls. These files don't accept input or return outputs, which means that they can only work on data that is hard-coded into the file. In other words, data that is in the workspace [49].

They're useful when the task in hand, doesn't require changes. They are used to specify a sequence of commands just as if you had typed these commands into the

command window. This way, they make all the variables created within the script been added to the workspace, for the current session. Furthermore, if any variable in the script have the same name as a current one in the workspace, then those variables in the workspace are updated with the new value.

2.5 GNU Octave compatibility with MATLAB

In this section, we will make a brief comparison between these two languages, MATLAB and GNU Octave, as it's explained in Section 2.1. To note that GNU Octave offers to its developers an identical basic syntax as the one described in Section 2.3, with minor differences that are explained in Subsection 2.5.3.

2.5.1 GNU Octave History

GNU Octave is an interpreted language similar to MATLAB (where it provides support to matrix data types and operations) and distributed under the GNU General Public License[3] terms [1]. GNU Octave is a structured programming language, with the possibility to use some of the most common C libraries functions, and certain UNIX systems calls and functions [18]. It's considered open-source, so any developer can contribute with new features and functions to the language.

The GNU Octave project started in 1988, with the idea of being a software a companion to a chemical reactor design textbook, written by two Professors from the Universities of Texas and Wisconsin-Madison. In 1992, by the hands of John W. Eaton, the project was launched into a full-time development and its popularity started reaching new highs. Today, it's used by thousands of people in teaching, research, and commercial applications. An example, it's a project created to find vulnerabilities related to guessing social security numbers, through the access to a large-scale parallel computer at Pittsburgh Supercomputing Center [4].

2.5.2 Similarities

GNU Octave is an open-source clone of MATLAB and the one with the most similarities[50], making these two languages have many features in common. These specific features enable the developers to write their code in one language, and still maintain (at

[3]GPL is published by the Free Software Foundation (learn more about it at `http://www.gnu.org/`).

[4]To learn more about the project and how was GNU Octave used see `https://www.psc.edu/publicinfo/news/2009/70809_SSVulnerabilities.php`.

a certain level) the ability to interpret it, in the other. This happens because they both offer:

- Matrices as the main data type;
- Native support for complex numbers;
- Powerful built-in math functions;
- Extensive functions libraries;
- Extensibility regarding user-defined functions (or UDF) [5].

To help the developers develop in GNU Octave, in a more compatible way to MATLAB, they can use it in traditional mode[6]. This mode makes GNU Octave interpret the code in a MATLAB-compatible mode

2.5.3 Differences

Even with the high compatibility between programs created in MATLAB and GNU Octave, there are still some differences. If they are considered minor differences, the developers can solve them by using user preference variables. They also can be irrelevant, when they don't affect the execution of the m-files. These differences can be caused by the ability GNU Octave gives, to customize the language to be either MATLAB-compatible or to use new and different features. Some of these differences presented in the following list, were previous listed by Lessa et al.[32] and by Sysnet[7] webpage[13]:

- In GNU Octave the relational operators are called comparison operations [17]), and all the available functions can be seen here at `http://octave.sourceforge.net/function_list.html`.
- In GNU Octave, toolboxes are actually called packages, and they're created by the developers as they need it[8]. These packages may lack some of the features given by MATLAB toolboxes, and they might not duplicate exactly the MATLAB functions or interface [15].

[5] UDF is a programmed routine with parameters set by the user of the system. The name of the function, in MATLAB, is determined by the name of the file containing the function.

[6] To learn more about this topic go to `https://www.gnu.org/software/octave/doc/interpreter/Command-Line-Options.html`.

[7] Service made available by the Institute for Computational Engineering and Sciences from the University of Texas at Austin.

[8] To see the list of packages available go to `http://octave.sourceforge.net/packages.php`.

- Some functions from MATLAB are not available in GNU Octave. That happens because of the large number of toolboxes, and proprietary libraries that belongs to MATLAB[39]. A clear example of this, is the Simulink toolbox;
- To end statements in GNU Octave, we can use either *end* or *endif*, *endwhile* and *endfor*. MATLAB only accepts *end*;
- Similar functions can have different names in each language;
- GNU Octave supports C-based operators like ++, –, +=, *= and /=;
- GNU Octave accepts ~= and *!*= as not-equal comparison, while MATLAB only accepts the former;
- Comments in MATLAB start with %, although GNU Octave can also use # to start a comment;
- GNU Octave developers have the possibility to use the do-until statement [20]. See an example in Algorithm 2.9;
- MATLAB uses ^ for exponentiation. GNU Octave also accepts the following syntax, **;
- GNU Octave use ‘ and ’’ as string delimiters, while MATLAB only use ‘;
- GNU Octave can use the logical operators | or & (for matrices), and || or && (for scalars). In case of using the scalar operators, the MATLAB compiler gives a warning that, those operations have short-circuiting behavior[9].

Finally, although MATLAB and GNU Octave have object-oriented capabilities in common, they have different implementations[10]. MATLAB can be extended by adding new objects using m-scripts, while in GNU new objects are implemented like C++ classes [35]. These capabilities are not part of our study, since the previous studies about limitations in modularity are related to the procedural paradigm version of each language.

[9]Logical short-circuiting evaluates the second operand, without first completely evaluating the result given by the first operand.

[10]To learn more about these capabilities, see the following links: `http://www.mathworks.com/discovery/object-oriented-programming.html` and `https://www.gnu.org/software/octave/doc/interpreter/Object-Oriented-Programming.html`.

```
do
    number = input('Give a number between 1 and 10: ');
until (number >= 1 & number <= 10)

for i = 1:10
    printf('%d x %2d = %2d\n', number, i, number*i);
end
```

Listing 2.9: Times table code with the use of the do-until statement

2.6 Conclusion

This chapter described the programming language review based on MATLAB. The most important facts that may contribute for our research study is the principal programming paradigm that MATLAB follows, the procedural paradigm.

It was also discovered that it's possible to program MATLAB in clone languages such as GNU Octave. That ability occurs because the similarities between these two languages are significant, and the differences can be either considered bug by the GNU Octave interpreter or are special features that only work in GNU Octave. Examples of such features are the 'Do-Until Statement' and the possibility to use specific end statements for each statement, instead of the MATLAB-compatible one.

The discovery of this level of compatibility between (at least) these two programming languages changed and helped defining our target population. Now, instead of only wanting to questioning MATLAB developers who work with the MathWorks workspace, we can ask any MATLAB developer regardless of its workspace of choice.

Chapter 3

MATLAB Modularity Study

3.1 Introduction

The present chapter is the second part of our related work/literature review. This chapter starts with a brief explanation about what is modularity, followed by a Paradigm Independent Meta-model (PIMETA) instantiation with the procedural programming model of MATLAB and a section comparing MATLAB to Java. Finally, we explain in more detail the limitations in the support to modularity found in MATLAB, that are summarized in Section 1.1.

The idea of modularity in Computer Software has been around for almost six decades [41]. Making modularity an important Software Engineering principle, with a strong impact on software maintenance and reusability. According to Myers, it's *the single attribute of software that allows a program to be intellectually manageable* [38]. Modularity in a software engineering, can be seen when a system (in this case, a programming language) is composed by features that when interacting with each other creates a dependency.

3.2 PIMETA instantiation of MATLAB Grammar

The PIMETA diagram was created as way to exemplify the relationships and/or dependencies that a paradigm causes when implemented in a certain language, since it brings a group of atomic and modular features. Atomic features are features that don't

allow the aggregation to other features. Consequently, modular features (also known as modules) are features that can aggregate other features. Those aggregated features can be other modular features or atomic ones.

The meta-model represented in Figure 3.1 can be logically divided into two different parts. The division is represented by the double line and inspired by the Meta-Object Facility (MOF).

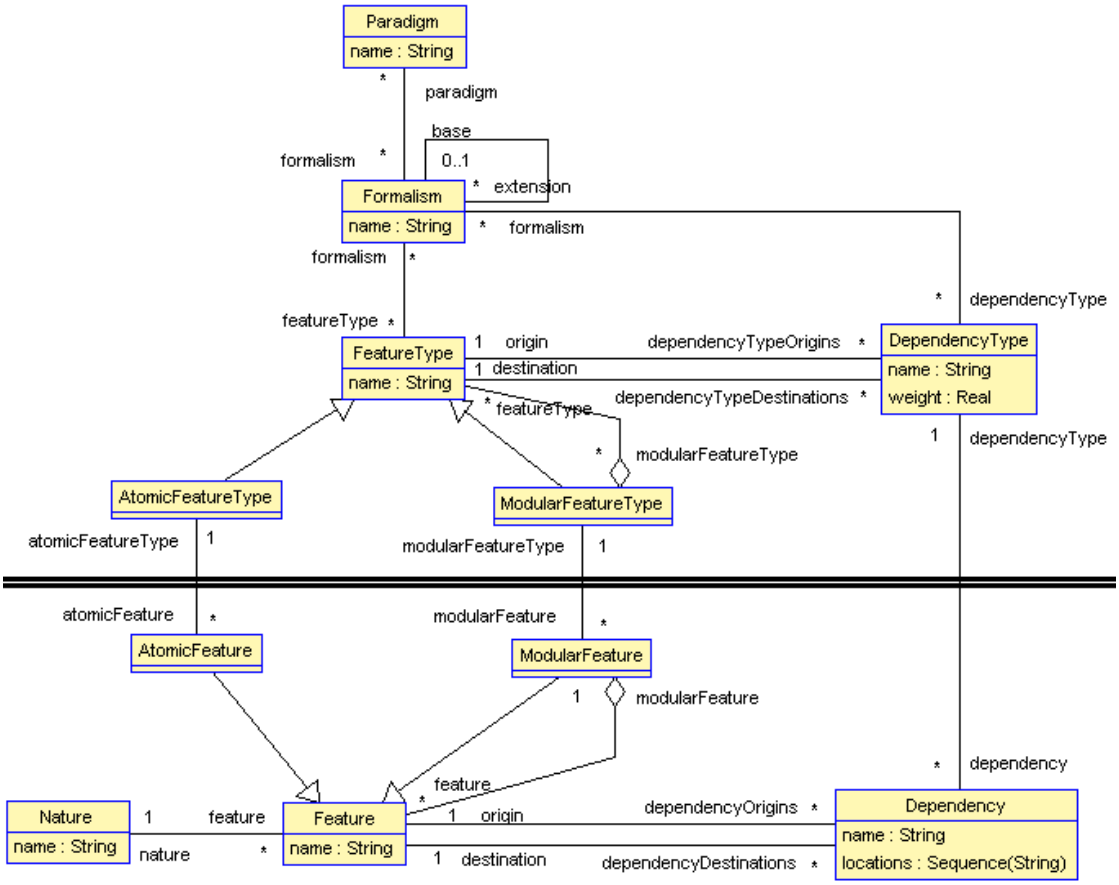


Figure 3.1: PIMETA Meta-class diagram

The top layer depicts Level M2 (or UML meta model) and it's the definition of concepts like classes, attributes, associations. In our diagram, the top layer is composed by the *Paradigm*, *Formalism*, *FeatureType*, *ModularFeatureType*, *AtomicFeatureType* and *DependencyType* meta-classes. These features allow to represent a paradigm, and more

specially a programming language, with the features and dependencies that they offer and how they can be organized. The *Paradigm* meta-class is meant to represent paradigms like procedural paradigm (PP) or object-oriented paradigm (OOP), while the *Formalism* is meant to represent languages like MATLAB or Java. The *DependencyType* meta-class that represents the different dependencies between features, like functions can call other functions in MATLAB. Finally, we have the abstract meta-classes *FeatureType*, *ModularFeatureType* and *AtomicFeatureType*. These three meta-classes put together create a structural design pattern where a group of objects are to be treated in the same way as a single instance of an object, called composite pattern (see figure 3.2). This design pattern was described at first in the book *Design Patterns: Elements of Reusable Object-Oriented Software* [55] from 1994.

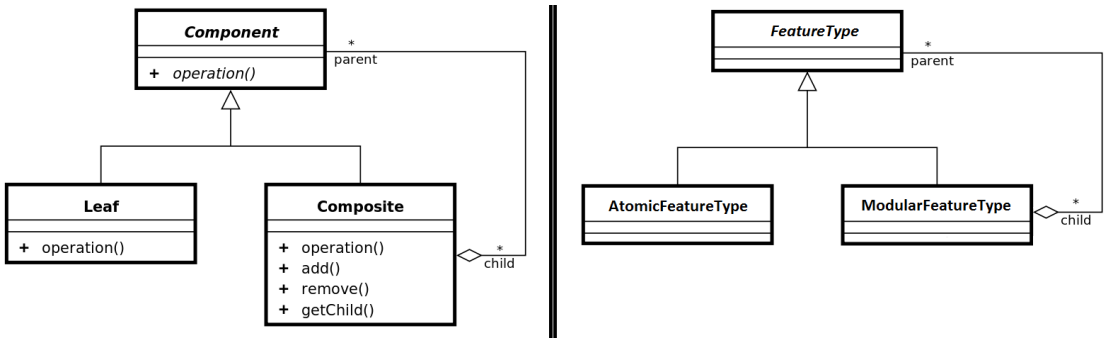


Figure 3.2: Composite Pattern in PIMETA diagram

In our case, *FeatureType* meta-class is the component, the abstract meta-class that defines the methods that the *ModularFeatureType* and *AtomicFeatureType* meta-classes, composite and leaf respectively, and must be implemented by them. Meta-class *ModularFeatureType* implements the methods from the textitFeatureType and adds others, while *AtomicFeatureType* is a meta-class that is not extended or inherited by others.

The bottom depicts the Level M1 (or the UML model). Level M1 is the instantiation of the UML meta model and it contains the application-specific models. In our diagram, the bottom layer is composed by the *Feature*, *ModularFeature*, *AtomicFeature* and *Dependecy* classes. These classes represent the organization and analysis of a programming language, where *ModularFeature*, *AtomicFeature* and *Dependecy* classes always have a correspondent meta-class in Level M2 Metamodel.

The PIMETA instantiation of the PP with the MATLAB grammar formalism, can be observed in the object diagram depicted in Figure 3.3. Due to the space and readability

limitations this diagram only contains the relationship between the *Paradigm*, *Formalism*, *FeatureType*, *ModularFeatureType* and *AtomicFeatureType* meta-classes. See Appendix A for further details in the instantiation of MATLAB grammar.

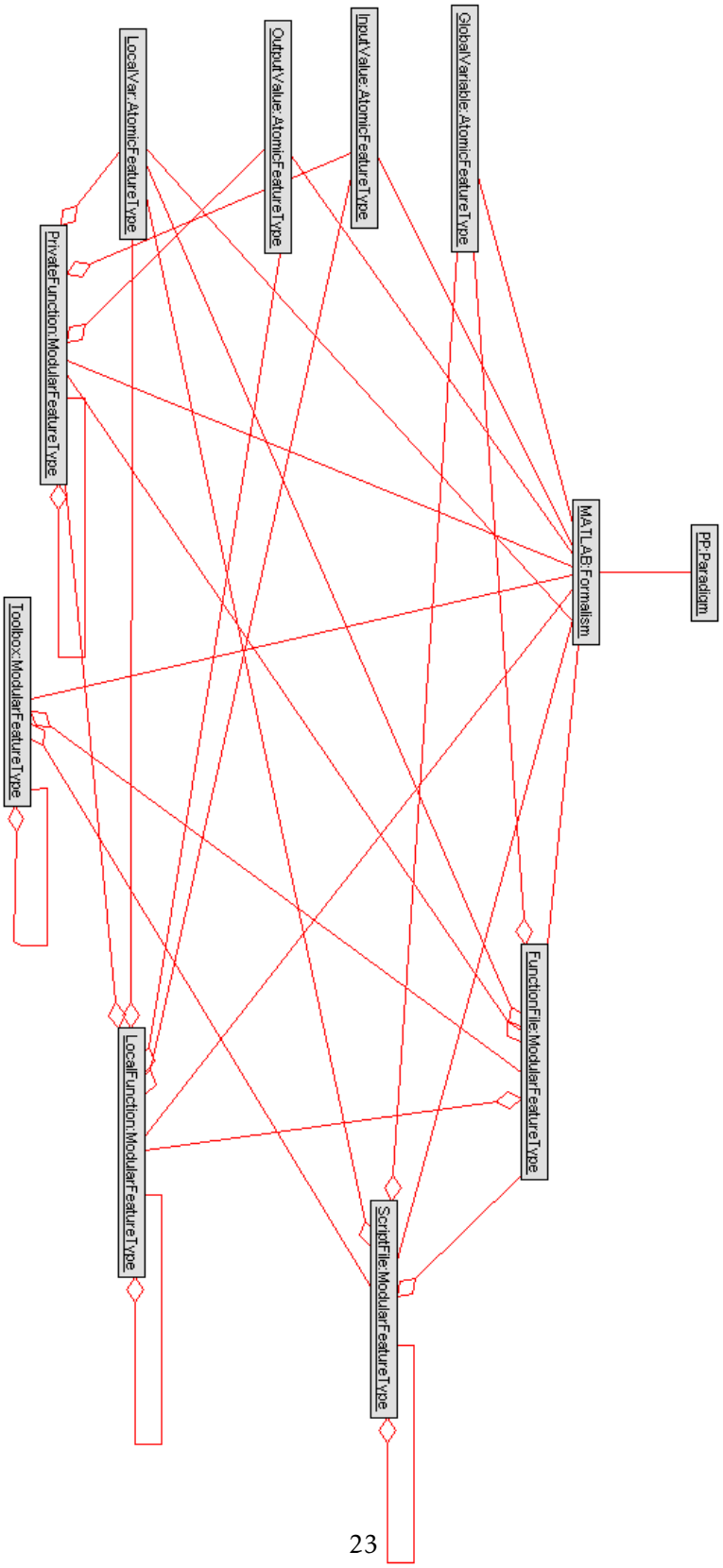


Figure 3.3: PIMETA instantiation extract for MATLAB Grammar

3.3 Comparing modularity between MATLAB and Java

Comparing the levels of modularity offered between MATLAB and Java, it is basically the same, as comparing modularity between a PP and an OOP one. There are some works like the paper by Ferrett and Offutt (2002) [16], that compares the modularity of procedural and object-oriented software using empirical studies. The conclusions described that:

> *The modules of the object-oriented programs were found to be half the size of those of the procedural programs and the average number of parameters per module for the object- oriented programs was approximately half that of the procedural programs. Thus, the object-oriented programs were twice as modular as the procedural programs.*

To assess the truth behind the previous statement, we used the instantiation of the PIMETA diagram for MATLAB and Java, to calculate the Coefficient of Network Complexity (CNC) and the Cyclomatic Complexity Metric (CCM). The instantiation for Java, shown in Figure 3.4, was done by Sergio Marques in is MSc dissertation [33]. The MATLAB instantiation can be seen in Figure 3.3.

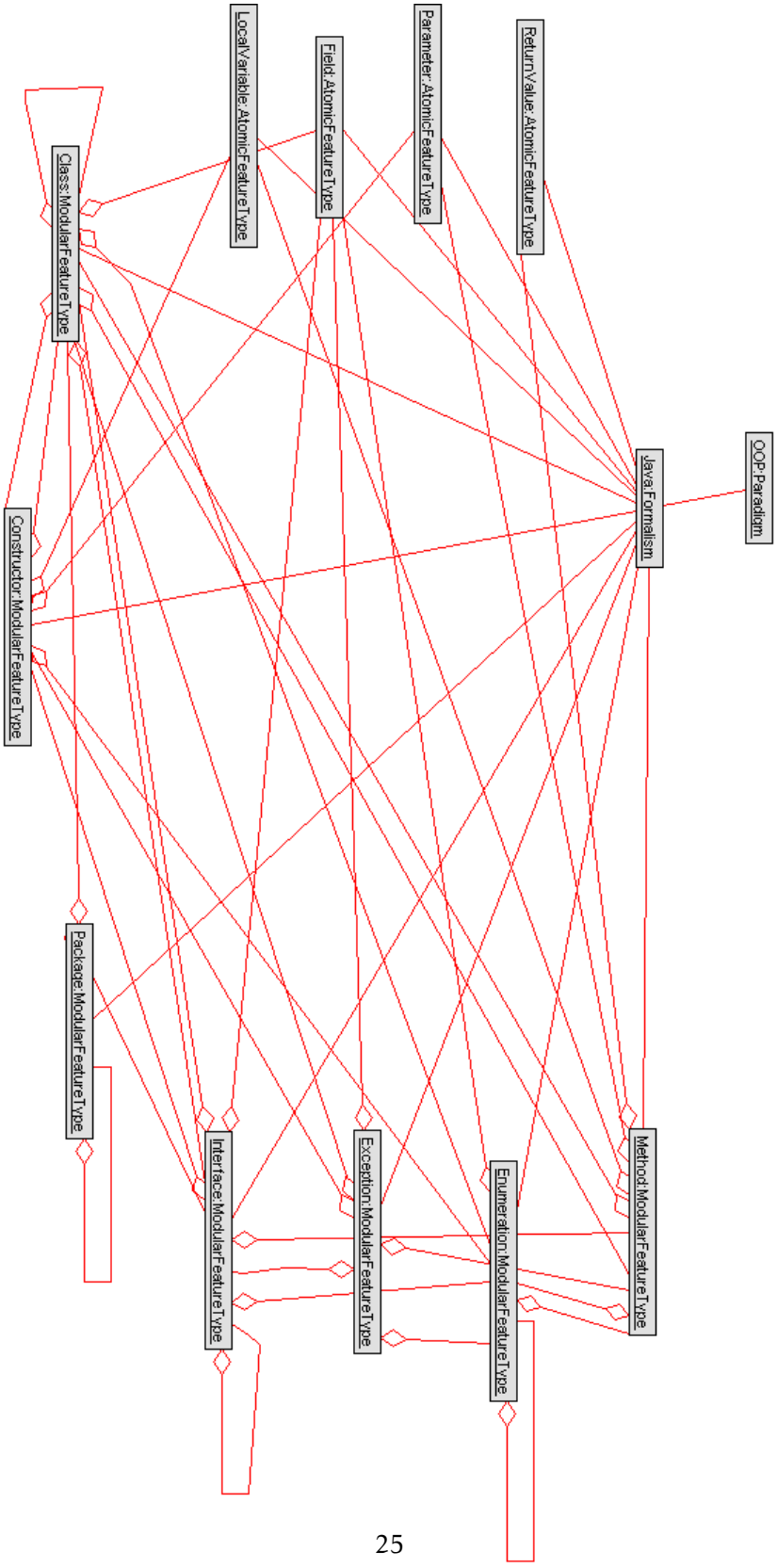


Figure 3.4: PIMETA instantiation extract for Java

CNC metric is used to calculate the degree of complexity of a critical path network, or a graph in our case. This metric was first proposed by Richard Kaimann in the paper with the same name as the metric in 1974 [24]. The complexity can be calculated by dividing the quotient of activities squared by events or preceding work items squared by work items. In this case, we adapt the CNC metric to use in graphs such as the one in Figure 3.5, where we divide the number of arcs by the number of nodes. Next, we have the mathematical formula of the metric and its adaptation to OCL for the PIMETA diagram.

$$CNC = \frac{number\_arcs}{number\_nodes}$$

```
1  ModularAggregationsSize(): Integer = ModularFeatureType.allInstances->select(
       formalism->includes(self)).featureType->size
2
3  DependencyRelationsSize(): Integer = DependencyType.allInstances->select(
       formalism->includes(self))->size * 2
4
5  FeatureTypesSize(): Integer = FeatureType.allInstances->select(formalism->
       includes(self))->size
6
7  CNC_FeatureTypes(): Real =
8    ModularAggregationsSize() / FeatureTypesSize()
9
10 CNC_DependencyTypes(): Real =
11   DependencyRelationsSize() / FeatureTypesSize()
```

Listing 3.1: CNC metric for the PIMETA diagram

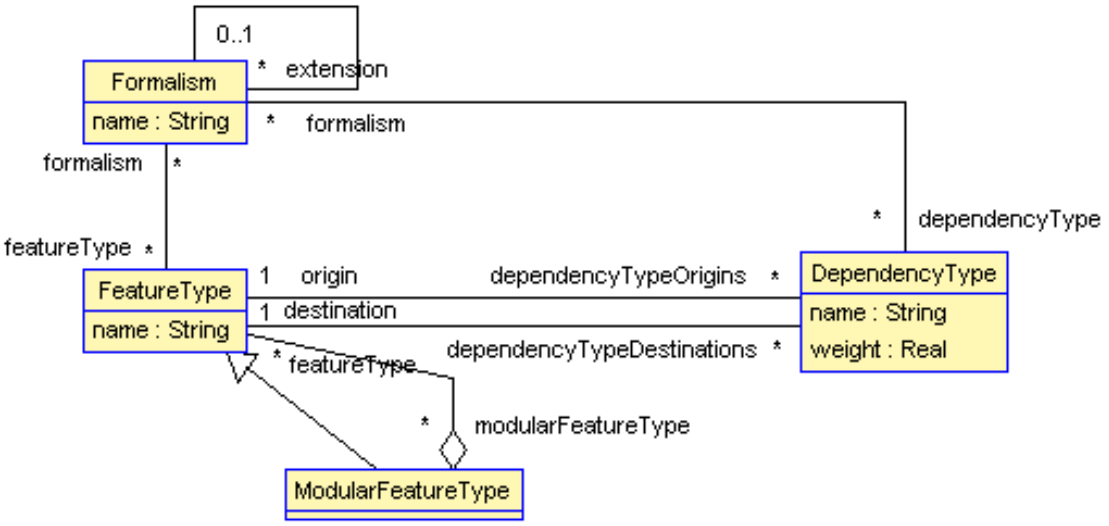


Figure 3.5: Graph from PIMETA meta-class diagram

The number of arcs can be the number of modular features, the number of dependencies multiplied by two because of the number of associations between the meta-class *FeatureType* and *DependencyType*, or the combination of both. The number of nodes are the number of elements that belong to the meta-class *FeatureType*. In other words, all the features that a programming language have.

	MATLAB	**Java**
Features	2.222	3
Dependencies	2.222	6

Table 3.1: CNC values comparison between MATLAB and Java

CCM was one of the first complexity measures developed by Thomas McCabe in 1976[36]. This software metric is used to calculate the control flow charts. In a simple way, it calculates the number of linearly independent paths that exist through the source code. In our case, is more through the number of possible dependencies between features. The original formula for this software metric is:

CCM = number of arcs - number of nodes + number of entry or exit points

In the CCM formula, the number of arcs and the number of nodes are the same as for the last formula. The number of entry or exit points, in our case, is 2. Next, we have the CCM metric adaptation to OCL for the PIMETA diagram.

```
1  ModularAggregationsSize(): Integer = ModularFeatureType.allInstances->select(
       formalism->includes(self)).featureType->size
2
3  DependencyRelationsSize(): Integer = DependencyType.allInstances->select(
       formalism->includes(self))->size * 2
4
5  FeatureTypesSize(): Integer = FeatureType.allInstances->select(formalism->
       includes(self))->size
6
7  CCM_FeatureTypes(): Real =
8    ModularAggregationsSize() - FeatureTypesSize() + 2
9
10 CCM_DependencyTypes(): Real =
11   DependencyRelationsSize() - FeatureTypesSize() + 2
```

Listing 3.2: CCM metric for the PIMETA diagram

	MATLAB	**Java**
Features	13	24
Dependencies	13	57

Table 3.2: CCM comparison between MATLAB and Java

After calculating this two complexities, we can say that the level of modularity in Java is superior than in MATLAB. However, there is a threat to the validity of results described in Table 3.1 and 3.2. Our MATLAB instantiation has had only one iteration and therefore it may not be detailed enough to compare the complexities results against Java. The PIMETA instantiation of Java doesn't have such problems, since it's the result of numerous iterations.

3.4 Limitations in the Support to Modularity in MATLAB

In the latest years, the scientific community in Portugal has done some research work related to modularity in MATLAB, which contributed to new perspectives – which comprises the basis for this dissertation – are mainly explained in papers SPLAT 2006 [6] and CoRTA 2010 [37] papers.

The problem described in Section 1.1, as it is described there is related to the limitations in the support to modularity in MATLAB programs. One of the symptoms

that help identify this problem, and is explained that Section, is the symptom of tangling. To recap, code tangling can be identified, when each individual function or m-file contains code related to more than one concern, such as illustrated in Figure 1.1. In that figure, we can observe code related to the primary concern been intertwined with code pertaining other concerns, which may hamper the code understandability [31] and maintainability.

The other symptom, that helps pinpoint these limitations, is called scattering. Code scattering is the representation of unmodularized concerns, in the form of small code fragments, spread throughout many functions or/and m-files instead of being localized within a single module. A visible consequence to code scattering is the creation of duplicated code through the program. That symptom causes difficulties towards the maintainability, understandability and the ability to reuse the code.

To help with the identification of these symptoms, we can use the functions in the Table 3.3. That table was created using a token-based approach, which extracts all names, filters out most keywords and variable names and assumes the rest are functions. Token in this context means certain words, that can be functions, that appear multiple times through a program. The idea was that the number of occurrences of a function in a piece of MATLAB code can be used as an indicator of code tangling. This part of the study and the Table 3.3 are based on the paper by Cardoso et al. (2013) [7] and the paper by Monteiro et al. (2010) [37].

Table 3.3: CCC Categories

Categories	**Description**	**Examples of MATLAB Functions**
Messages and monitoring	Messages to the user, warnings, errors, graphics visualization, monitoring, etc.	plottools, semilogx, semilogy, loglog, plotyy, plot3, grid, title, xlabel;
I/O data	Reading data from file, writing data to file, saving an image, loading an image, etc.	imwrite, imread, imformats, hgsave, saveas, hgload, save;
Verification of function arguments and return values	Default shapes and values for the arguments that may not be passed in certain function calls.	nargchk, nargin, nargout, varargin, varargout;
Data type verification and specialization	Check whether a variable is of certain type, configuring the assignment of data types to variables, etc.	quantize, quantizer, fi, isscalar, isstruct, isempty, iscell;
System	Code that verifies certain system environment properties, to pause execution, etc.	pause, print, printopt, wait, last, input, syntax, run, tic, start;

Categories	Description	Examples of MATLAB Functions
Memory allocation/ deallocation	The use of the 'zeros' function is most of times used to allocate a specific array size. This avoids the reallocation for each new item to be stored in an array. Use of the 'clear' instruction that appears in some MATLAB functions is another example.	clear, delete, zeros, persistent, global;
Parallelization	Use of parallel primitives such as 'parfor'.	parfor, spmd, feval, demote, cancel, submit, resume;
Dynamic properties	Constructing, inline, function, objects(inline), executing a string containing MATLAB expressions('eval'), etc.	eval, eval, evalc, evalin, inline.

3.5 Conclusion

This chapter presented our implementation of the PIMETA instantiation for MATLAB grammar, a comparison between MATLAB and Java and a study about the limitations to the support in MATLAB.

Concerning the modularity limitations, there is some research done about the detection and effects about non-modularized concerns in object-oriented systems. An example of that research, is the seminal paper on AspectJ [25]. The problem is that, as we verified in Section 3.3 and Chapter 2, MATLAB follows mainly a PP and that the level of modularity in Java is superior than in MATLAB. We notice this even without taking in account that the typical uses for MATLAB are different than the ones given to OO languages[37]. These reasons among others, means that MATLAB may give rise to new and different code symptoms that don't exist in Java, which significantly reduces their usefulness in the new context.

Nevertheless, we don't know if the MATLAB community feels this problem, while coding in MATLAB. So, the first step we need to take, before trying to find a solution or study which are the concerns that standouts the most in MATLAB, is to assert if this is a problem or not for MATLAB developers.

Chapter 4

Study Design

4.1 Introduction

This chapter presents the planning for our empirical study. It serves as a blueprint for the execution of the survey and interpretation of its results.

The design is based on the problem, hypothesis and research questions that support it, and are described in Chapter 1. A matching research paradigm is then selected, along a type of empirical study that better suites our needs. Following that, the details for our survey design are discussed, including its variables, planning, how we choose our participants, sample methods, instrumentation and the procedures for the data collection and analysis. Finally, we discuss the problems that may occur regarding the validity of our study.

4.2 Research Paradigms

Empirical studies can be approached through two different types of research paradigms, exploratory and explanatory. There is a need to clarify that these two research paradigms don't mutually exclude each other, but that they complement each other.

Exploratory research finds the answers to the problem through observations given by the subjects. This entails the use of a flexible research type of design (also referred as a qualitative research) because of the variation that takes place during observations.

This design is used when the desired result is to study the beliefs, and understandings of the population [57].

Explanatory research, meanwhile, is more focused in comparing at least two groups and identify the cause-effect relationship between them, or just quantifying a relationship. For those reasons, plus the fact that this type of research tends to be conducted in a controlled set up, the design used is a fixed one. Fixed design or quantitative research, entails that the main conditions in the study can't be modified in the middle of the study. This design will test the effect that a certain treatment (or form of tackling a problem) can have into solving the problem [57].

4.3 Types of Empirical Studies

According to Wohlin et al. [57] and Pfleeger [42], there are some conditions that should be taken in consideration when deciding which type of empirical study, the researcher will use. Depending on those conditions, there are three major types of research strategies that can be used: surveys, case study and experiment. See Table 4.1 (based on Table I from paper [53]) for a short comparison between strategies.

	Survey	**Case Study**	**Experiment**
Level of Control	Low	Low	High
Difficulty of Control	High	High	Low
Level of Replication	Low	Low	High
Cost of Replication	Low	Low	High
Investigation Cost	Low	Medium	High
Design Type	Fixed	Flexible	Fixed
Data Type	Qualitative & Quantitative data	Qualitative & Quantitative data	Quantitative data

Table 4.1: Conditions affecting the choice of empirical study

In the rest of this section, we give a brief explanation about each type of empirical study. However, for this study we decided to use the survey approach as suggested by this chapter and dissertation title. The main factor contributing for this choice is the type of design chosen, fixed design. After making that choice, we have two possible options, survey and experiment. Out of these two, the one that gives the best conditions overall, for our type of study, is the survey.

4.3.1 Survey

In his book[57], Wohlin et al. describes survey research as follows:

> *A survey is a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes and behavior.*

One of the defining characteristic in this type of study is the selection of a sample that represent the population in question, with the data analysis techniques are used to generalize the results gathered.

Survey research can be conduct through the use of three different types of instruments: *questionnaires*, *interviews* and *data logging* techniques. However, the instrument that researcher tend to use more is the questionnaire. That occurs because it's the cheapest instrument, and the easier and faster to construct and deliver to its population.

A condition to use this empirical study is to have a clear idea of what are the research questions. The major challenge is control of the sampling bias and ensuring that the questions are written in a way that every participant understands them, especially when the target population has different backgrounds.

4.3.2 Case Study

Yin introduces case studies in his book [58] as:

> *An empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident.*

Case study can use different types of instruments to acquire data. Those instruments, in case of the data type being qualitative, can be either interviews or direct observation of the participants.

A condition to use this empire study is to have a clear idea of what is the research question related to how or why the phenomena happens. This helps the researcher to derive the study proposition, which in turn declares what is the final objective of the study and how to select the cases and the type of data he wishes to collect [14]. A major challenge is that the data collection and its analysis are more susceptible to open interpretation and bias.

4.3.3 Experiment

Experiments are a type of empirical study where the main objective is to test a hypothesis where we have at least more than one independent and dependent variable. After creating a testable hypothesis, the next step is to determine how the variables are related and if there is a cause-effect relationship between them, with each combination being called a treatment [14].

4.4 Survey Design

There are different types of survey designs that are created through the combinations of design, instrument, distribution and scientific method used. In this section, we explain the design and scientific method we used, while instrument and distribution method are described in Section 4.9 and Section 4.10, respectively.

There are two types of design research, descriptive and experimental. The main difference between these two designs is the way we manipulate our variables. In the descriptive design, we already know what are our variables and we only going to observer them, making this type of research and observational study. On the other hand, in the experimental design the variables may change according to response gathered from the experiment done. With that in mind, it makes sense to choose the descriptive design, since we know already what our problem and its variables.

A descriptive design can be divided into three big groups: cross sectional, cohort and case control. Cross sectional and case control both question the subjects about their past, while cohort control is more focused on the future. Sometimes, there can be some doubt in whether to choose cross sectional or case control, considering that both look at past actions. The biggest difference between them, is that cross sectional gives a snapshot of what is happening on the matter, and case control is a retrospective type of study [43]. Seeing that we are going to ask our subjects about their recent experience with MATLAB, we will use case control as our descriptive design.

According to the characteristics displayed in our research, with the formulation of our hypothesis and the type of empirical study been use is survey research, we can say that we have adopted a hypothetic-deductive positivist scientific method. The hypothetic-deductive method comes from the declaration of the hypothesis in Chapter 1, while the positivist (or positivism) is because the survey research follows this philosophy. The hypothetic-deductive scientific method was proposed by Karl R. Popper in his book [44], and consists in offering possible solutions to our problem. So, the researcher

then, through the application of various tests, will try to refute the hypothesis thus creating new knowledge. This knowledge may help other researchers identifying similar or new problems. The steps proposed by Popper for this specific scientific method are described in Figure 4.1, followed by a brief explanation for each step.

Figure 4.1: Popper's hypothetic-deductive method steps

- **Knowledge of the Problem:** Consider the problem and try to make sense of it. That can be done through gathering data, and/or considering previous explanations. See Section1.1.

- **Hypothesis Formulation:** Try to state an explanation, when nothing else is yet known by formulating a hypothesis regarding the problem. See Section 1.5.

- **Assume hypothesis is true:** In this step, we will build our empirical study around the belief that our hypothesis, H_0, is true.

- **Hypothesis Testing:** Look for scientific evidence that conflicts with the hypothesis H_0. That evidence is obtained through the data gathered with the help of the instrument, used in the empirical study.

If the hypothesis is proven false, we must go back to the hypothesis formulation and create a new one. Figure 4.2, based on the diagram created by Lakatos and Marconi [29], demonstrates this cycle through the use of an activity diagram.

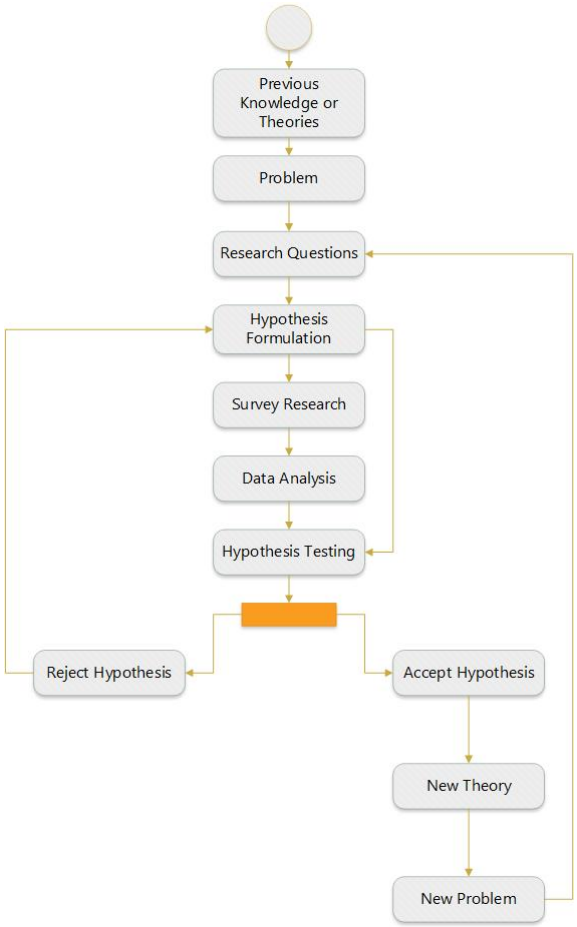


Figure 4.2: Hypothetic-deductive method schema [29]

4.5 Variables

4.5.1 Dependent Variables

Dependent variables, are normally just a single variable per hypothesis. The dependent variables chosen for this study are modularity for *Hypothesis 4*, code tangling for *Hypothesis 5*, code duplication avoidance for *Hypothesis 7*, code structure for *Hypothesis 8* and code maintenance for *Hypothesis 3* and *Hypothesis 6*. Although, it is possible to choose other dependent variables, the ones that made the most sense regarding the hypotheses proposed at the beginning of our study and dissertation (see Section 1.5).

4.5.2 Independent Variables

Independent variables are the ones we can change and control during our research [57]. In this survey-based empirical study we have 4 variables. Code tangling, code structure, modularity and participant background. The first three independent variables correspond to the first three hypotheses, in the same order. The last independent variable, participant background, is used in hypothesis 4 to 8. This variable is about if the participants develop MATLAB software in an academic or industrial background.

4.6 Planning

The survey research ran from April 4 to September 23, 2016. This time frame encompasses the realization of the pilot test, the delivery of the questionnaire to the participants and the data analysis. Most of the assigned time was dedicated towards the broadcasting of the questionnaire throughout the communities.

The research followed the activity diagram described in Figure 4.3.

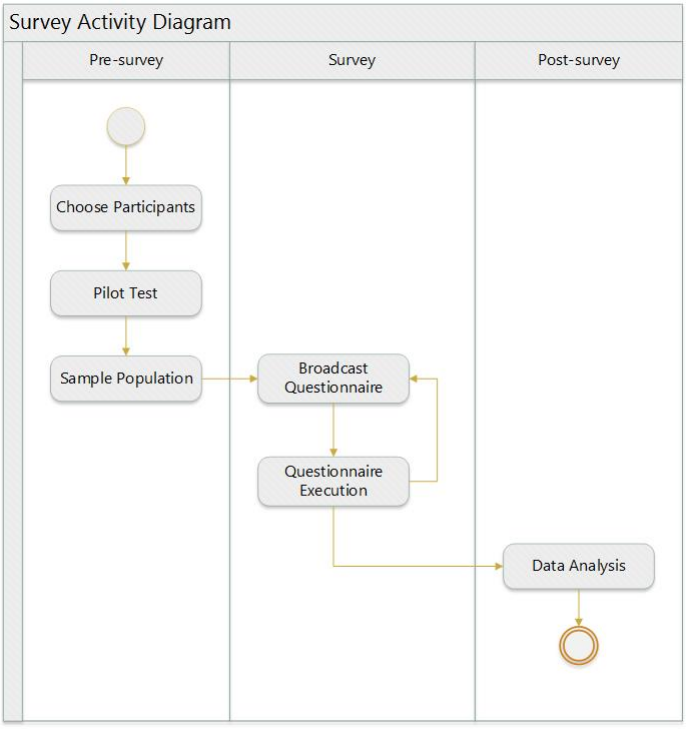


Figure 4.3: Survey plan activity diagram

1. **Choose Participants.** The first thing we did, is define our target population, out of all the MATLAB developers.

2. **Pilot test.** In this step, we tested our questionnaire. After each pilot iteration, we perfected the questionnaire following received feedback.

3. **Sample Population.** Before we begin our survey study, we need to sample the population created with the help of step 1 – Choose Participants.

4. **Broadcast Questionnaire.** The broadcast of our questionnaire starts with the delivery of an invitation to answer it, through the chosen social networks communities and email. If the number of responses per week decreases, we send a new broadcast to all the participants to remind them about our questionnaire.

5. **Questionnaire Execution (15 minutes).** All the participants who are, either invited by email to respond to our questionnaire or decide to do it after seeing one of the broadcast invitations, needs around 15 minutes to complete it. After reading the questionnaire introduction, explaining our research problem and accepting the conditions of the questionnaire, they get access to the questionnaire. When they reach the end, the participants can opt to leave their email and later receive the preliminary results.

6. **Data analysis.** With all the other steps completed, analysis of the collected data begins.

4.7 Participants

Identification and selection of possible participants for our study is an important step, since our ability to extrapolate the result gathered from the questionnaire to our population is related to the representativeness of the sample.

In this study, instead of using all the population of MATLAB programmers, we decided that for our results to be more realistic to use a target population. The target are developers who had worked in large projects using MATLAB and recently, or at least in the last 10 years.

For our pilot test, we invited Universities Professors. Those Professors are considered specialist in either programming and maintaining software created in MATLAB, or in the topic of the problem described in Section 1.1.

For our questionnaire, we decided to find MATLAB and GNU Octave developers communities available online. The outcome of that search is presented in the next subsection, and in a summarized form in Table 4.7.1. The other selection method used, to gather participants for our questionnaire, was to create a filter and search through all of the papers in Google Scholar[1] until we had at least 25 papers, and then collect the author's emails and send them the questionnaire. For further details, see Subsection 4.7.2.

4.7.1 Communities

There are multiple communities of MATLAB and GNU Octave developers online. Those communities exist so they can share their success stories, test theories, share code, and interact with others. Some of those communities can be found in social networks like LinkedIn[2] and Facebook, or at MATLAB Central[3]. Table 4.2 contains some information about a couple of communities reachable online, by any MATLAB or GNU Octave developers. The data collection that yielded this table initially took place in January 10th, and was revised in April 1st, 2016.

MATLAB and GNU Octave communities in LinkedIn are very active ones where participants with different backgrounds share their doubts and thoughts about these two languages. The MATLAB communities have over 6.958 members, while the GNU Octave have about 386.

MATLAB Central is an online forum launched in 2001 by MathWorks. Here the MATLAB developers can access as registered community members to all the capabilities normal in an online community, plus the fact that they can interact with MathWorks technical experts and suggest product improvements.

[1] Google Scholar is a free web search engine that indexes the paper/book o metadata of academic literature across an array of publishing formats and disciplines. To see more about it go to `https://scholar.google.com/`.

[2] Business-oriented social networking service with active communities of MATLAB and GNU Octave developers. See more about LinkedIn at `https://www.linkedin.com/`.

[3] To learn more about MATLAB Central, see `http://www.mathworks.com/matlabcentral/`.

	Link	Members
MATLAB Central	http://www.mathworks.com/matlabcentral/	225.000
MATLAB (Facebook)	https://www.facebook.com/groups/Matlab.Simulink.for.All/	73.769
Matlab Programming (Facebook)	https://www.facebook.com/groups/44152194130/	26.055
MATLAB Users and Integrators (LinkedIn)	https://www.linkedin.com/groups/134533	24.978
Matlab (LinkedIn)	https://www.linkedin.com/groups/68980	13.792
Matlab for beginners and experts (LinkedIn)	https://www.linkedin.com/grp/home?gid=1843503	6.958
MATLAB (Google+)	https://plus.google.com/communities/112299668727392739262?hl=pt-PT	6.441
GNU Octave (Google+)	https://plus.google.com/communities/112976184309608583944?hl=pt-PT	557
GNU Octave users and developers (LinkedIn)	https://www.linkedin.com/groups/4044339	464
OCTAVE Programme (LinkedIn)	https://www.linkedin.com/groups/4328430	386

Table 4.2: Online Communities

4.7.2 Participants Filter

When conducting a scientific research of this kind, where the population (and the target population) is difficult to pinpoint, we can create a frame population that fits our criteria using a search engine and a filter. Although this solution may give rise to some limitations when we try to generalize our results to all the population, it's still considered a viable method. It helps the researcher overcome problems related to the lack of a target population with well-defined number of members and, even, when there is a lack of answers from other communities.

Since we are trying to find researchers that have a good knowledge about MATLAB (or any of its clone language, e.g. GNU Octave and Scilab) we choose to use Google Scholar as our search engine. As for filter or criteria, the main ones or key words are ***"matlab" "software engineering" "modularity"***. This translates to all the papers (and books, but for our case we decided to discard those) that have these specific words mentioned somewhere in it.

The result for this search, made on June 27, 2016, is of almost 2000 papers/books. We believed that to be too big for what we had in mind, so we decided to restrict our filter even more. That restriction is limiting the publishing date of the paper, to be either from 2012 or earlier. From this search version, we restrict our results to 700 papers/books. With this result, we decided to only consider papers who had been cited at least 5 times. So, in the end we were left with 26 papers and 79 available researcher emails.

4.8 Sampling

Depending of the size of the sample, it's possible to divide the survey into two categories: census and sample survey. When we do a census, it means that we intend to contact and survey the entire population. On the other hand, in a sample survey we choose to only

contact a part of the population, called sample. This is the most common option and it enables the researcher to infer information about the population, while only using a portion of it.

Since it's a sample survey, we stat by defining our target population. According to Houston, he says in his book [22], that:

> *A population consists of all members of an organization or group of people who possess the desired traits, knowledge, experience, or characteristics of interest to the survey project.*

The problem is when the population is too big and/or unknown. In these case, we may want to narrow down our population numbers. We can do that by adding certain aspect that must be complied, thus creating our target population. This means that our target population is a subset of our initial chosen population (see Figure 4.4). After doing this step, we need to create a valid sample from our target population.

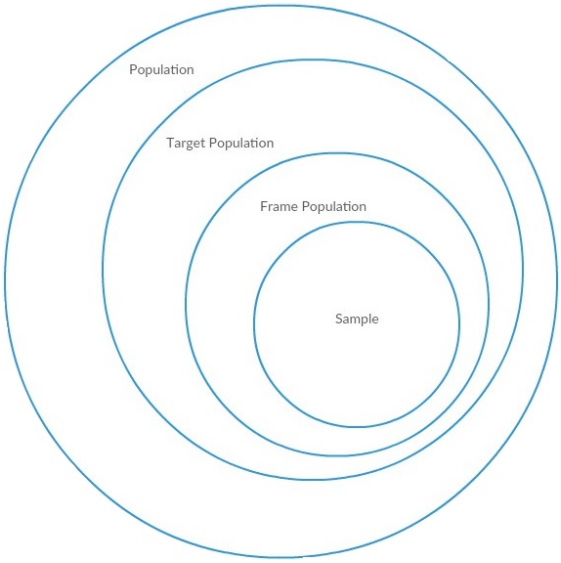


Figure 4.4: Venn Diagram for Population and Samples

A valid sample is a representative subset of the target population (see Figure 4.4). If our sample isn't representative, we will not be able to generalize our results to all the population. For that, we use sampling methods that can be either probabilistic or non-probabilistic. The next figure (Figure 4.5) explains through an activity diagram, all the steps necessary to sample our population.

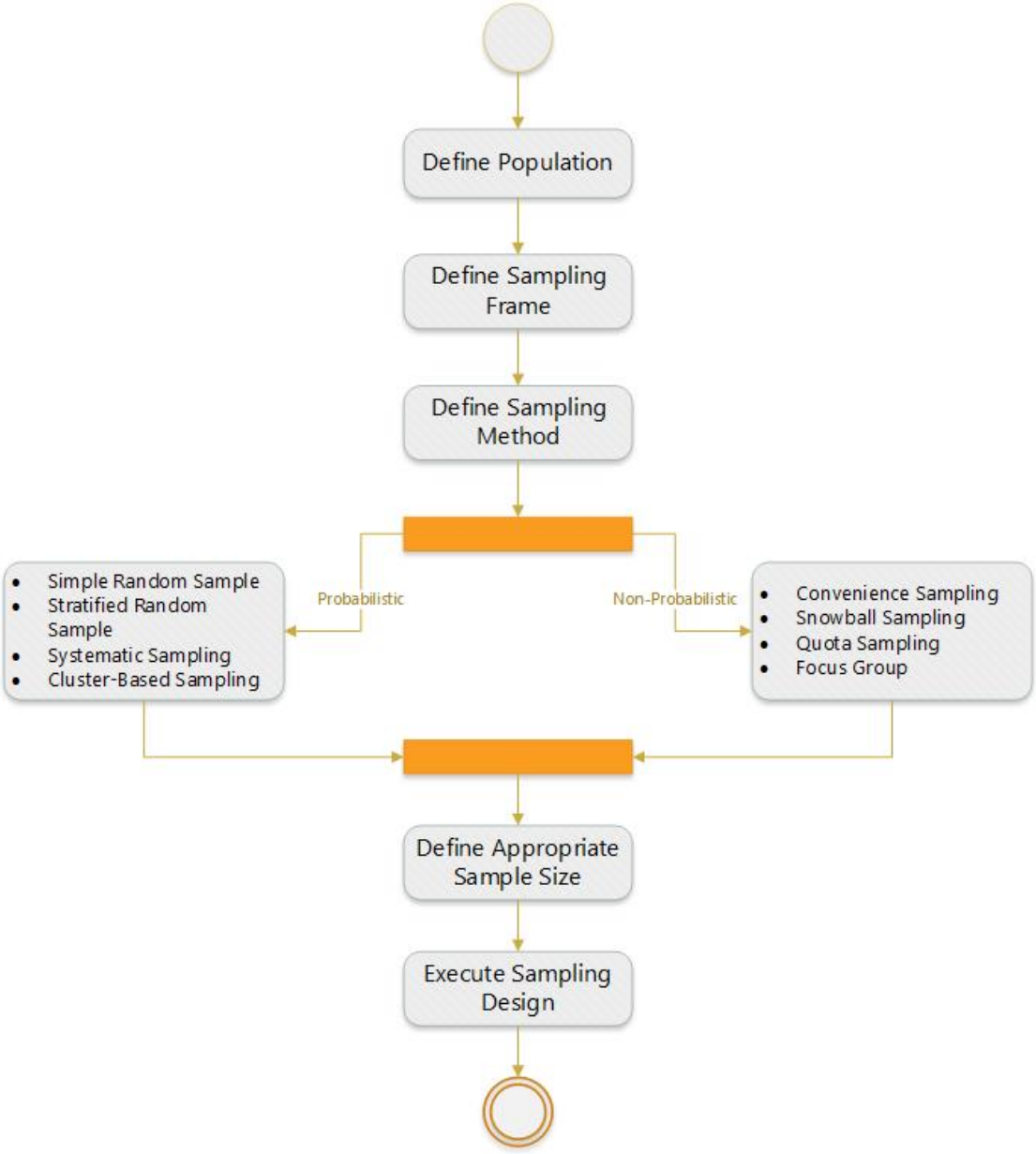


Figure 4.5: Sampling Activity Diagram

4.8.1 Sampling Methods

A probabilistic sampling method helps making reliable inferences about the target population, meaning, that we can draw statistical inferences from such samples. This

is a method where anyone in the population can be chosen to participate in the survey. All the members of the target population have a known, non-zero probability [27] of being chosen and to enter the survey population sample. This way we can claim, that the sample is truly a representative one and it eliminates any chance of the survey results being deemed biased by the scientific community. Table 4.3 describes a set of probabilistic sampling methods.

	Definition
Simple Random Sample	Subjects are selected from a list of the population at random.
Stratified Random Sample	The population is divided into a number of groups or strata with a known distribution between groups. Random sampling is then applied within the strata.
Systematic Sampling	The first subject is selected from the list of the population at random and then every n:th person is selected from the list.
Cluster-Based Sampling	Term given to surveying individuals that belong to defined groups.

Table 4.3: Probabilistic Sampling Methods

Non-probabilistic sampling methods are more commonly used, when either the target is very small or when we are doing a pilot study to our survey. These methods have a high risk of creating biased results, considering that we can't guarantee that the answers represent all the population. So, it's very risky to draw any type of inferences. See Table 4.4 for examples of non-probabilistic sampling methods.

	Definition
Convenience Sampling	The nearest and most convenient persons are selected as subjects.
Snowball Sampling	Involves asking people who have participated in a survey to nominate other people they believe would be willing to take part. Sampling continues until the required number of responses is obtained. This technique is often used when the population is difficult for the researchers to identify.
Quota Sampling	The type of sampling is used to get subjects from various elements of a population. Convenience sampling is normally used for each element.
Focus Group	Formed by specialists of the problem known to the creator of the survey. Consists of 5 to 10 people and they are intended to represent a part of the population.

Table 4.4: Non-Probabilistic Sampling Methods

4.8.2 Calculate a Sample Size

While sampling, it's very important to determine the appropriate sample size. Unfortunately, there are three problems to take in account. First, if the sample size is too low it can lead to poor results. The results will not produce verifiable conclusions, which makes it impossible to generalize the results to all the population [27]. The second problem is that poor sampling of strata will damage the capacity to compare different subsets of the population [27]. Finally, there is the issue of how many risks we are willing to take. The fewer risks, the larger the sample should be.

The risk determining the sample size, is directly correlated with two factors: confidence level and precision range. If we want to have the minimum risk possible then we need to use a high level of confidence and a small interval for the precision range. The most often used confidence level used according to Renckly [46] is a 95 percent with a +- 5 percent precision level as the absolute minimum. This means that out of 100 questionnaires sent, it is expected that 5 are unanswered.

Once the level of precision and confidence is chosen, we can use one of the formulas in Table B.2[4] to determine the sample size that we are going to use. The first formula is used if we want to show the results as percentages, while the second one is used if we show them in averages of the responding sample. The latter formula is chosen, in case we want to report said result in multiple ways, or if we are (for some reason) unable to use one of the first two formulas.

After using one of the formulas from the Table B.2 to discover the computed sample size (n), we proceed by adjusting it. This adjustment takes place by dividing the sample size by the awaited response rate, this way getting the optimal sample size. If it's not possible to anticipate the response rate, we should assume that it will be at least 50%. This adjustment should be able to ensure that the questionnaire will get enough responses without the need to care about the rate expected.

4.9 Instrumentation

The starting point to build a questionnaire is by defining the purpose and objectives. After gathering that information, we start to decide which areas we are going to cover. After that, we start creating the questions and choosing which response format is better suited for each of them.

[4] Z Factor is chosen depending of the confidence level we wish to apply. To find out more about it see Table of Z Values in Appendix B - Table B.1, from the book by Renckly [46] – page 57.

Besides creating the questions, this process also involves other factors we need to pay attention to. One is the number of questions we are going to ask. If it is too high, that can lead to the respondents to give up on the questionnaire. The other is to take special care in how we are going to organize them.

At the start of the questionnaire, we should write a short explanation about the purpose of the study, also known as a cover note. That explanation will also provide a realistic estimate of the time needed to complete the questionnaire, and the means for the respondent to contact the author of the questionnaire, in case of doubts or questions. Afterwards, we can use the following pattern proposed in the book by Renckly [46] in how to structure the questionnaire.

- The easier questions should appear at the beginning, and they should spark the interest of the respondents.
- Group related questions together.
- Demographic/background questions should be placed at the end of the questionnaire.
- Interpolate general questions with specific ones. This way the respondents don't get accustomed to a pattern.

4.9.1 Create the Questions

Once we know what we want to ask, we need to think of how we are going to pose the questions. The book by Renckly [46] offers some helpful hints on how to create a question:

- Keep the language simple, so we can avoid any misunderstanding.
- Keep the questions short to avoid ambiguity, and causing the respondents to be confused.
- Each question should only treat one concept at a time.
- Try to avoid the use of "should" and "would" questions. Should questions appeal to the respondents social/moral opinion, and the would questions to their personal preference.
- Use some of the questions as an accuracy and consistency control.

- Don't use questions that may drive the respondent to some specific answer. That means, that the questions shouldn't be worded in a suggestive manner.

- Give the respondents all possible options for a response, including the case that they don't know the answer or simply don't want to answer.

After creating the questions, we need to worry about what's the most adequate response format for each question. That format should make the question easier to answer without consuming more time than is desirable.

4.9.2 Response Formats

To reduce the necessary time to complete the questionnaire, and help with the treatment of the collected data, it's preferable to standardize the response format. If the responses are created in a standardized mode the surveyed population will know, beforehand, what are their options for answers. This will, in the end, make the process of reading and completing the questionnaire more accessible and faster for the surveyed population [28].

There are many response formats available. These formats can go from the typical 'yes' or 'no' questions to open-ended questions. Next, we present some of the most common response format used in surveys to collect data.

- **Ranking format** is used to prioritize what are the most important choices to the respondents, after being given a list of choices.

- **Selecting options format** gives a list of options to choose from, and then, respondents choose one of these options.

- **Open-ended format** are used when we want their full opinion without the risk of it being biased.

- **Rating scales format** can become difficult to treat, if the endpoints aren't equal and opposite since they risk having biased responses. The most common and easy to use is Likert scale. Likert scale have, at least, five different degrees of feelings, i.e. *strongly agree*, *somewhat agree*, *neither agree nor disagree*, *somewhat disagree* and *strongly disagree* [46].

4.10 Data Collection

A questionnaire has multiple ways to be delivered to its target population. It can be done through interviews, mailed, or using modern technologies such as the internet. In the first two options the questionnaire is normally presented in paper, although in the interviews case it also can be done through an online questionnaire. See Table 4.5[5] to see a small comparison between the types of delivery systems.

When the researcher chose to create, and deliver its questionnaire through the mail, the advantages of that method is the lower cost and the absolute guarantee of anonymity. Unfortunately, it also brings certain disadvantages like lower response rate, since we can't ask them to repeat the questionnaire, and lower control of the sample [52].

If the researcher decides to use the internet as a delivering system, then he has two options, either deliver the questionnaire using an e-mail or broadcast the message with the link to the questionnaire in a social network. If the delivery its done through email with either a link to the questionnaire or the questionnaire itself, then it is directly sent to the participant. However, if the researcher chooses to broadcast message type of delivery then he must choose a social network like LinkedIn, or Facebook, or a specialized forum where the target population interacts.

Factors	**Questionnaire**		
	Mailed	**E-mail**	**Broadcast in Social Network**
Cost	High	Low	Low
Application Time	Medium	Low	Low
Sample Control	High	Medium	Low
Response Rate	High	Medium	Low
Direct Contact with Participant	High	High	Medium
Contact Group Administrator	None	None	High
Help to spread the questionnaire	None	Low	Medium
Use of online software to create questionnaire	Low	Medium	High
Automatic partial treatment of the data	None	Medium	Medium

Table 4.5: Conditions affecting the choice of questionnaire delivery

4.11 Survey Software

After creating our questionnaire, we need to choose the best way to deliver it to the participants. Our approach to the delivery is done through the internet, and for that we need to select the most adequate survey software for our needs. That selection

[5]Table was partial created with the help from *Quadro 5*, page 13 from Vasconcellos and Guedes paper [52].

involves many factors such as, number of questions, expected number of responses, money available, among others.

These factors may influence how we choose the best survey software for our research. The following table (Table 4.6) compares Survey Monkey, Google Forms, Survey Gizmo and Qualtrics.

	Survey Monkey	**Google Forms**	**Survey Gizmo**	**Qualtrics**
Free Version	10 questions 100 respondents 15 questions types No exporting data	Unlimited surveys and questions Unlimited respondents Survey answers are automatically collected in Google Spreadsheets Add images or videos Skip logic and page branching Imbed survey into emails or website Add collaborators	Unlimited surveys and questions 50 respondents Several basic question types Basic reporting Option to export to CSV	100 Responses 10 Outgoing Emails 8 Question Types One Active Survey Unlimited Questions Summary Reports Filtering Survey Logic Randomization Online Reporting
Paid Version	More questions More respondents Exporting data	-	Skip logic and piping Multiple users API Data encryption Integration with 3rd party software tools	Multiple active surveys CSV/SPSS export More responses
URL Link	`https://pt.surveymonkey.com/`	`https://docs.google.com/forms/`	`https://www.surveygizmo.com/`	`https://www.qualtrics.com/`

Table 4.6: Comparing Survey Software's

For our study, the software chosen to create and to help us deliver our questionnaire was the paid version of Qualtrics.

4.12 Analysis Procedure

The web application used to create our questionnaire has built-in support to calculate the descriptive statistics with a complete report. The report contains graphics and tables with the respective information, such as average, variance, standard deviation and total responses and respondents per question. Another functionality is the support to cross-tabulation analysis. Cross-tabulation (or contingency table analysis) is a two or more-dimensional table that records the number of participants that have the same specific answers, thus providing information about a relationship between variables. This analysis is normally used in questions with answers in nominal scale. Finally, all of the data collected can be imported to SPSS 23.0, directly through a file with '.sav' extension. SPSS will be used to perform all the other statistical analysis like internal consistency and hypothesis testing.

4.13 Validity Evaluation

There are a few aspects in a questionnaire that can give rise to threats to its validity. For that reason, we need to verify that the questionnaire does measure the characteristics that are intended to be measured [26].

Campbell and Stanley defined two types of threats to validity, while Cook and Campbell extended that list to four types of threats. The four threats are conclusion, internal, construct and external validity [57][14]:

- **Conclusion validity** is occasionally indicated as a statistical conclusion validity. Threats to this validity are associated to problems that affect the capability to draw any type of conclusion between the treatment given to the data collected, and the outcome of the experimentation. These problems can be caused by the wrong choice of statistical test, or wrong sample size.

- **Internal validity** concerns the study design, particularly the research instrument used, and whether or not the results follow from the data. The most common errors include choosing the wrong statistical analysis, or failing to handle the variables properly.

- **Construct validity** relates to threats to how the experiment context reflects the questionnaire created, that is being study.

- **External validity** is related to conditions that may limit the generalization of the results gathered from our experiment. This tends to happen because we may have problems in sampling used.

Finally, we have another threat to the validity that our research may suffer, **reliability**. Reliability focus on whether our research results are the same, when others researcher replicate our study. In the case of not happening, it can mean that the research introduced any form of bias in the research [14].

4.14 Conclusion

This Chapter presents a blueprint for our survey study, as well as a study about how we can organize/prepare a survey-base empirical study. All the options chosen for the creation and execution of our study can be seen in the following table.

Research Paradigm	Explanatory Research
Design Type	Fixed Design or Quantitative Research
Type of Empirical Study	Survey
Survey Design	Descriptive Design: Case Control
Scientific Method	Hypothetic-deductive Positivist
Participants	Social Networks Communities: LinkedIn and Facebook; Researchers Community (Created through a filter)
Instrumentation	Questionnaire
Questionnaire Delivery	Broadcast in Social Network; E-mail
Survey Software	Qualtrics

Table 4.7: Chapter Summary

Table 4.7 presents in a condensed way, all the strategies and procedures that we will use during the execution of our survey study.

Chapter 5

Execution

5.1 Introduction

This chapter describes each step in the production and execution of our questionnaire. At first, we explain how we divided the questions per groups, what type of response format we selected and which questions we choose for internal validation of the questionnaire. The next section refers to the sample methods and communities used during the execution. The other sections are related to the pilot test phase, the actual execution of the questionnaire and the data collected from it. At last, we identified possible the threats to validity of our study.

5.2 Questionnaire Structure

This section explains the process we followed to build our questionnaire. Here we explain how the questions were created, and how we structured the questionnaire.

The creation phase of the questions started with a brainstorming for questions related to each research question and further refined sub-questions. This process took some iterations until we stabilized the set of questions that appeared in pilot test version of the questionnaire. After that, we decided which response format was the most adequate for each question.

For response format, we choose to use mostly a five point Likert scale. That scale

goes from 'strongly agree' to 'strongly disagree', to make easier and faster for the participants to choose the correct answer. Regarding the questions about the participants' background the response format chosen is selecting options, where some of them are either multiple or single answer.

After selecting the questions and their respective formats, we start to build the proper questionnaire in Qualtrics. The first thing we did was choose the most important questions to show up in our questionnaire. Out of 66 questions created, we decided to include 37. This process occurred by seeing which of the questions were the most important per research sub-question. See Appendix E -Table **??** for the relation between the questions in the questionnaire and their respective research sub-question, and the same in Table **??** for the questions that were left out of the final version of the questionnaire.

The other factor behind the choice we made was to have some questions for internal validation of the answers gathered, accuracy and consistency control. Examples of that from the final version of the questionnaire are shown next:

- **1.** I usually find tangling of concerns in the same function or m-file, such as described in the example provided.
- **7.** How often do you see examples of tangling in your code?

- **6.** When working in a MATLAB program, I try to avoid duplicated code whenever possible.
- **13.** When I program in MATLAB, I do not make an effort to eliminate duplicated code.

- **8.** I often think in terms of modularity when programming in MATLAB.
- **10.** When I program in MATLAB, I try to divide my code in small modules (functions or m-files) as a strategy to mitigate complexity.

- **17.** In the past, I had to maintain a MATLAB program for a long period of time.
- **20.** My MATLAB programs are mainly developed for solving short-term problems.

- **18.** Sometimes I feel the need to have a tool that helps visualize the structure of my MATLAB code.

- **22.** The dependency report tool offered by MathWorks is enough to visualize the structure of my code.

Concerning the questionnaire structure, we decided to put 5 questions per page with a page break at end of each group of questions from the same topic without allowing participants to move backwards. This preventing was implemented after receiving the feedback from Professor Glauco Carneiro (Appendix D) concerning our pilot test, where he said that this ability would help us validate internally the answers, since the participant cannot go back and change our answers. The main topics and their relation to the research questions described in the section 1.4, can be seen in Table 5.1.

Questionnaire Sections	**Number of Questions**	**Research Questions**
Tangling/Scattering	13	2 & 3
MATLAB Legacy Code	9	1 & 4
Background Information	15	3

Table 5.1: Relation between the questionnaire and the research questions

Figure 5.1, describes the above paragraph by means of an activity diagram.

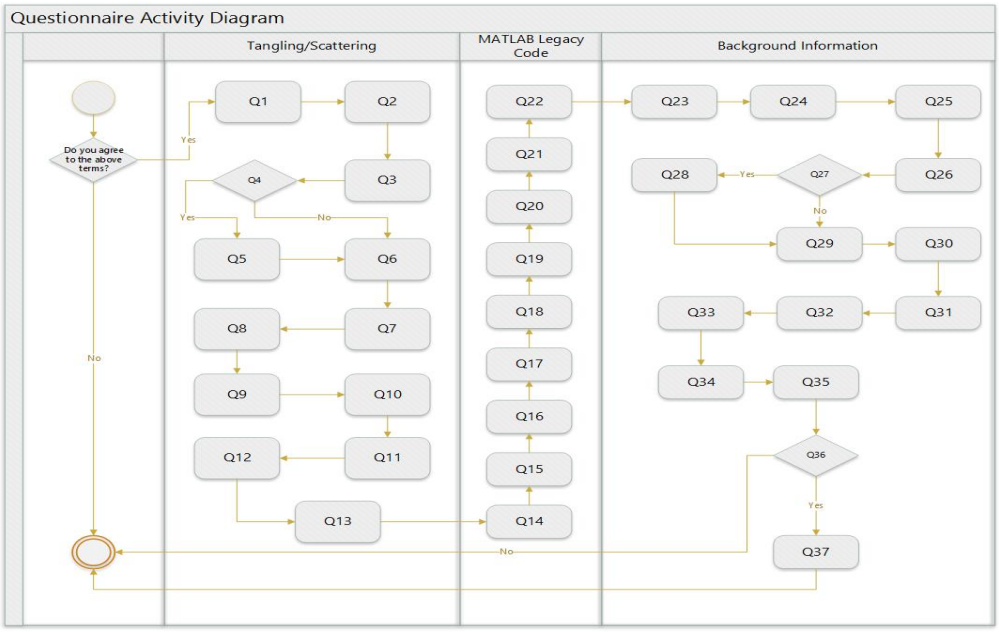


Figure 5.1: Questionnaire activity diagram

5.3 Sample

The study had two different phases in the creation and delivery of the instrument, pilot test and the actual execution of the questionnaire. That led to two separate phases for our sampling study.

In the pilot test, we used the Focus Group sampling method. The sample was composed by 12 developers with extensive knowledge of MATLAB and, in some cases about the limitations in the support to modularity. These developers are University Professors from FCT-UNL (mainly from the Electrical Engineering Department), Computer Science and Telecommunications Department in ISCTE-Instituto Universitário de Lisboa (ISCTE-IUL), Faculdade de Engenharia da Universidade do Porto (FEUP) and from other Universities.

Regarding the execution of the questionnaire, we started by defining our target population. In our case is, all the MATLAB developers, in either the industrial or academic world, with recent experience programming big projects. Unfortunately, we found out that, there wasn't enough reliable information in the MathWorks site or any other forum, that gave us those numbers and contacts. Faced with this difficulty, we decided to use a frame population, i.e., a discriminated group where we can find

elements from our target population. The frame populations used in the study, that fit most of our requisites for our initial target population, are the byproduct of the research done and described in Section 4.7. That research gave us a list of communities (see Table 4.2) available in popular social networks, where the participants are MATLAB developers, even if with different backgrounds and experience levels.

The communities used for the study are the MATLAB Users and Integrators and Matlab from LinkedIn, and MATLAB from Facebook. The choice behind these three communities refers, not only to the fact that they are the biggest communities in their respective social network, but also because they (at the time of the study) were the communities where the members interacted more with each other. That gives the post a greater chance of being watched/seen by members the post inviting them to participate in our study, and in turn a higher number of answers. This type of questionnaire delivery system (message broadcast through social network) means that we can't say for sure that our sampling is probabilistic or non-probabilistic, making this a *quasi-experiment*.

For the answers gathered from these three communities to be considered verifiable conclusions, regarding the frame population in question, we calculated the optimal sample size. For MATLAB Users and Integrators, we need at least 400 answers and in the Matlab one, we need 374. The MATLAB community in Facebook, we need at least 382. These values are calculated based on the 3^{rd} formula in Table B.2, as you can see in Table 5.2.

	MATLAB (Facebook)	**MATLAB Users and Integrators (LinkedIn)**	**Matlab (LinkedIn)**
Data	n = ? N = 73 769 (frame population size) d = +- 5 = 0.05 (precision level) Z = 95% = 1.96 (confidence level)	n = ? N = 24 978 (frame population size) d = +- 5 = 0.05 (precision level) Z = 95% = 1.96 (confidence level)	n = ? N = 13 792 (frame population size) d = +- 5 = 0.05 (precision level) Z = 95% = 1.96 (confidence level)
Calculation	$n = \frac{73769 * 1.96^2 * 0.25}{(0.05^2 * [73769 - 1]) + (1.96^2 * 0.25)}$	$n = \frac{24978 * 1.96^2 * 0.25}{(0.05^2 * [24978 - 1]) + (1.96^2 * 0.25)}$	$n = \frac{13792 * 1.96^2 * 0.25}{(0.05^2 * [13792 - 1]) + (1.96^2 * 0.25)}$
Result	n = 382	n = 400	n = 374

Table 5.2: Sample Size Calculations

Another community used in this study is the academic one. In other words, specialists/researchers who have published or wrote/co-wrote papers related to MATLAB. This community was created through the filter described in Subsection 4.7.2. Considering that we created this community through a specific process of exclusion, the type of sample used is non-probabilistic and the specific method is *convenience sampling*. For this case, we used all the members of the community, which makes a total of 79 people.

The total sample for the pilot test is of 12 participants. For the questionnaire itself, from the four communities, the total number of expected participants is 1235.

5.4 Pilot Test

Pilot test entails trying out the survey instrument that we created, see Appendix C, before sharing it with our intended population. With this tryout, we expect to ensure that the questionnaire is easy to understand and will collect the correct data.

This phase of our research occurred between April 4 and May 17, 2016 with a population of 12 invited participants (see Section 4.7). Where those invited participants, created our focus group (see Table 4.4) and through the feedback received (see Appendix D) from them, we refined and improved our instrument. That final version of our questionnaire and the one used in the execution can be seen in Appendix G.

Furthermore, it was timed how long took the participants to complete the questionnaire. That information available in Appendix D, it gave an average of 10 minutes to complete the all questionnaire. However, since not all of the possible participants have the same level of knowledge that our invited participants do, we put the time frame to complete the questionnaire around 15 minutes. Finally, it's important to note that the data collected in this phase, is not used in the analysis described in Chapter 6.

5.5 Questionnaire Execution

This phase of our research comprises the execution of the questionnaire. In other words, the broadcast of the questionnaire through the communities chosen and that are described in Section 4.7.

The process of delivering the questionnaire, started right after we finished the pilot tests phase. Unfortunately, since we had to complete some steps before launching the questionnaire in a social networks community (see Subsection 5.5.1), the execution of our questionnaire only started in May 20 and it end in September 02, 2016. The dates referring to the launching and closing of the questionnaire for each community can be seen in Table 5.3.

	Start Date	**End Date**
MATLAB (Facebook)	15/06/2016	05/07/2016
MATLAB Users and Integrators (LinkedIn)	20/05/2016	02/09/2016
Matlab (LinkedIn)	12/07/2016	02/09/2016
Researchers	29/06/2016	02/09/2016

Table 5.3: Delivery dates of the questionnaire per community

Another thing we must consider in this phase is, how we are going to understand where the answers came from. In specific, which community gave that answer. For this point, we decide to use the final version of our questionnaire and replicate it for each community. This way, we can differentiate the answers per community and, even, compare answers between communities. The questionnaire, and all its versions, are made available through a different anonymous link. Table 5.4 shows the associations made between the community and its respective anonymous link.

	URL Link
MATLAB (Facebook)	https://iscteiul.co1.qualtrics.com/SE/?SID=SV_3xdwD37B9L4rjoh
MATLAB Users and Integrators (LinkedIn)	https://iscteiul.co1.qualtrics.com/SE/?SID=SV_eL1aYvlilL77IxL
Matlab (LinkedIn)	https://iscteiul.co1.qualtrics.com/SE/?SID=SV_31TKxNv4qEuIrVX
Researchers	https://iscteiul.co1.qualtrics.com/SE/?SID=SV_5dyExSnuTQrVEjj

Table 5.4: Questionnaire links per communities

In the next subsection, we explain in more detail the process for which we must pass, so we can distribute the questionnaire in our chosen social network communities. Although, we use another delivering method, that is sending an email with and invitation and the link towards the researchers/academic community created by us (see Subsection 4.7.2). We consider this method, the internet version of the traditional mode of inviting someone to participate in a study. The only thing you need to do is write a similar email to ours, see Appendix F.2, and send it to your community by email.

5.5.1 Social Networks

For each community, or in this case social networks, there is a certain protocol/pattern that helps us distribute our research instrument. In this dissertation, since we used two LinkedIn communities and one from Facebook, we will only explain the process for these two cases.

A common phase, to both social networks, is to do a similar research to the one represented in Subsection 4.7.1, and then ask to join those communities or groups. This way, we can see how the community members participate (in other words, their content), and how often. Only after gathering these data, combined with the total number of members (at that precise moment), are we able to choose our preferred communities to execute our questionnaire. After this initial process, our decisions on how to deliver the questionnaire changes depending on the social network in question.

In LinkedIn, the first step after reading the group rules (if there is any) is to ask the group administrator (or one of them, in case of multiple administrators) permission

to use his group in our research, with private messages. This step will bring a couple of benefits in the long haul. One of them, is the administrator help announcing our message and/or questionnaire. This process can be done through the broadcast of a personal message to each member or by highlighting (putting at the top of the group page) our post with the message and the questionnaire, making our questionnaire reaching most of the community participants. Finally, you only need to create the post with all the necessary information. See our example in Appendix F, Section F.1.

With Facebook communities, considering that the nature of this social network is less serious and professional than LinkedIn, there is no specific need to contact the group administrator to ask permission to use his group. So, the only thing we need to do is create our post with the invitation to our questionnaire plus the link to it, and keep repeating the process until we have all the necessary answers or the time has run out.

The post in both cases, should be short and clear, so it doesn't misguide potential participants. See our message in Appendix F.1. Finally, the post can be written in a more informal way in Facebook than in LinkedIn, because the different uses of the two networks.

5.6 Data Collection Performed

During the pilot test, all the invited participants accepted the challenge. Unfortunately, the same did not occur during the execution of the questionnaire (to read more about this limitation go to Section 7.4). Of the expected 1235 participants, only 42 decided to participate in our study by responding to our questionnaire.

The first community to which we broadcast our study was the MATLAB Users and Integrators. Unfortunately, even updating our broadcast message we had a poor reception to our request. So, we decided to try other options, like the two other communities from LinkedIn and Facebook and the research community. However, at the end of the questionnaire delivery, the community that had more participants was the one we chose to use initially.

In total, we had 10 partial and 32 complete responses, with one of the complete response being a negative one. In other words, a participant decided to not answer to our study after reading the description and terms proposed by us. These numbers make for a total completion rate of 76.19%. However, since response rate is calculated when we have a definitive number of participants approached (something that happens when the questionnaire is broadcast through email), we can't calculate. This occurs because 3

of 4 communities used are social networks, and the participants are the ones who chose if they want to respond or not.

Table 5.5 describes the response (when available) and completion rate for each community, plus the anticipated values of participants and the number of complete and partial responses. The formulas used for the calculations performed in this Section, can be seen in Appendix B – Table B.3.

	Sample (Participants)	Complete Responses	Partial Responses	Response Rate (%)	Completion Rate (%)
Pilot Test	12	12	0	100	100
MATLAB (Facebook)	382	3	3	-	50
MATLAB Users and Integrators (LinkedIn)	400	20	6	-	76.92
Matlab (LinkedIn)	374	4	0	-	100
Researchers	79	5	1	6.33	83.33

Table 5.5: Response and Completion Rate

5.7 Threats to Validity

During the execution of our research study there were some deviations regarding the number of communities used and participants' distribution. We initially expected a much higher response and completion rate for each community that received the invitation to participate in our survey. These were discussed above in Section 5.6.

During and after closing the questionnaires in the communities that we deliver it, we noted and it was noted (in one case) by a participant, the following threats to validity of our instrument and possibly the final results:

- **Ineffective explanation.** One of our participants with a higher understanding of MATLAB and software engineering/modularity concepts, pointed out that our explanation for 'concerns' was too generalized. That could create a possible misunderstand of what are the limitations and the concerns. Although that is true, we tried to make the explanation as simple as possible, so people with less knowledge regarding software engineering or modularity could still participate. This problem is related to the construct validity.

- **Ambiguous questions.** Another threat that our participant pointed out was that, some of our questions can be considered ambiguous. Those concerns came up in questions that didn't specify whose code did the participant worked. This problem is related to the construct validity.

- **Lack of motivation, concentration or reading/understanding ability.** The number of answers previewed in the beginning of the execution, for each community, in Section 5.3, was not met. We only had 41 responses and out of these answers, 10 were partial responses. This can mean that our questionnaire was that either too long (37 questions) or the type of language was too technical in some cases. This problem is related to reliability.

- **Participants experience.** The only community that we knew beforehand, what kind of experience they had was the researchers' community. The participants from the other three communities in social networks, although they have experience programming MATLAB, they may not have experience regarding concepts about modularity or/and software engineering. So, the participants experience is a significant factor for the understanding of the problem and type of answers that they give. This threat is related to the internal validity of our study. Also, the varied participant experience can be considered a positive trait, since it enables us to do cross-tabulation between the research questions and factors related to the participant experience, work areas, among other factors.

- **Low and restricted number of participants/responses.** This problem may contribute to results that don't represent our population, making this a threat to external validity. Since we applied convenience sampling and quasi-experiment instead of a probabilistic method of sampling, we may not be able to extend our results to the entire population. However, our sample was not restricted to a group of developers with specific skills/characteristics.

5.8 Conclusion

The execution generally proceeded well and as initially planned but suffered from lack of willing participants from all the communities contacted through the questionnaire execution. An evaluation of the validity revealed that some of the participants may not have understood all the concepts pertaining the problem we are researching or there was some ambiguity in the questions. Those threats may have been the reason behind the lack of answers and the existing of 10 partial responses in 41.

Chapter 6

Analysis

6.1 Introduction

This Chapter summarizes the treatments given to the collected data. Several analyses were made regarding the representativeness of the population and how the background of each participant affects their opinion regarding our problem.

Another part of our analysis, is the analysis of the internal consistency of our survey. That analysis is carried out by means of the process of comparing the answers given to certain questions. For more information, about which questions see Section 5.2, by each participant.

6.2 Internal Consistency

As described in Section 5.6, we have a total of 41s responses with 31 positive and complete responses and 10 partials. The first step in our data analysis is to verify its internal consistency. The questions involved in this analysis will be only the questions that have Likert-scale as a response format, i.e., question 1 through 22 except 4 and 5.

We started by running principal component analysis test. This test is used is *to reduce a larger set of variables into a smaller set of 'artificial' variables, called 'principal components', which account for most of the variance in the original variables* [23]. At this phase, we only will use this analysis to help see if we need to change the scale used in any question. After a couple of iterations, we concluded that the questions 13, 15, 16, 20

and 22 had to invert its scale. This scale inversion occurred because our questions were created in the negative. In other words, instead of asking the participant if he agreed with something, we would ask if he didn't agree. This is a common method to help the researcher verify if the participant was paying attention to the questionnaire or not.

After this process we were able to start calculating the Kendall tau rank distance between the scrutiny questions described in section 5.2

6.2.1 Kendall tau distance

Kendall tau rank distance (also known as bubble-sort distance) is a metric that counts the number of pairwise disagreements between two ranking lists created by Maurice Kendall. The bigger the distance between the lists, the most different they are. The mathematical formula is the following:

$$K(t_1, t_2) = |(i, j) : i < j, (t_1(i) < t_1(j) \wedge t_2(i) > t_2(j)) \vee (t_1(i) > t_1(j) \wedge t_2(i) < t_2(j))|$$

Our adaptation of the Kendall tau rank distance metric can be seen in the following pseudo-code. In this case, as we want to compare the answers given between the screening questions described in section 5.2, and we have 10 cases where the participants gave up in the middle of the questionnaire, our formula is as follows.

```
mean(kendall(X, Y) = X[i] && Y[i] ? abs(X[i]-Y[i]) : 0)
```

We first verify that there is answer to both questions, if not then the distance is considered zero. If it's true, then the distance is the absolute value of the difference between the value in the first question and the value in the second question. These results can be observed in the columns colored in orange, in Figure 6.1.

PARTICIPANT ID	VAR01	VAR07	kendall(VAR01, VAR07)	VAR06	VAR13	kendall(VAR06, VAR13)	VAR08	VAR10	kendall(VAR08, VAR10)	VAR17	VAR20	kendall(VAR17, VAR20)	VAR18	VAR22	kendall(VAR18, VAR22)	Σ (kendall results per row)
1	5	2	3	1	5	4	5	5	0			0			0	7
2	4		0			0			0			0			0	0
3	5		0	4	3	1	4	4	0	1	2	1	3	3	0	5
4	3	2	3			0			0			0			0	0
5	4	3	1	5	2	3	5	5	0	1	4	3	1	3	2	9
6	4	3	1	5	5	0	4	5	1	4	2	2	4	3	1	5
7	4	4	0	5	5	0	5	5	0	4	2	1	5	2	3	4
8	4	4	0	5	5	0	5	5	0	4	3	1	2	3	1	2
9	1	1	0	5	5	0	5	3	2	5	5	0	1	3	2	4
10	3	3	0	5	4	0	4	4	0	1	2	1	2	3	2	4
11	5	2	3	4	3	1	3	4	1	5	1	4	3	3	0	9
12	4	3	1	5	5	0	1	2	1	4	4	0	4	3	1	3
13	1	3	2	4	5	1	4	3	1	4	3	1	4	3	1	6
14	5		0			0			0			0			0	0
15	5	4	1	1	1	0	2	2	0	3	2	1	5	3	0	4
16	3	2	1	4	4	0	4	4	0	3	2	1	4	3	1	3
17	4	4	0	5	4	1	5	5	0	4	5	1	5	3	2	4
18	4	4	0	5	5	0	4	5	1	3	2	1	4	2	2	4
19	4		0			0			0			0			0	0
20	4	2	2	4	2	0	4	4	0	4	2	2	3	3	0	6
21	5		0			0			0			0			0	0
22	4	4	0	4	2	2	3	3	1	1	2	1	4	4	0	4
23	4		0			0			0			0			0	0
24	4	4	0	4	4	0	5	5	1	3	1	2	4	3	1	4
25	4	3	1	5	5	1	5	5	1	4	2	2	4	3	1	5
26	4	4	0	5	4	1	4	4	0	4	3	1	3	3	0	2
27	5	5	0	4	4	0	2	3	1	2	1	1	5	3	0	4
28	1	2	1	5	5	0	5	5	0	5	4	1	5	3	0	4
29	5	4	1	5	5	0	5	5	0	5	4	1	5	3	0	4
30	3	2	1	5	4	1	5	4	1	5	5	0	5	3	2	5
31	3	2	1	5	5	0	5	5	0	4	3	1	4	4	0	2
32	5	5	0	4	2	2	3	1	2	4	2	2	3	3	0	8
33	3	3	0	1	3	2	4	5	1	3	2	1	3	2	1	5
34	5	4	1	4	4	0	4	2	2	4	5	1	4	5	1	5
35	4	2	2	3	3	2	5	5	0	2	4	2	1	3	2	8
36	4	4	0	5	4	1	5	5	0	4	3	1	5	2	0	5
37	4	4	0	5	4	1	5	5	0	4	3	1	5	2	0	5
38	4	4	0	5	4	1	5	5	0	3	2	1	4	4	0	3
39	4	3	1	4	3	1	4	5	1	4	2	2	4	3	1	6
40	5	5	0	5	2	0	1	1	0	5	5	0	5	5	0	3
41			0			0			0			0			0	0
42	3	5	2	2	2	3	4	2	2	3	2	1	4	3	1	9

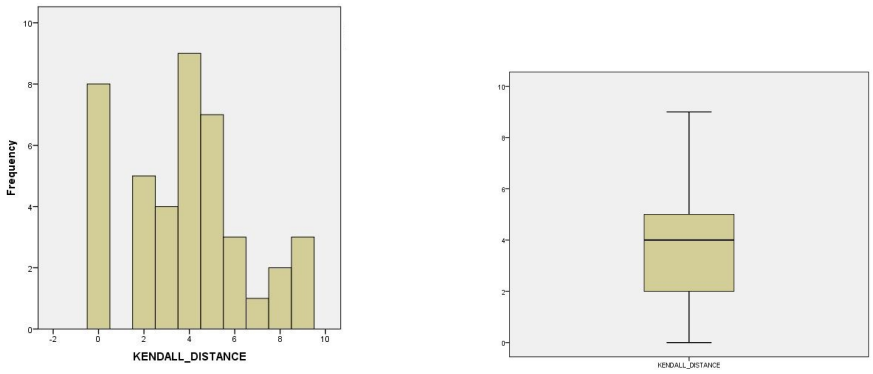
Figure 6.1: Kendall tau distance

Afterwards, we calculate the mean of all the values in each column with the Kendall tau distance metric results, to see if the mean is either 0 or 1. The results of this step in the metrics calculation can be observed in the next figure:

kendall(VAR01, VAR07)	kendall(VAR06, VAR13)	kendall(VAR08, VAR10)	kendall(VAR17, VAR20)	kendall(VAR18, VAR22)
0	0	0	1	1

Figure 6.2: Kendall tau distance results

According to these results, the answers given in questions 17 – 20 and 18 – 22 are not in the same order. This means that some participants responded at random in those questions and without certainty that they did the same in the other questions, we need to remove the responses of some participants. To see which participants answers we would remove, we created two graphs, a histogram and a scatter plot.



(a) Histogram (b) Scatter Plot

Figure 6.3: Results of Kendall Tau distance in graphs

The histogram and scatter plot in Figure 6.3 are the graphical representation of the results obtained in the last column of the table in Figure 6.1. The figures job is to help us decided which participants' responses we should remove, to make our survey more internal consistent.

The standard procedure for the removal of all cases is to remove those who are contained in the last quartile of the scatter plot (see Figure 6.3). That would be removing 25% of the responses obtained, making our sample go from 41 to 30. However, our available sample is already relatively small. Therefore, to minimize the number of responses removed and at the same time increase the internal consistency, it was decided to remove just 10% of the responses.

10% of our sample means removing the participants who didn't passed this formula:

$$\sum(\text{kendall results per row}) < 8$$

Applying this formula to our data, we manage to remove the responses of 5 (12%) participants, making our sample decrease from 41 to 36. The participants whose responses were removed from our sample have the following participant id: 5, 11, 32, 35 and 42.

6.2.2 Principal component analysis

After removing the five participants' responses from our data collection, we started the actual principal component analysis that we explained at the beginning of this section. For us to accept the result given by this analysis the data needs to first pass five assumptions. Those assumptions are:

1. Variables should be able to the measured at the continuous level, like ratio or interval or ordinal(Likert-scale) variables.

2. There needs to be a linear relationship between variables.

3. Sample adequacy.

4. Data should be suitable for data reduction.

5. There should be no significant outliers.

Our data failed assumption 3 when we run the test. One way to test assumption 3 is to run the Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy for the overall data set and the result should be bigger than 0.5/0.6. In our case the KMO is 0.264.

However, we still can factorize our set of variables into a smaller one. This can only happen since we were the ones who created and known the questionnaire, which enables us to make a good judgment in how to reduce the variables number. With this knowledge and with the help of the Cronbach test, we manage to reach a final reliable group of factors. See Table 6.1.

Factors	Variables/Questions
Code Tangling	VAR01; VAR02; VAR03; VAR07; VAR11; VAR12
Code Duplication Avoidance	VAR06; VAR13
Modularity	VAR08; VAR10; VAR19; VAR21
Code Structure	VAR14; VAR16; VAR18; VAR22
Clone Recurrence	VAR09
Code Maintenance	VAR15; VAR17; VAR20

Table 6.1: Variables grouped by factors

6.2.3 Cronbach's alpha

Cronbach's alpha (α) is an indicator of internal consistency and can be used to measure the reliability, in our case of the factors that we created and are shown in Table 6.1. A downside for us is that we need to have at least three variables to run it. This means that we can't calculate the alpha for the *Clone Avoidance* and *Clone Recurrence* factors.

A sufficient level of reliability as measured by the alpha is 0.7 or more. The result of the application of this measure in our factors can be observed in Table 6.2.

Factors	Variables/Questions	α
Code Tangling	VAR01; VAR02; VAR03; VAR07; VAR11; VAR12	0.737
Code Duplication Avoidance	VAR06; VAR13	-
Modularity	VAR08; VAR10; VAR19; VAR21	0.726
Code Structure	VAR16; VAR18; VAR22	0.511
Clone Recurrence	VAR09	-
Code Maintenance	VAR15; VAR17; VAR20	0.779

Table 6.2: Cronbach's alpha per Component

As we can see, for all the factors except *Code Structure*, the Cronbach's alpha is between 0.7 and 0.8. This indicates an acceptable level of internal consistency between our variables in each factor for this specific sample, according to Table 6.3.

Cronbach's alpha	Internal Consistency
$\alpha \geq 0.9$	Excellent
$0.9 > \alpha \geq 0.8$	Good
$0.8 > \alpha \geq 0.7$	Acceptable
$0.7 > \alpha \geq 0.6$	Questionable
$0.6 > \alpha \geq 0.5$	Poor
$0.5 > \alpha$	Unacceptable

Table 6.3: Cronbach's alpha intraclass correlation score

However, the *Code Structure* factor suffered some modifications when comparing with the original table regarding the number of variables/questions. This happen because when we run the Cronbach's alpha the result was unacceptable with 0.452. A way solve this problem, according to Table 6.4, was to remove VAR14 and rise the internal consistency of the factor from 0.452 to 0.511, making it poor.

	Cronbach's Alpha if Item Deleted
VAR14	0.511
VAR16	0.300
VAR18	0.179
VAR22	0.478

Table 6.4: Cronbach's alpha for *Code Structure*

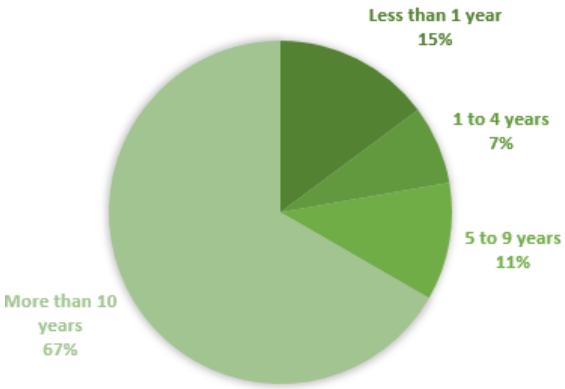
Unfortunately, VAR14 didn't fit in another factor without affecting the alpha. So, we decided to not include this question in our study.

6.3 Participants Profile

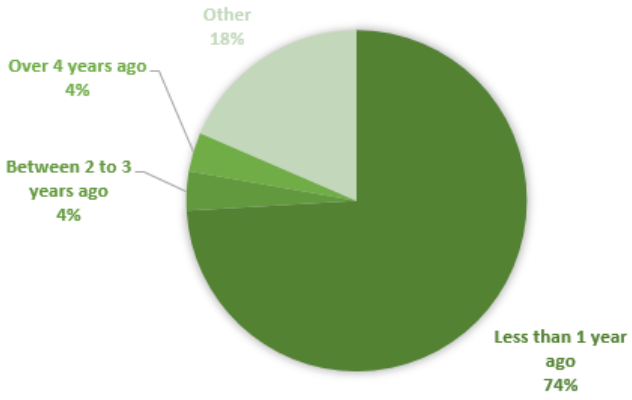
For this section, we used the available sample after the internal consistency process done to the data collected and described in the previous section, which is a sample of 36 positive answers. For this section in particular, we are going to only use the complete responses and the answers given to questions in the *Background Information* survey section (see Appendix G). This happens because we can't profile the participants who chose not to complete the survey. Our available sample, with these conditions, for this specific analysis is reduced from 36 to 27 responses.

More than half of our sample of 27 participants, 67% (18), have over 10 years of experience programming in MATLAB, while only 15% (4) have less than 1 year. The other 18% (5) have between 1 and 9 years of experience. Most of these developers

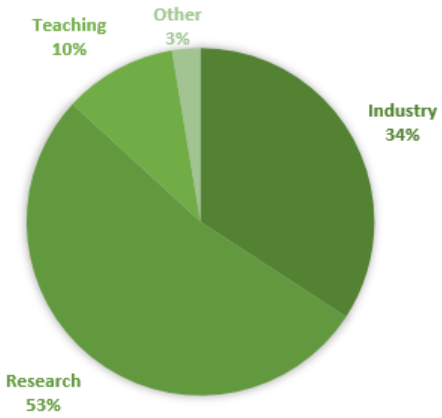
are still programming in MATLAB, especially in the research field (53% participants selected this option), with just 11.11% (3) admitting to not have used it in over 2 years. See the following graphs for more information.



Graph 6.4: Question 23 - *How many years of experience do you have programming in MATLAB?*



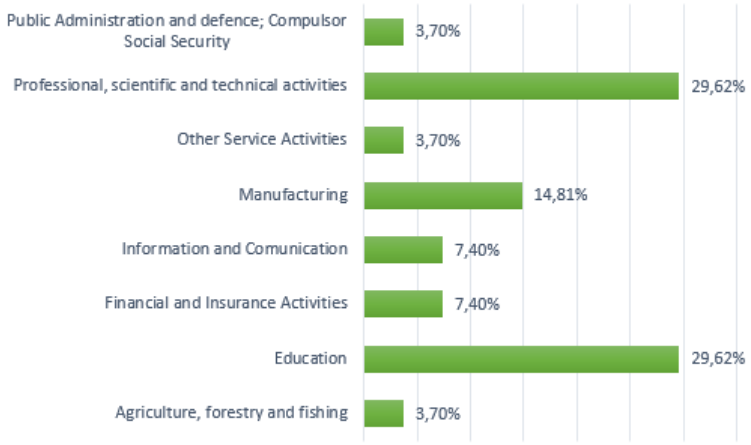
Graph 6.5: Question 24 - *Last time I programmed in MATLAB program was ...*



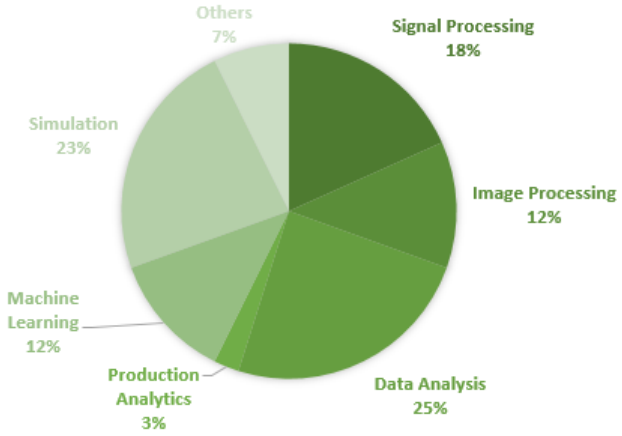
Graph 6.6: Question 29 - *How would you classify the nature of your work when using MATLAB:*

74.07% of participants out of 27 said that they don't work with a developing team when developing a MATLAB system. The ones that do worked in teams with 1 to 7 or more developers, besides them.

27 participants work, mostly (29.62%) in the *Professional, scientific and technical activities* and in the *Education* field, each. The other fields selected by our participants were *Manufacturing* with 14.81% (4), *Information and communication* with 7.40% (2), *Financial and Insurance Activities* with 7.40% (2), and *Public Administration and defence; Compulsory Social Security*, *Agriculture, forestry and fishing* and *Other Service Activities* with 3.70% (1) each. The type of work that they performed when developing in MATLAB, in their respective fields, is mostly data analysis with 25% (20) and simulation with 23% (15). These fields are closely followed by signal processing with 18% (15), image processing and machine learning with 12% each. Some of the participants mentioned performing other types of work when using MATLAB. These works are chip simulation, system identification, control system and modelling, optimization and financial modeling. See the next graphs for more detailed information.

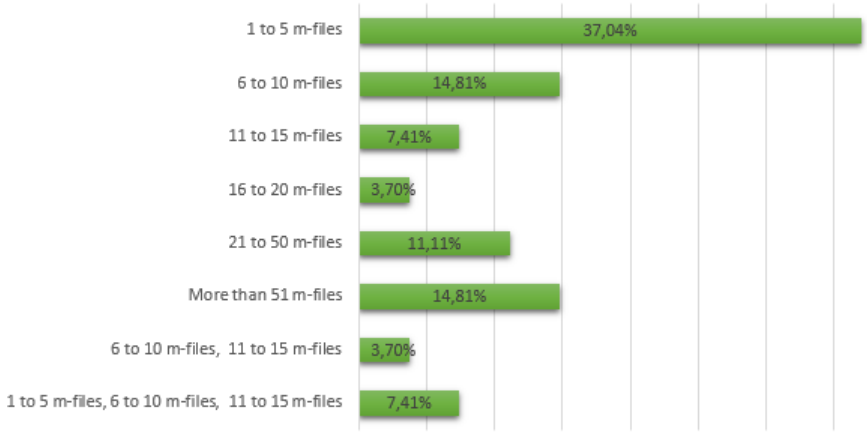


Graph 6.7: Question 33 - *Using the United Nations - International Standard Industrial Classification, where do you care out your work?*

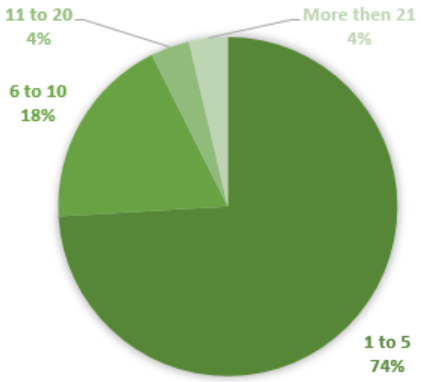


Graph 6.8: Question 34 - *I use MATLAB to perform this kind of work:*

Regarding the participants programming practices, most of them only deals with MATLAB programs with 1 to 5 m-files and toolboxes, closely followed by programs with 6 to 10 and 11 to 15 m-files (see graphs 6.9 and 6.10). In those m-files, 66.67% of the 18 participants said to work with 2 to 10 functions per m-file, while 18.52% usually work just 1 function.

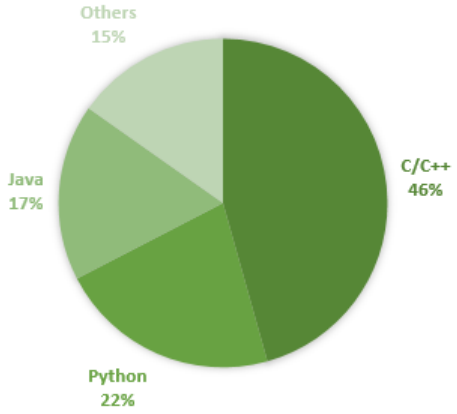


Graph 6.9: Question 25 - *I normally deal with MATLAB programs with...*



Graph 6.10: Question 31 - *How many toolboxes you tend to use?*

Finally, 88.89% (24) of our available sample (27 participants) admit using only MathWorks workspace to program in MATLAB, while 3.70% (1) says to only use GNU OCTAVE workspace. The rest of the participants uses a mix of MathWorks, GNU OCTAVE and Scilab workspaces, and all of them are familiar with other programming languages. 46% (21) of the participants claimed to be familiar with C/C++, 22%(10) with Python and 17% (8) with Java and other programming languages, each. Graph 6.11 depicts the frequency of the familiarity of our participants with these programming languages.



Graph 6.11: Question 37 - *Which languages are you familiar with?*

6.4 Descriptive Statistics

The following descriptive analysis is a continuation of the analysis performed in the previous section. In this section, the analysis will be done to the questions pertaining the 6 factors created in section 6.2, that belong to the first two groups of our questionnaire, *Tangling/Scattering* (1-13) and *MATLAB Legacy Code* (14-22).

The available sample for this analysis will vary between 36 and 28. We begin with a sample of 36 participants, but at end of the first page of the questionnaire 19.44% of the participants gave up, making the available sample go from 36 to 29 participants. Finally, the second group of questions (*MATLAB Legacy Code* group) has an available sample of 28 throughout all of its questions. These questions have all in common a Likert-scale composed by five Likert-type items. That scale for all the questions except for question 7, goes from "Strongly disagree" to "Strongly agree", while the scale from question 7 goes from "Never" to "Always".

Likert-type items fall into the ordinal measurement scale. Descriptive statistics recommended for ordinal measurement scale items include mode and/or median for central tendency and frequencies for variability. In this analysis, we use frequencies and percentages for measures of variability, along with the sample size for each question (to help reading the values). See Table 6.5.

	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree	Sample Size
VAR01	3 (8.3%)	0 (0%)	6 (16.7%)	18 (50%)	9 (25%)	36
VAR02	4 (11.1%)	4 (11.1%)	6 (16.7%)	12 (33.3%)	10 (27.8%)	36
VAR03	3 (8.3%)	3 (8.3%)	1 (2.8%)	18 (50%)	11 (30.6%)	36
VAR06	3 (10.3%)	0 (0%)	1 (2.7%)	9 (31%)	16 (55.2%)	29
VAR07*	1 (3.4%)	6 (20.7%)	8 (27.6%)	12 (41.4%)	2 (6.9%)	29
VAR08	2 (6.9%)	2 (6.9%)	0 (0%)	15 (51.7%)	10 (34.5%)	29
VAR09	4 (13.8%)	7 (24.1%)	3 (10.3%)	11 (37.9%)	4 (13.8%)	29
VAR10	1 (3.4%)	3 (10.3%)	4 (13.8%)	6 (20.7%)	15 (51.7%)	29
VAR11	2 (6.9%)	2 (6.9%)	7 (24.1%)	14 (48.3%)	4 (13.8%)	29
VAR12	3 (10.3%)	8 (27.6%)	6 (20.7%)	9 (31%)	3 (10.3%)	29
VAR13	12 (41.4%)	10 (34.5%)	2 (6.9%)	4 (13.8%)	1 (3.4%)	29
VAR14	11 (39.3%)	6 (21.4%)	6 (21.4%)	3 (10.7%)	2 (7.1%)	28
VAR15	7 (25%)	5 (17.9%)	6 (21.4%)	5 (17.9%)	5 (17.9%)	28
VAR16	6 (21.4%)	10 (35.7%)	5 (17.9%)	5 (17.9%)	2 (7.1%)	28
VAR17	3 (10.7%)	1 (3.6%)	6 (21.4%)	14 (50%)	4 (14.3%)	28
VAR18	1 (3.6%)	2 (7.1%)	4 (14.3%)	13 (46.4%)	8 (28.6%)	28
VAR19	0 (0%)	0 (0%)	2 (7.1%)	10 (35.7%)	16 (57.2%)	28
VAR20	5 (17.9%)	3 (10.7%)	6 (21.4%)	12 (42.9%)	2 (7.1%)	28
VAR21	0 (0%)	3 (10.7%)	2 (7.1%)	11 (39.3%)	12 (42.9%)	28
VAR22	2 (7.1%)	4 (14.3%)	18 (64.3%)	4 (14.3%)	0 (0%)	28

*This question Likert-type items are in this order: "Never", "Sometimes", "About half the time", "Most of the time" and "Always"

Table 6.5: Measures of Variability

As for measures of central tendency, we use all the available statistics, i.e., mode and median. In one case, *VAR20*, we don't specify since the value obtained is not an integer value. So, we cannot match a Likert- scale item. See Table 6.6.

	Median	Mode
VAR01	Somewhat agree	Somewhat agree
VAR02	Somewhat agree	Somewhat agree
VAR03	Somewhat agree	Somewhat agree
VAR06	Strongly agree	Strongly agree
VAR07	About half the time	Most of the time
VAR08	Somewhat agree	Somewhat agree
VAR09	Somewhat agree	Somewhat agree
VAR10	Strongly agree	Strongly agree
VAR11	Somewhat agree	Somewhat agree
VAR12	Neither agree nor disagree	Somewhat agree
VAR13	Somewhat disagree	Strongly disagree
VAR14	Somewhat disagree	Strongly disagree
VAR15	Neither agree nor disagree	Strongly disagree
VAR16	Somewhat disagree	Somewhat disagree
VAR17	Somewhat agree	Somewhat agree
VAR18	Somewhat agree	Somewhat agree
VAR19	Strongly agree	Strongly agree
VAR20	-	Somewhat agree
VAR21	Somewhat agree	Strongly agree
VAR22	Neither agree nor disagree	Neither agree nor disagree

Table 6.6: Measures of Central Tendency

6.5 Hypothesis Testing

For this part of our analysis, instead of using the questions related to each dependent and independent variable, we use the factors that we created in Table 6.2. To use these factors, we need to combine first the answers given to all the questions that belong to a component into one single variable, that will be then used in the test. The method used to create this new variable (with the same name of the component) was by using the next formula:

IF (MOD(MEDIAN(VAR[01,...,22])), 1) != 1)
MEAN(VAR[01,...,22]))
ELSE
MOD(VAR[01,...,22])

This process creates 6 new variables. These variables except *Code Structure* follow the next order *1* = *"Strongly disagree"* up to *5* = *"Strongly agree"*. In variable *Code Structure*, we had to invert this scale which means that in this case *1* = *"Strongly agree"* and *5* = *"Strongly disagree"*.

Regarding the type of hypothesis tests that can be done with variables, in our case we can only do non-parametric tests. This happens because our variables are ordinal ones with a discrete distribution, which invalidates the assumptions needed to run parametric tests. The hypothesis tests used in our analysis are *One-Sample Chi-Square*, *Spearman's Correlation* and *Mann-Whitney U*.

6.5.1 One-Sample Chi-Square Test

One-Sample Chi-Square (χ^2) tests the hypothesis that each item in the Likert-scale (or proportions), as likely to be selected as the others. A guideline for reading the results in Table 6.5.1 is provided here:

- Significance levels equal to or less than 0.05 indicates that there is a statically difference between the proportions.
- χ^2 value describes the test statistic for a χ^2 test. It is used to describe the shape of the distribution of the χ^2 test.

Hypothesis	**Independent Variable**	**Significance**	χ^2	**df**
1	Code Tangling	0.000	24.833	4
2	Code Structure	0.003	15.929	4

Table 6.7: Chi-Square Test Result

There were statistically differences between proportions at $p < 0.003$ level in both hypotheses.

6.5.2 Spearman's Correlation Test

Spearman's correlation test or Spearman's rho (ρ) is a non-parametric measure of rank correlation between two variables on at least an ordinal variable (or interval, or ratio

scale). This test assesses how well the relationship between the variables can be measured using a monotonic function. The test was named after Charles Spearman.

We only have one hypothesis that can use this test, hypothesis 3. This hypothesis has one dependent variable and one independent, that are the equivalent to two of our factors. Modularity is our independent variable, while code maintenance is the dependent one. A guideline for reading the results in Table 6.5.2 is provided here:

- Significance levels equal to or less than 0.05 indicates an evidence that the limitations in modularity affects the code maintenance.
- Value is the correlation strength between the two variables.
- Asymptotic standard error approximates the standard error, based upon some mathematical simplification.

Hypothesis	**Dependent Variable**	**Independent Variable**	**Significance**	**Value**
3	Code Maintenance	Modularity	0.069	0.349

Table 6.8: Spearman's Correlation Test Result

There were no statically significant at the $p < 0.05$ level.

6.5.3 Mann-Whitney U Test

Mann-Whitney U test is the non-parametric alternative version to the independent sample t-test. It is used to compare two sample means from the same population and see if they're equal or not. This test is normally used, when our variables are ordinal or the assumptions for the t-test are not met.

This test has a set of assumptions that need to pass, before we use it. Those assumptions are:

- Sample drawn from the population is random.
- Independence within the samples and mutual independence is assumed.
- Ordinal variables, only.

All these assumptions were checked before we started our analysis. A guideline for reading the results in Table 6.9 is provided here:

- U is the result to the calculation of the Mann-Whitney U, see the mathematical formula beneath, where R_i is the rank of the sample size, N_1 is the sample of the first group (industry) and N_2 of the second (research).

$$U = n_1 n_2 + \frac{n_2(n_2+1)}{2} - \sum_{i=n_1+1}^{n_2} R_i$$

- Exact Significance levels equal to or less than 0.05 indicate a significance difference between the participants that develop software in MATLAB in the industry vs. the ones who do it in the academic world.

					Mean Rank	
Hypothesis	**Dependent Variable**	**Independent Variable**	**U**	**Exact Sig. [2*(1-tailed sig)]**	**Industry (13)**	**Research (14)**
4	Modularity	Participant Background	61.5	0.155	16.27	11.89
5	Code Tangling		68	0.280	12.33	15.64
6	Code Maintenance		85.5	0.793	14.42	13.61
7	Code Duplication Avoidance		83	0.720	13.38	14.57
8	Code Structure		89	0.943	14.15	13.86

Table 6.9: Mann-Whitney U Test Results

None of the hypothesis tested using the Mann-Whitney U test had a significance level beneath 0.05.

Chapter 7

Conclusions and Future Work

7.1 Summary

The background and one of the hypothesis for this research was to evaluate if the MATLAB developers feel/see the outcome provoked by the limitations in the support to modularity offered by the language and its programming paradigm (PP). A survey-based empirical study was designed and executed to validated our hypotheses. The limitations that we refer to, in this dissertation and the questionnaire, are code tangling and code scattering. However, in the questionnaire code scattering was translated to duplicated code. To be noted that these two concepts are not synonyms of each other, duplicated code is a symptom of code scattering. This exchange of concepts, only happened because we though that our target population would not have an extensive knowledge of software engineering and, code scattering is not an easy concept to explain in a short introduction.

A review of the existing literature or related work was not found, considering that it's the first time a study with this problem was done. However, we did a programming language review regarding MATLAB and a brief comparison between MATLAB and one of its clone language, GNU Octave. This helped us defining our target population, because instead of only searching for developers that used MATLAB in the MathWorks workspace, we also accepted the ones who used a clone language. Another advantage brought by this review, was the discovery that even though MATLAB is a

multi-paradigm language, the main paradigm (and the most used by the developers) is the procedural programming paradigm. This leads us to another review and related work research regarding the levels of modularity offered by a procedural language comparing to an oriented-object one, and the impacts they have in the modularity of the language. In this topic, we explained in more depth the concepts of code tangling and scattering, and we instantiate the procedural version of MATLAB to the PIMETA meta-model.

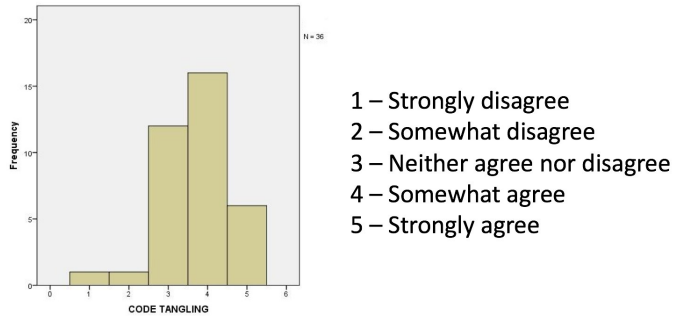
Our survey research used as instrument the questionnaire and the delivery method chosen were post the invitation to participate in our study in the communities of a social networks, more specifically LinkedIn and Facebook, and through email towards the researchers' community. The social network delivery method was chosen as way to reach a wider number of developers/possible participants. Unfortunately, out of the four communities used we only received 42 responses. Out of the 32 complete responses received, 15 participants sent comments and expressed their interested in learning about the results and/or participating in future sections of the research.

7.2 Results

The following conclusions were drawn in respect to hypothesis created in Section 1.5, together with the Chapter 6. Unfortunately, we can extrapolate these results to the general population, i.e., all the MATLAB developers because of the existing limitations in the research. **Hypothesis 1.** MATLAB developers don't find or care about tangling of concerns.

A one-sample chi-square test was used to analyze if proportions for each Likert-scale item were uniforms, i.e., all the possible options had the same number of responses. We could demonstrate that the five possible answers are not equally attractive to the MATLAB developers, rejecting the test null hypothesis; $\chi^2(4) = 24.83$, $p < .001$.

If we compare this test result with the histogram in Figure 7.1, then we can safely say that we reject *Hypothesis 1*. This means that MATLAB developers do find and care about code tangling.

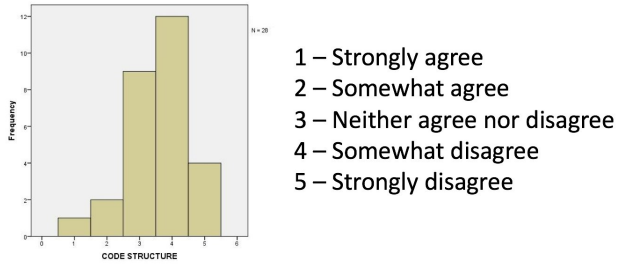


Graph 7.1: Code tangling histogram

Hypothesis 2. While developing, be either creating or maintaining a system, MATLAB developers never felt the necessity of using a tool to help visualize the code structure.

A one-sample chi-square test was used to analyze if proportions for each Likert-scale item were uniforms, i.e., all the possible options had the same number of responses. We could demonstrate that the five possible answers are not equally attractive to the MATLAB developers, rejecting the test null hypothesis; $\chi^2(4) = 15.93$, p = .003.

If we compare this test result with the histogram in Figure 7.2, then we can't reject the hypothesis we initially proposed because the majority of the developers disagree with the statement. However, if we take in account the answers (see Tables 6.5 and 6.6) given to question 18 - *Sometimes I feel the need to have a tool that helps visualize the structure of my MATLAB code.*, is possible that the MATLAB developers don't feel that the MathWorks tool for visualizing the code structure is enough.



Graph 7.2: Code structure histogram

Hypothesis 3. There are no evidences that the limitations in the support to modularity in MATLAB, affect the maintainability (and readability) of the code.

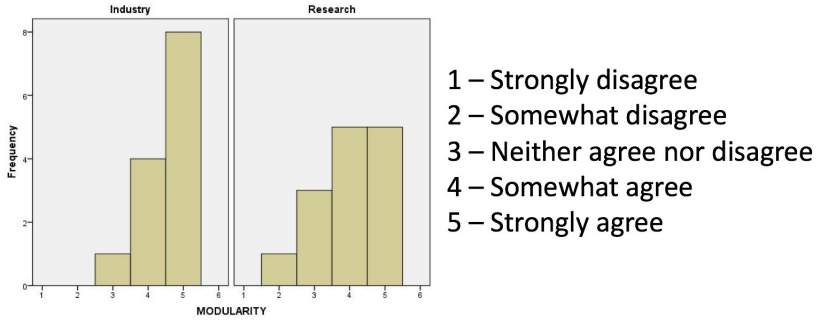
A Spearman's rank-order correlation was run to analyzed whether the code maintainability was affected by the modularity limitations in MATLAB. The result obtained was that there was no correlation between the two variables with a $\rho = 0.349$, p = 0.069.

If the *statistical significance* level (α) for testing the *null hypothesis* was 10% instead of 5%, we would be able to reject the null hypothesis, and prove that there was correlation between modularity and code maintenance. Even if it was a weak correlation due to the small ρ value.

Hypothesis 4. The academic and industrial professional background of a developer doesn't influence the importance given to the limitations (and its consequences) in the support to modularity in MATLAB.

A Mann-Whitney U test was conducted to determine whether there was a difference in the academic community and in the industrial community about the importance given to the limitations (and its consequences) in the support to modularity in MATLAB. Results of that analysis indicated that even if there was a difference, we couldn't reject the null hypothesis, U = 61.5, p = .155.

Even though we can't reject the hypotheses proposed initial, we can still take some conclusions using the Figure below.



Graph 7.3: Modularity histograms per group

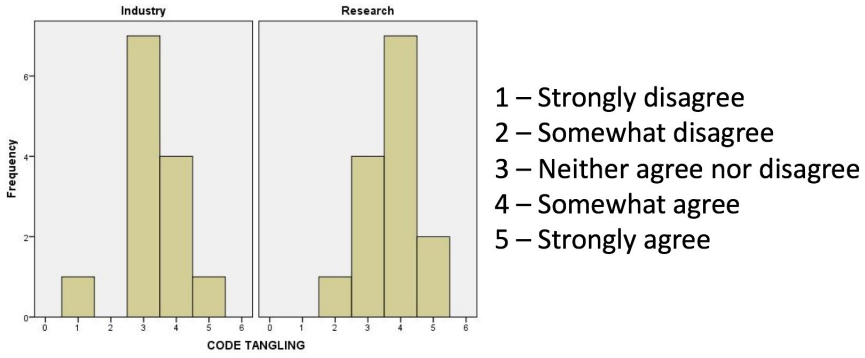
According to this histogram, we can say that MATLAB developers in the industry care more about modularity than the ones that use MATLAB just for research. This may be a side effect of the type of work produce by the developers in each side. Developers

in the industry tend to build long term solutions so the code needs to be prepared to evolve, while in research it can be just proofs of concept.

Hypothesis 5. The academic and industrial professional background of a developer doesn't influence the importance given to code tangling in MATLAB code.

A Mann-Whitney U test was conducted to determine whether there was a difference in the academic community and in the industrial community about the importance given to code tangling in MATLAB code. Results of that analysis indicated that even if there was a difference, we couldn't reject the null hypothesis, U = 68, p = .280.

Even though we can't reject the hypotheses proposed initial, we can still take some conclusions using the Figure below.



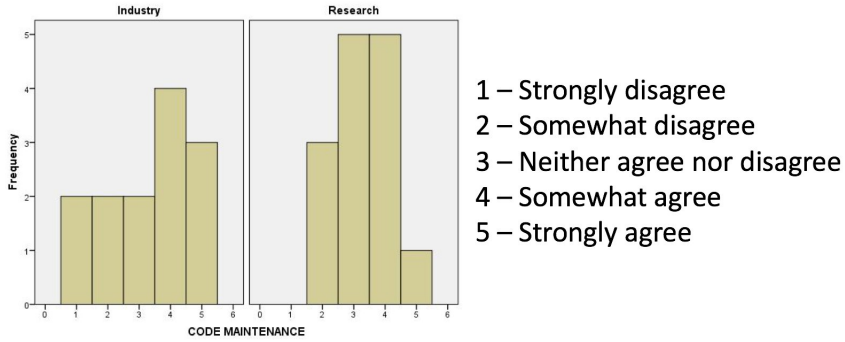
Graph 7.4: Code tangling histograms per group

According to this histogram, we can say that MATLAB developers in the industry care less about code tangling than the ones that use MATLAB just for research. But with the help of the result obtained in *Hypothesis 1*, we can say that both groups care about tangling of concerns in MATLAB code.

Hypothesis 6. The academic and industrial professional background of a developer doesn't influence the importance given to MATLAB code maintenance.

A Mann-Whitney U test was conducted to determine whether there was a difference in the academic community and in the industrial community about the importance given MATLAB code maintenance. Results of that analysis indicated we can't reject the null hypothesis, U = 85.5, p = .793.

Even though we can't reject the hypotheses proposed initial, we can still take some conclusions using the Figure below.



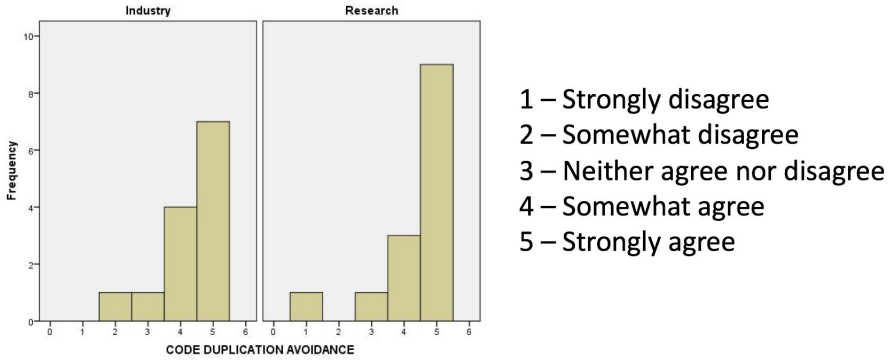
Graph 7.5: Code maintenance histograms per group

According to this histogram, we can say that MATLAB developers in the industry care the same about code maintenance than the ones that use MATLAB just for research.

Hypothesis 7. The academic and industrial professional background of a developer doesn't influence the importance given to code duplication avoidance while developing software in MATLAB.

A Mann-Whitney U test was conducted to determine whether there was a difference in the academic community and in the industrial community about the importance given code duplication avoidance while developing software in MATLAB. Results of that analysis indicated we can't reject the null hypothesis, U = 83, p = .720.

Even though we can't reject the hypotheses proposed initial, we can still take some conclusions using the Figure below.



Graph 7.6: Code duplication avoidance frequencies per group

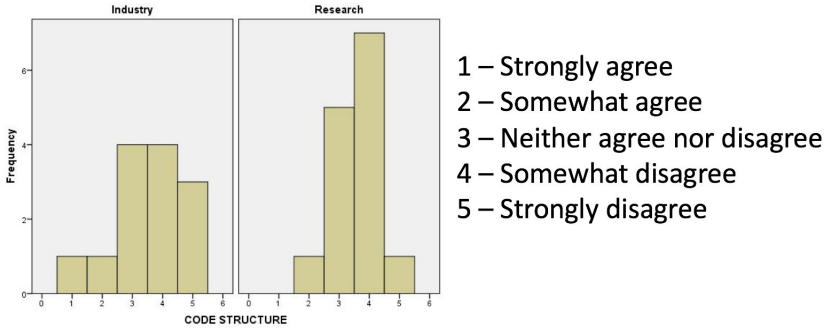
According to this histogram, we can say that MATLAB developers in the industry

care the same about code duplication avoidance than the ones that use MATLAB just for research.

Hypothesis 8. The academic and industrial professional background of a MATLAB developer doesn't influence the importance given to code structure.

A Mann-Whitney U test was conducted to determine whether there was a difference in the academic community and in the industrial community about the importance given code structure. Results of that analysis indicated we can't reject the null hypothesis, U = 89, p = .943.

Even though we can't reject the hypotheses proposed initial, we can still take some conclusions using the Figure below.



Graph 7.7: Code structure histograms per group

According to this histogram, we can say that MATLAB developers that uses this language in either the industry or academic world don't care much about the concept of code structure.

7.3 Contributions

Through the MATLAB modularity study, the execution of the survey-based empirical study with the use of a questionnaire, the statistical analyses and hypothesis testing this research reveals the following contributions:

- First step towards understanding if MATLAB developers acknowledge the modularity limitations and its consequences in MATLAB code/systems.

- The MathWorks tool to visualize the code structure may not be enough for the developers.
- There are some difficulties in trying to explain the concept of code scattering to developers that don't have a software engineering background.
- Finding that MATLAB developers feel and care about code tangling.
- PIMETA instantiation of the MATLAB grammar.
- Comparison of the modularity between MATLAB and JAVA, i.e., between procedural and object-oriented paradigms. The result of this study is that object-oriented languages have higher level of modularity than MATLAB.

7.4 Research Limitations

The results of this empirical study are subject to the following limitations:

- **Low number of responses:** Although the expected sample size was of 1235, we only had 32 completed responses and 10 partial. That number is too small to generalize the results gathered to all the MATLAB developers.
- **Low participant representativeness:** Not every person that belongs to our target population can be found through the chosen communities. So, there are no strong evidence that the response gathered can be translated to the opinion of all the MATLAB developers. The survey research should be repeated with different participants/communities to assert the external validity of the study.
- **Low participant reliability:** Not all of the participants who entered the questionnaire decided to answer it or complete it, thus creating partial responses. In most cases, the respondents give up after reading the explanation or completing the first questions. However, it's probable that, a shorter and even more focused survey, may produce more reliable results.
- **Data pollution:** The last limitation can be closely related with this one. Data pollution occurs when there is ill intention by the participants, and that can include multiple submissions of the questionnaire by the same participant or, lack of commitment to finish the questionnaire. Considering that our questionnaire was anonymous and we had some partial responses, we must consider this a possible limitation to our study. A way to overcome this problem would be allowing

only one response per Internet Protocol (IP) address and not accept the partial responses.

- **Researchers community:** The community of researchers used in the study was created thoughts non-probabilistic methods. That created a high risk for bias in our research and the respective results, disabling us from generalizing our results towards all community of MATLAB developers. A way to solve this problem is to find a population with similar characteristics to one we used, that is well defined and that lends itself to be sampled through some probabilistic method.

- **Limited control over who participate:** Three of the four communities used in this research were from social networks, where we couldn't control who responded to our questionnaire and who didn't. That power is relegated to the participants of the community, making our control over the sample size and the number of responses almost nonexistent. The solution for this limitation is like the one presented in the last point.

- **Saturation of members in communities from social networks:** All of the online communities used had at least 13 792 or more participants, and it was expected a high response/completion rate. However, what occurred was the opposite. That may have happened because of the saturation that communities in social networks suffer nowadays. Most of the participants tend to participate in certain communities to say that they have a certain skill that area. Another, factor that occurs especially in LinkedIn, is HR recruiters that join those communities to filter through possible future employees. These tendencies make almost impossible for all the communities' participants to see a post advertising a survey research, and those who see might to even know much about the topic. A possible solution, that wasn't tested in this research would have been using small and active communities, for a social network.

- **Time constraint:** This problem was particular felt when we sent the emails to the researchers community. Unfortunately, we started that process at the end of June, and many of the researchers contacted were unavailable due to their vacations. To get more responses from these communities we should have chosen a better time frame to send the email inviting them. An example of such time, would have been at the beginning of a new semester.

7.5 Future Work

The validation of our initial problem can still be considered an open-end worth pursuing, regardless the findings that we gathered. That happens because not only our sample is not representative of our target population, and the other limitations that we discovered (see Section 7.4), there are other works that can be done to extend the validation of the research. This section presents some of those directions.

- Validate the PIMETA instantiation of MATLAB. That could be done through more iteration of the instantiation and a more profound study about all the features and dependencies that MATLAB holds. A special case would be comparing the PP instantiation of MATLAB versus the OOP instantiation of MATLAB.

- Replicate our survey-base empirical study, with the same hypothesis and research questions. The questions itself, could be either improved versions of our or completely different questions, they only should be related to our problem. This way, it would be possible with a bigger and different sample population, to test the external validation and reliability of our study/results.

- Find a company that have a R&D or development department, where the employees' use MATLAB as their principal programming language. After doing that research, use that community to do a survey-based empirical study using the same hypotheses or problem proposed in the dissertation. The survey could be applied using questionnaires or interviews, and would serve to get feasible response from the industrial community about our problem.

- Create a small questionnaire and deliver it to the developers whose work nature occurs in the industrial work. That questionnaire focus would be to verify, if these developers use MATLAB to create concept tests or for actual software development. Depending of the type of response gather, it would be possible to create specific target population for a similar study to ours, and verify which community cares more about the modularity and the appearance of symptoms of the limitations in their code.

- Further the state-of-the art in the study of modularity in MATLAB, especially in the procedural paradigm version of the language since it's the one that most developers use. Much of the existing documentation is either related to Java or of the AOP version of MATLAB. The research should be focused on how to find the

occurrences of the symptoms in the code and what are the possible solutions to eliminate or mitigate the existence of said occurrences.

- Create a catalog of refactoring and code smells like the one written by Martin Fowler [1] . The creation of such book or handbook, would start by focusing the research towards a database filled with MATLAB developers, and study the programs until it found common design patterns between them. This would probably be an important tool for MATLAB developers, to help them fight the limitations in the support to modularity. It would also, help them improve their capability to refactor legacy code, and through that the maintainability of the code.

Although is not related to our research validation, in the last years MATLAB (MathWorks and GNU Octave version), started to implement the OO paradigm. A recommendation for future work, would be studying this version of MATLAB (and code produce in it) and verify if that version can eliminate or mitigate some of the symptoms produce by the limitations. In case the answer is positive, see if the developers are starting to be adopted this version of MATLAB, as a possible way to improve the code maintainability and eliminate code scattering and tangling.

[1] Fowler, Martin. *Refactoring: improving the design of existing code*. Pearson Education India, 2009.

Bibliography

[1] *About GNU Octave*. http://www.gnu.org/software/octave/about.html. Accessed: 2015-12-31.

[2] *Array Comparison with Relational Operators - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/array-comparison-with-relational-operators.html. Accessed: 2016-03-08.

[3] *Array vs. Matrix Operations - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/array-vs-matrix-operations.html. Accessed: 2016-03-06.

[4] T. Aslam, J. Doherty, A. Dubrau, and L. Hendren. "AspectMatlab: An aspect-oriented scientific programming language". In: *Proceedings of the 9th International Conference on Aspect-Oriented Software Development*. ACM. 2010, pp. 181–192.

[5] J. M. Cardoso, J. Fernandes, and M. Monteiro. "Adding aspect-oriented features to matlab". In: *workshop on Software Engineering Properties of Languages and Aspect Technologies (SPLAT*. 2006.

[6] J. M. Cardoso, J. Fernandes, and M. Monteiro. "Adding aspect-oriented features to matlab". In: *workshop on Software Engineering Properties of Languages and Aspect Technologies (SPLAT)*. 2006.

[7] J. M. Cardoso, J. M. Fernandes, M. P. Monteiro, T. Carvalho, and R. Nobre. "Enriching MATLAB with aspect-oriented features for developing embedded systems". In: *Journal of Systems Architecture* 59.7 (2013), pp. 412–428.

[8] *Cell Arrays - MATLAB Simulink*. http://www.mathworks.com/help/matlab/cell-arrays.html. Accessed: 2016-05-02.

[9] *Control Flow - MATLAB Simulink*. http://www.mathworks.com/help/matlab/control-flow.html. Accessed: 2016-01-06.

[10] *Create Variables - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/create-variables.html. Accessed: 2016-05-03.

[11] *Declare function name, inputs, and outputs - MATLAB function*. http://www.mathworks.com/help/matlab/ref/function.html. Accessed: 2016-04-06.

[12] *Declare variables as global - MATLAB global*. http://www.mathworks.com/help/matlab/ref/global.html. Accessed: 2016-05-03.

[13] *Differences between Octave and MATLAB - Sysnet Documentation 0.0.1*. https://www.ices.utexas.edu/sysdocs/Octave-Matlab/. Accessed: 2015-11-23.

[14] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian. "Selecting empirical methods for software engineering research". In: *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[15] *FAQ - Octave*. http://wiki.octave.org/FAQ#Toolboxes. Accessed: 2016-04-16.

[16] L. K. Ferrett and J. Offutt. "An empirical comparison of modularity of procedural and object-oriented software". In: *Engineering of Complex Computer Systems, 2002. Proceedings. Eighth IEEE International Conference on*. IEEE. 2002, pp. 173–182.

[17] *GNU Octave: Comparison Ops*. https://www.gnu.org/software/octave/doc/v4.0.1/Comparison-Ops.html#Comparison-Ops. Accessed: 2016-03-10.

[18] *GNU Octave explained*. http://everything.explained.today/GNU_Octave/. Accessed: 2015-12-31.

[19] *GNU Octave: Functions Files*. https://www.gnu.org/software/octave/doc/interpreter/Function-Files.html#Function-Files. Accessed: 2015-12-23.

[20] *GNU Octave: The do-until Statement*. https://www.gnu.org/software/octave/doc/interpreter/The-do_002duntil-Statement.html. Accessed: 2016-01-05.

[21] M. Goulão, B. E. Abreu, et al. "Modeling the experimental software engineering process". In: *Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the*. IEEE. 2007, pp. 77–90.

[22] A. Houston. *Survey Handbook*. 2003.

[23] *How to perform a principal components analysis*. https://statistics.laerd.com/spss-tutorials/principal-components-analysis-pca-using-spss-statistics.php. Accessed: 2017-03-10.

[24] R. A. Kaimann. "Coefficient of network complexity". In: *Management Science* 21.2 (1974), pp. 172–177.

[25] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. "An overview of AspectJ". In: *European Conference on Object-Oriented Programming*. Springer. 2001, pp. 327–354.

[26] B. Kitchenham and S. L. Pfleeger. "Principles of survey research part 4: questionnaire evaluation". In: *ACM SIGSOFT Software Engineering Notes* 27.3 (2002), pp. 20–23.

[27] B. Kitchenham and S. L. Pfleeger. "Principles of survey research: part 5: populations and samples". In: *ACM SIGSOFT Software Engineering Notes* 27.5 (2002), pp. 17–20.

[28] B. A. Kitchenham and S. L. Pfleeger. "Principles of survey research: part 3: constructing a survey instrument". In: *ACM SIGSOFT Software Engineering Notes* 27.2 (2002), pp. 20–24.

[29] E. M. Lakatos and M. d. A. Marconi. "Fundamentos da metodologia científica". In: *Fundamentos da metodologia científica*. Altas, 2010.

[30] *Local Functions - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/local-functions.html. Accessed: 2016-01-27.

[31] C. I. V. Lopes. "D: A language framework for distributed programming". PhD thesis. Northeastern University, 1997.

[32] I. de M Lessa, G. de F Carneiro, M. J. T. Monteiro, B. E. Abreu, et al. "A multiple view interactive environment to support MATLAB and GNU/Octave program comprehension". In: *Information Technology-New Generations (ITNG), 2015 12th International Conference on*. IEEE. 2015, pp. 552–557.

[33] F. S. B. D. Marques. "Modularity Improvements with Aspect-Oriented Programming". PhD thesis. Universidade Nova de Lisboa, 2008.

[34] *MATLAB - The Language of Technical Computing*. http://www.mathworks.com/products/matlab/. Accessed: 2015-12-30.

[35] *MATLAB Programming/Advanced Topics/Object Oriented Programming - Wikibooks, open books for an open world*. https://en.wikibooks.org/wiki/MATLAB_Programming/Advanced_Topics/Object_Oriented_Programming. Accessed: 2015-10-26.

[36] T. J. McCabe. "A complexity measure". In: *IEEE Transactions on software Engineering* 4 (1976), pp. 308–320.

[37] M Monteiro, J. M. Cardoso, and S. Posea. "Identification and characterization of crosscutting concerns in MATLAB systems". In: *Conference on Compilers, Programming Languages, Related Technologies and Applications (CoRTA 2010), Braga, Portugal*. Citeseer. 2010, pp. 9–10.

[38] G. J. Myers et al. *Composite/structured design*. Van Nostrand Reinhold, 1978.

[39] *Octave - General - query on using matlab toolboxes in Octave*. http://octave.1599824.n4.nabble.com/query-on-using-matlab-toolboxes-in-Octave-td4527064.html. Accessed: 2016-04-10.

[40] *Operators and Elementary Operations - MATLAB Simulink*. http://www.mathworks.com/help/matlab/operators-and-elementary-operations.html. Accessed: 2016-01-06.

[41] D. L. Parnas. "On the criteria to be used in decomposing systems into modules". In: *Communications of the ACM* 15.12 (1972), pp. 1053–1058.

[42] S. L. Pfleeger. "Design and analysis in software engineering: the language of case studies and formal experiments". In: *ACM SIGSOFT Software Engineering Notes* 19.4 (1994), pp. 16–20.

[43] S. L. Pfleeger and B. A. Kitchenham. "Principles of survey research part 2: designing a survey". In: *Software Engineering Notes* 27.1 (2002), pp. 18–20.

[44] K. Popper. *The logic of scientific discovery*. Routledge, 2005.

[45] *Private Functions - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/private-functions.html. Accessed: 2016-01-27.

[46] R. T. Renckly. *Air university sampling and surveying handbook*. 1996.

[47] E. M. Rogers. "Physics for the inquiring mind". In: (1960).

[48] *Scripts vs. Functions - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/scripts-and-functions.html. Accessed: 2015-12-23.

[49] *Scripts vs. Functions - MATLAB Simulink*. http://www.mathworks.com/help/matlab/matlab_prog/scripts-and-functions.html. Accessed: 2016-04-23.

[50] N. Sharma and M. K. Gobbert. "A comparative evaluation of Matlab, Octave, FreeMat, and Scilab for research and teaching". In: *Department of Mathematics and Statistics* (2010).

[51] *Toolbox Distribution - MATLAB Simulink*. http://www.mathworks.com/help/matlab/creating-help.html. Accessed: 2016-04-16.

[52] L. Vasconcelos and L. F. A. Guedes. "E-surveys: vantagens e limitações dos questionários eletrônicos via internet no contexto da pesquisa científica". In: *X SEMEAS, FEA-USP* (2007).

[53] S. Vegas. "What Makes a Good Empirical Software Engineering Thesis?: Some Advice". In: (2015).

[54] J. Visser, S. Rigal, R. van der Leek, P. van Eck, and G. Wijnholds. *Building Maintainable Software, Java Edition: Ten Guidelines for Future-Proof Code*. O'Reilly Media, Inc., 2016.

[55] J. Vlissides, R. Helm, R. Johnson, and E. Gamma. "Design patterns: Elements of reusable object-oriented software". In: *Reading: Addison-Wesley* 49.120 (1995), p. 11.

[56] *What Makes a Good Research Question? - Duke*. `http://twp.duke.edu/uploads/media_items/research-questions.original.pdf`. Accessed: 2016-03-24.

[57] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. 2012.

[58] R. K. Yin. *Case study research: Design and methods*. Sage publications, 2013.

Appendix A

MATLAB Feature and Dependency Types

This appendix presents the MATLAB Feature Types, their mutual aggregation possibilities and their DependencyTypes with which PIMETA is instantiated.

Modular Feature Types	Atomic Feature Types
Toolbox	GlobalVariable
ScriptFile	LocalVariable
FunctionFile	InputValue
LocalFunction	OutputValue
PrivateFunction	

Table A.1: MATLAB Feature Types

Aggregations
(Toolbox, Toolbox)
(Toolbox, ScriptFile)
(Toolbox, FunctionFile)
(ScriptFile, ScriptFile)
(ScriptFile, FunctionFile)
(ScriptFile, GlobalVariable)
(ScriptFile, LocalVariable)
(FunctionFile, GlobalVariable)
(FunctionFile, LocalVariable)
(FunctionFile, PrivateFunction)
(FunctionFile, LocalFunction)
(LocalFunction, LocalVariable)
(LocalFunction, LocalFunction)
(LocalFunction, PrivateFunction)
(LocalFunction, InputValue)
(LocalFunction, OutputValue)
(PrivateFunction, LocalVariable)
(PrivateFunction, PrivateFunction)
(PrivateFunction, InputValue)
(PrivateFunction, OutputValue)

Table A.2: Features Aggregations

DependencyTypes
DeclareGlobalVariable: (GlobalVariable, LocalFunction)
DeclareLocalFunction: (FunctionFile, LocalFunction)
DeclarePrivateFunction: (FunctionFile, PrivateFunction)
ScriptFileImportsFunctionFile: (ScriptFile, FunctionFile)
ScriptFileImportsLocalFunction: (ScriptFile, LocalFunction)
FunctionFileImportsPrivateFunction: (FunctionFile, PrivateFunction)
FunctionFileImportsToolbox: (FunctionFile, Toolbox)
ScriptFileImportsToolbox: (ScriptFile, Toolbox)
LocalFunctionReturnsOutputValue: (LocalFunction, OutputValue)
PrivateFunctionReturnsOutputValue: (PrivateFunction, OutputValue)

Table A.3: Dependencies Types

Appendix B

Formulas

This appendix presents all the mathematical formulas needed to calculate a sample size and rates.

Confidence Level	Z Factor
99.9	3.2905
99.7	3
99.5	2.807
99	2.5758
98	2.3263
95.5	2
95	1.96
90	1.6449
85	1.4395
80	1.2816

Table B.1: Z Values

Type of Results	Formulas	Subtitles
Percentage	$n = \frac{P(1-P)}{\frac{A^2}{Z^2} + \frac{P(1-P)}{N}}$	n = sample size required; N = number of people in the population; P = estimated percentage of the population possessing attribute of interest; A = accuracy desired, expressed as a decimal; Z = number of standard deviation units of the sampling distribution corresponding to the desired confidence level.
Average	$n = \frac{P^2}{\frac{A^2}{Z^2} + \frac{P^2}{N}}$	n = sample size required; N = number of people in the population; P = estimated standard deviation of the attribute of interest in the population; A = accuracy desired, expressed as a decimal; Z = number of standard deviation units of the sampling distribution corresponding to the desired confidence level.
Multiple Ways	$n = \frac{NZ^2 * 0.25}{(d^2 * [N-1]) + (Z^2 * 0.25)}$	n = sample size required; N = total population size [known or estimated]; d = precision level [usually .05 or .1] Z = number of standard deviation units of the sampling distribution corresponding to the desired confidence level.

Table B.2: Determining the size of the sample

	Formulas
Completion Rate (%)	(Complete number of surveys / Number of respondents who entered the survey) * 100
Response Rate (%)	(Complete number of surveys / Number of emails sent) * 100

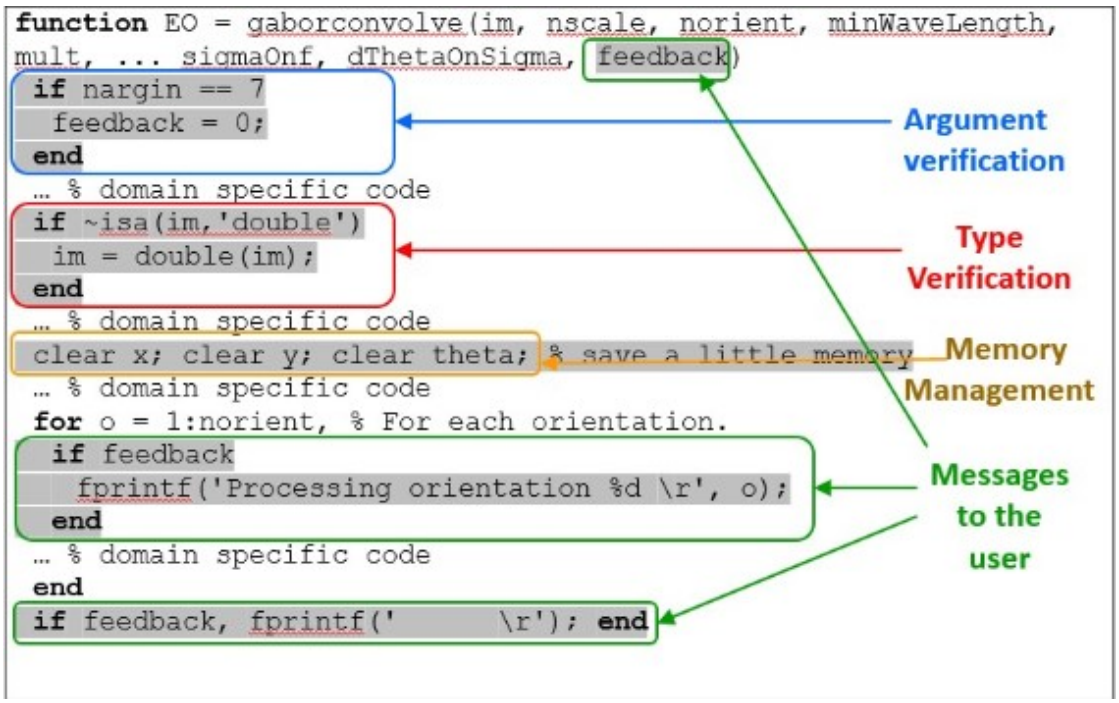
Table B.3: Rates Formulas

Appendix C

Modularity in MATLAB - Pilot Test

The purpose of the survey is to collect feedback from MATLAB programmers regarding the symptoms of code tangling (as illustrated in the figure below). These symptoms may hamper understandability and ease of maintenance.

The code example illustrates how different sections of code can be related to different "concerns" or kinds of functionality, which programmers would ideally place into separate functions or m-files. Note that languages other than MATLAB may provide better support for modular organization.



Your participation in this research study is voluntary. You may choose not to participate. If you decide to participate in this research survey, you may withdraw at any time. If you decide not to participate in this study or if you withdraw from participating at any time, you will not be penalized.

We will do our best to keep your information confidential. All data is stored in a password protected electronic format. To help protect your confidentiality, the surveys will not contain information that will personally identify you. The results of this study will be used for scholarly purposes only and may be shared in papers of the specialty.

The procedure involves filling an on-line survey that will take approximately 10 minutes. Your responses will be confidential and we do not collect identifying information such as your name, email address or IP address.

Do you agree to the above terms? By clicking Yes, you consent that you are willing to answer the questions in this survey.

◯ Yes

◯ No

Tangling/Scattering

1. I often come across tangling of concerns in the same function or m-file, such as described in the example provided.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

2. It is easier to understand code where multiple concerns (such as described in the example provided) are not tangled with each other in the same module (function or m-file).

   ◯ Strongly agree

   ◯ Somewhat agree

   ◯ Neither agree nor disagree

   ◯ Somewhat disagree

   ◯ Strongly disagree

3. Code tangling in MATLAB may reduce understandability in the long term for the original programmer or by other programmers.

   ◯ Strongly agree

   ◯ Somewhat agree

   ◯ Neither agree nor disagree

   ◯ Somewhat disagree

   ◯ Strongly disagree

4. When working in a MATLAB system, I try to avoid duplicated code whenever possible.

   ◯ Strongly agree

   ◯ Somewhat agree

   ◯ Neither agree nor disagree

   ◯ Somewhat disagree

   ◯ Strongly disagree

5. I often think in terms of modularity when programming in MATLAB.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
6. When I work on a MATLAB program, I normally find duplicated code across the various m-files.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
7. When I program in MATLAB, I try to divide my code in small modules (functions or m-files) as a strategy to mitigate complexity.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
8. I consider symptoms of code tangling to be something normal.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
9. My problems of code tangling are caused by lack of modularity mechanisms in MATLAB.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

10. Duplicated code is not an issue for me when I program in MATLAB.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

MATLAB Legacy Code

11. The first thing I do, when I start maintaining a MATLAB system is to visualize and analyze the source code.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

12. I do not expect anybody besides me to run or improve my MATLAB programs.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

13. While maintaining a MATLAB program, I did not experience any difficulties in understanding the code or its structure.

◯ Strongly agree

◯ Somewhat agree
◯ Neither agree nor disagree
◯ Somewhat disagree
◯ Strongly disagree

14. In the past, I had to maintain a MATLAB program for a long period of time.
◯ Strongly agree
◯ Somewhat agree
◯ Neither agree nor disagree
◯ Somewhat disagree
◯ Strongly disagree

15. Sometimes I feel the need to have a tool that helps visualize the structure of my MATLAB code.
◯ Strongly agree
◯ Somewhat agree
◯ Neither agree nor disagree
◯ Somewhat disagree
◯ Strongly disagree

16. Good modularity brings benefits to the maintainability of a MATLAB program.
◯ Strongly agree
◯ Somewhat agree
◯ Neither agree nor disagree
◯ Somewhat disagree
◯ Strongly disagree

17. My MATLAB programs are mainly developed for solving short-term problems.
◯ Strongly agree
◯ Somewhat agree
◯ Neither agree nor disagree
◯ Somewhat disagree
◯ Strongly disagree

18. When I develop a MATLAB program, I make a strong effort to make it reusable.
    - ◯ Strongly agree
    - ◯ Somewhat agree
    - ◯ Neither agree nor disagree
    - ◯ Somewhat disagree
    - ◯ Strongly disagree

19. The dependency report tool offered by MathWorks is enough to visualize the structure of my code.
    - ◯ Strongly agree
    - ◯ Somewhat agree
    - ◯ Neither agree nor disagree
    - ◯ Somewhat disagree
    - ◯ Strongly disagree

Background Information

20. How many years of experience do you have programming in MATLAB?
    - ◯ Less than 1 year
    - ◯ 1 to 4 years
    - ◯ 5 to 9 years
    - ◯ More than 10 years

21. Last time I programmed in MATLAB program was ...
    - ◯ less than 1 year ago.
    - ◯ between 2 to 3 years ago.
    - ◯ over 4 years ago.
    - ◯ Other: ☐

22. I have experience in dealing with large MATLAB systems.
    - ◯ Yes
    - ◯ No

23. What do you use MATLAB for?

- ☐ Modeling
- ☐ Prototyping of hardware platforms
- ☐ Calculator
- ☐ Scientific computation
- ☐ Others: [ ]

24. How would you classify the nature of your work when using MATLAB:

- ◯ Industry
- ◯ Research
- ◯ Student

25. In what industry do you carry out your work?

- ☐ Energy Production
- ☐ Financial Services
- ☐ Automotive
- ☐ Industrial Automation and Machinery
- ☐ Communication Infrastructure
- ☐ Software Library Development
- ☐ Computer Electronics
- ☐ Other: [ ]

26. What is your area of study?

- ☐ Energy Production
- ☐ Financial Services
- ☐ Automotive
- ☐ Industrial Automation and Machinery
- ☐ Communication Infrastructure
- ☐ Software Library Development
- ☐ Computer Electronics
- ☐ Other: [ ]

27. When I use MATLAB I work in the following application domain:
    - ☐ Signal Processing
    - ☐ Image Processing
    - ☐ Data Analysis
    - ☐ Production Analytics
    - ☐ Machine Learning
    - ☐ Simulation
    - ☐ Other: [ ]

28. What is your area of study?
    - ☐ Energy Production
    - ☐ Financial Services
    - ☐ Automotive
    - ☐ Industrial Automation and Machinery
    - ☐ Communication Infrastructure
    - ☐ Software Library Development
    - ☐ Computer Electronics
    - ☐ Other: [ ]

29. When I use MATLAB I work in the following application domain:
    - ☐ Signal Processing
    - ☐ Image Processing
    - ☐ Data Analysis
    - ☐ Production Analytics
    - ☐ Machine Learning
    - ☐ Simulation
    - ☐ Others: [ ]

30. I normally work with a team of developers rather than alone, when I am developing a MATLAB system.
    - ◯ Yes
    - ◯ No

31. How many developers (besides yourself) make up the developing team:

    [ ]

32. Are you familiar with other programming languages?

    ◯ Yes

    ◯ No

33. Which languages are you familiar with?

    ☐ C/C++

    ☐ GNU Octave

    ☐ Python

    ☐ Java

    ☐ Others: [ ]

Appendix D

Feedback from Pilot Test

Professor Miguel Monteiro

Date: April 4th
Time: 00:09:55
Feedback:

- Lack of illustrative figure in the first page with questions.
- Question: *How would you classify the nature of your work when using MATLAB"*. Missing option "Teaching" – not the same as "Student".

Professor Manuel Ortigueira

Date: April 5th
Time: 00:05:37
Feedback:

- Tangling is a concept a bit hard to understand and the examples for "concerns", at the begin, are not the ones I normal see/use.
- I hoped that the questions were about the concerns illustrated in the image at the beginning of the questionnaire.

Professor Paulo Gil & Professor Luís Palma

Date: April 6th

Time: 00:10:48 & 00:08:00
Feedback:

- The concepts of concern and tangling are a bit confusing.
- Question: *What is your area of study?*. It would help that in the option "Other:" to write more than one option.

Professor Arnaldo Batista
Date: April 13th
Time: 00:06:32
Feedback:

- The concept of tangling should be explained in a simpler way.

Professor Francisco Monteiro
Date: April 14th
Time: 00:18:07
Feedback:

- It's not clear what you mean with large program.
- Question: *Which languages are you familiar with?*. It's pointless to indicate GNU Octave as an option since they have the same syntax as MATLAB.

Professor Glauco Carneiro
Date: April 24th
Time: 00:00:00
Feedback:

- It's not clear what you mean with large program.
- The navigation should only let the person go forward. This way, it may help to validate the answers (internally) since we can't go back and change our answers.
- Add question: *Beyond the "concerns" highlighted in the previous figure, can you identify other concerns?*.
- Add question: *How often do you see examples of tangling in your code?*

- Don't talk about "concerns" per say, the industrial people won't understand. Instead use the words concepts and/or functionalities.
- Question: *For which purpose(s) do you use MATLAB?*. Change response format to open-end.
- Question: *How would you classify the nature of your work when using MATLAB:*. Change response format to selecting multiple options and add option "Other".
- Question: *Experience with "large" MATLAB programs*. It's too subjective! Create an ordinal scale.

Professor João Cardoso
Date: April 24th
Time: 00:17:10
Feedback:

- Add question: *I normally deal with MATLAB programs with....*
- Add question: *I normally deal with MATLAB m-files with....*
- Add question: *How many toolboxes you tend to use.*
- Add question: *Which toolboxes do you use the most?*.
- Add question: *Which workspace do you use when programming MATLAB?*.

APPENDIX E

Questions created for the survey

Variable Name	Research Question	Type	Question	Response Format
VAR01	2.1	Numerical	I usually find tangling of concerns in the same function or m-file, such as described in the example provided.	Likert Scale
VAR02	2.1	Numerical	It is easier to understand code where multiple concerns (such as described in the example provided) are not tangled with each other in the same module (function or m-file).	Likert Scale
VAR03	2.1	Numerical	Code tangling in MATLAB may reduce understandability in the long term for the original programmer or by other programmers.	Likert Scale
VAR04	3.1	Numerical	Beyond the "concerns" highlighted in the previous figure, can you identify other concerns?	Selecting Option
VAR05	3.1	Short Text	If yes, please name the ones you can think of.	Open-ended
VAR06	2.2	Numerical	When working in a MATLAB program, I try to avoid duplicated code whenever possible.	Likert Scale
VAR07	2.1	Numerical	How often do you see examples of tangling in your code?	Likert Scale
VAR08	3.2	Numerical	I often think in terms of modularity when programming in MATLAB.	Likert Scale
VAR09	2.2	Numerical	When I work on a MATLAB program, I normally find duplicated code across the various m-files.	Likert Scale
VAR10	3.2	Numerical	When I program in MATLAB, I try to divide my code in small modules (functions or m-files) as a strategy to mitigate complexity.	Likert Scale
VAR11	2.1	Numerical	I consider symptoms of code tangling to be something normal.	Likert Scale
VAR12	2.1	Numerical	My problems of code tangling are caused by lack of modularity mechanisms in MATLAB.	Likert Scale
VAR13	2.2	Numerical	When I program in MATLAB, I do not make an effort to eliminate duplicated code.	Likert Scale

Table E.1: Questions related to the identification the subject opinion on code tangling and scattering

Variable Name	Research Question	Type	Question	Response Format
VAR14	4	Numerical	The first thing I do, when I start maintaining a MATLAB program, is visualize and analyze the source code using a tool such as Mathworks' Dependency Report.	Likert Scale
VAR15	1.1	Numerical	I do not expect anybody besides me to run or improve my MATLAB programs.	Likert Scale
VAR16	1.3	Numerical	While maintaining a MATLAB program, I did not experience any difficulties in understanding the code or its structure.	Likert Scale
VAR17	1.1	Numerical	In the past, I had to maintain a MATLAB program for a long period of time.	Likert Scale
VAR18	4	Numerical	Sometimes I feel the need to have a tool that helps visualize the structure of my MATLAB code.	Likert Scale
VAR19	1.2	Numerical	Good modularity brings benefits to the maintainability of a MATLAB program.	Likert Scale
VAR20	1.1	Numerical	My MATLAB programs are mainly developed for solving short-term problems.	Likert Scale
VAR21	1.2	Numerical	When I develop a MATLAB program, I make a strong effort to make it reusable.	Likert Scale
VAR22	4	Numerical	The dependency report tool offered by MathWorks is enough to visualize the structure of my code.	Likert Scale

Table E.2: Questions related to the identification the subject habits and opinions about MATLAB legacy code

Variable Name	Research Question	Type	Question	Response Format
VAR23	3.2	Numerical	How many years of experience do you have programming in MATLAB?	Selecting Option
VAR24A	3.2	Numerical	Last time I programmed in MATLAB program was ...	Selecting Option
VAR24B	3.2	Short Text	Last time I programmed in MATLAB program was ... (Other)	Open-ended
VAR25	3.2	Numerical	I normally deal with MATLAB programs with...	Selecting Multiple Options
VAR26	3.1	Short Text	For which purpose(s) do you use MATLAB?	Open-ended
VAR27	3.2	Numerical	I normally work with a team of developers rather than alone, when I am developing a MATLAB system.	Selecting Option
VAR28	3.2	Short Text	How many developers (besides yourself) make up the developing team:	Open-ended
VAR29A	3.1	Numerical	How would you classify the nature of your work when using MATLAB:	Selecting Multiple Options
VAR29B	3.1	Short Text	How would you classify the nature of your work when using MATLAB: (Other)	Open-ended
VAR30	3.2	Numerical	I normally deal with MATLAB m-files with...	Selecting Option
VAR31	3.1	Numerical	How many toolboxes you tend to use?	Selecting Option
VAR32	3.1	Short Text	Which toolboxes do you use the most?	Open-ended
VAR33	3.1	Numerical	Using the United Nations - International Standard Industrial Classification, where do you care out your work?	Selecting Option
VAR34A	3.1	Numerical	How would you classify the nature of your work when using MATLAB:	Selecting Multiple Options
VAR34B	3.1	Short Text	How would you classify the nature of your work when using MATLAB: (Other)	Open-ended
VAR35	3.2	Numerical	Which workspace do you use when programming MATLAB?	Selecting Multiple Options
VAR36A	3.2	Numerical	Are you familiar with other programming languages?	Selecting Option
VAR36B	3.2	Numerical	Which languages are you familiar with?	Selecting Multiple Options
VAR36C	3.2	Short Text	Which languages are you familiar with? (Others)	Open-ended

Table E.3: Questions related to the identification the subject background and habits when it comes to programming in MATLAB

Appendix F

Invitation Texts

F.1 Post

Modularity in MATLAB - Research Survey

Hi! I'm doing a survey as a part of my MSc dissertation. I would be grateful if you could help me by answering it. It takes around 15 minutes to complete. This is the link to it: https://iscteiul.co1.qualtrics.com/SE/?SID=SV$_e$L1aYvlilL77IxL

Thank you for your attention.

F.2 Email

Dear participant,

I am a MSc student at the Faculty of Sciences and Technology of the Universidade Nova de Lisboa in Portugal and, as part of the preparation of my dissertation, I am conducting a survey on the subject of modularization in MATLAB code and its consequences on the programmer's work.

Through my research, I found a paper with your name on it as one of the authors. Which made us think that you would be interested in participating in this study. For that, I need your opinion and kindly ask for your support and involvement by answering the following questionnaire: https://iscteiul.co1.qualtrics.com/SE/?SID=SV$_5$dyExSnuTQrVEjj

Participation is voluntary and anonymous. Although there is no fixed time for completing this questionnaire, we estimate that, overall, it should be around 10 minutes. All data conducted in this survey will only be evaluated by me and my supervisors - Professor Fernando Brito e Abreu (ISCTE-IUL) and Professor Miguel Monteiro (FCT-UNL) - and included in an aggregated form in my MSc dissertation.

In the end of this questionnaire, you will be able to state if you want us to send you the preliminary results of this study.

If you have further questions, please feel free to contact me at: k.duarte@campus.fct.unl.pt

Thanks for your support,
Katia Duarte

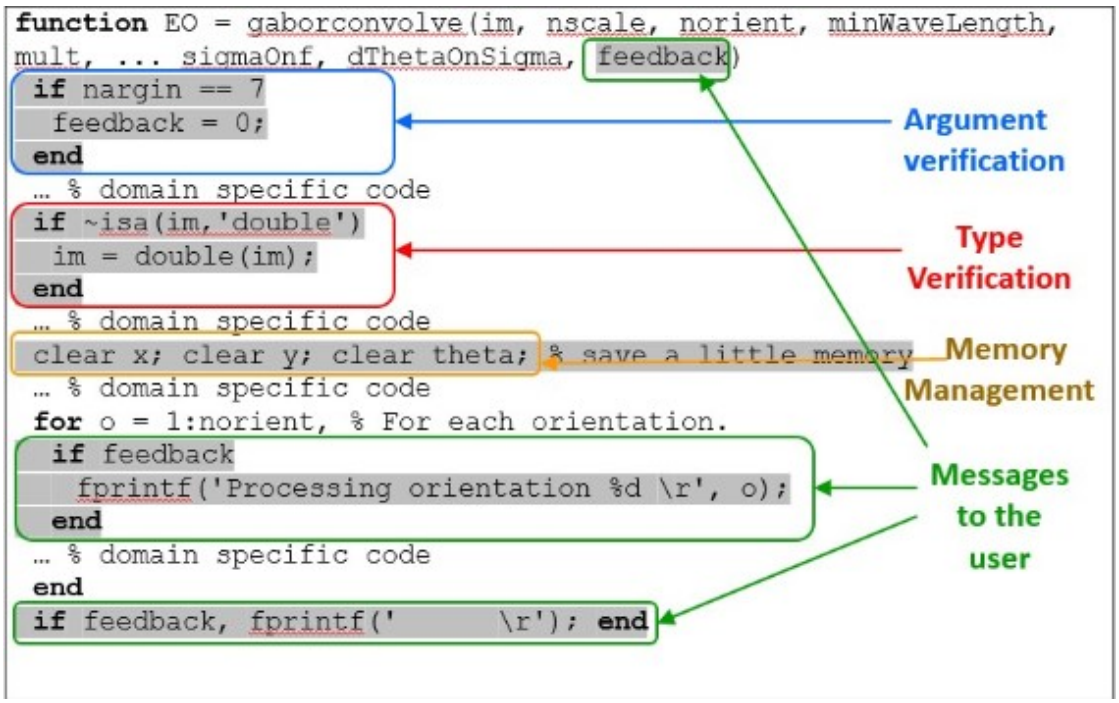
Appendix G

Modularity in MATLAB

The purpose of the survey is to collect feedback from MATLAB programmers regarding the adoption of modularization practices.

The code example illustrates how different sections of code can be related to different “concerns”, which programmers would ideally place into separate modules - in MATLAB, functions or m-files. Here, a "concern" is synonym of "abstraction", "concept", or "additional functionality".

Ideally, each MATLAB function or m-file would contain code related to just one concern. When more concerns are involved, we observe a "tangling of concerns" such as illustrated here: code related to the primary concern appears intertwined with code pertaining to other concerns, which may hamper understandability. Ideally, additional concerns would be modules we would be able to plug and unplug. In the example, note how the ’feedback’ argument is used to switch on and off the "Messages to the user" concern.



Many other concerns may be found in MATLAB systems. For example, checking of arguments to determine the "mode" under which the function will run, checking whether a variable is of a certain type or shape; parallelization; specialized instructions to force variables to keep using some specific type or shape, etc. Other languages may provide features that prevent some of these symptoms to emerge.

Your participation in this research study is voluntary. You may choose not to participate. If you decide to participate in this research survey, you may withdraw at any time.

We will do our best to keep your information confidential. All data is stored in a password protected electronic format. To help protect your confidentiality, the surveys will not contain information that will personally identify you. The results of this study will be used for scholarly purposes only and may be shared in papers of the specialty.

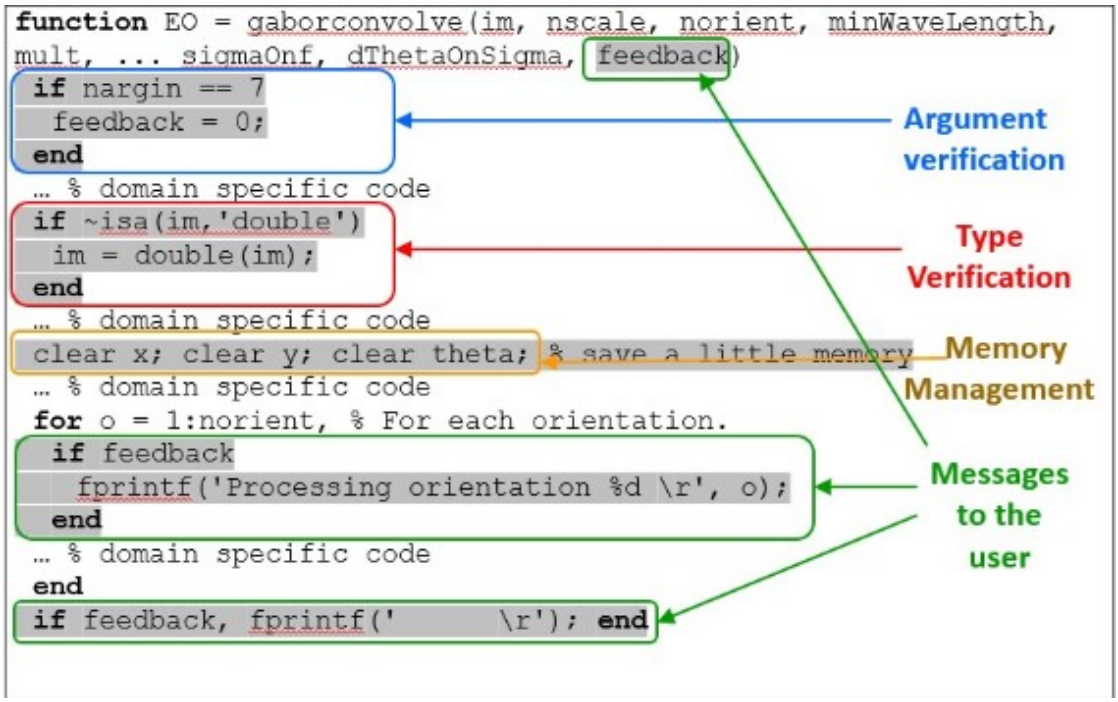
The procedure involves filling an on-line survey that will take approximately 10 minutes. Your responses will be confidential and we do not collect identifying information such as your name, email address or IP address.

Do you agree to the above terms? By clicking Yes, you consent that you are willing to answer the questions in this survey.

◯ Yes

◯ No

Tangling/Scattering



1. I usually find tangling of concerns in the same function or m-file, such as described in the example provided.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
2. It is easier to understand code where multiple concerns (such as described in the example provided) are not tangled with each other in the same module (function or m-file).
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree

3. Code tangling in MATLAB may reduce understandability in the long term for the original programmer or by other programmers.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
4. Beyond the "concerns" highlighted in the previous figure, can you identify other concerns?
   - ◯ Yes
   - ◯ No
5. If yes, please name the ones you can think of.

6. When working in a MATLAB program, I try to avoid duplicated code whenever possible.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
7. How often do you see examples of tangling in your code?
   - ◯ Always
   - ◯ Most of the time
   - ◯ About half the time
   - ◯ Sometimes
   - ◯ Never

8. I often think in terms of modularity when programming in MATLAB.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
9. When I work on a MATLAB program, I normally find duplicated code across the various m-files.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
10. When I program in MATLAB, I try to divide my code in small modules (functions or m-files) as a strategy to mitigate complexity.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
11. I consider symptoms of code tangling to be something normal.
   - ◯ Strongly agree
   - ◯ Somewhat agree
   - ◯ Neither agree nor disagree
   - ◯ Somewhat disagree
   - ◯ Strongly disagree
12. My problems of code tangling are caused by lack of modularity mechanisms in MATLAB.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

13. When I program in MATLAB, I do not make an effort to eliminate duplicated code.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

MATLAB Legacy Code

14. The first thing I do, when I start maintaining a MATLAB program, is visualize and analyze the source code using a tool such as MathWorks' Dependency Report.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

15. I do not expect anybody besides me to run or improve my MATLAB programs.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

16. While maintaining a MATLAB program, I did not experience any difficulties in understanding the code or its structure.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

17. In the past, I had to maintain a MATLAB program for a long period of time.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

18. Sometimes I feel the need to have a tool that helps visualize the structure of my MATLAB code.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

19. Good modularity brings benefits to the maintainability of a MATLAB program.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

20. My MATLAB programs are mainly developed for solving short-term problems.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

21. When I develop a MATLAB program, I make a strong effort to make it reusable.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

22. The dependency report tool offered by MathWorks is enough to visualize the structure of my code.

◯ Strongly agree

◯ Somewhat agree

◯ Neither agree nor disagree

◯ Somewhat disagree

◯ Strongly disagree

Background Information

23. How many years of experience do you have programming in MATLAB?

◯ Less than 1 year

◯ 1 to 4 years

◯ 5 to 9 years

◯ More than 10 years

24. Last time I programmed in MATLAB program was ...

◯ less than 1 year ago.

◯ between 2 to 3 years ago.

◯ over 4 years ago.

◯ Other: [ ]

25. I normally deal with MATLAB programs with...

- ☐ 1 to 5 m-files.
- ☐ 6 to 10 m-files.
- ☐ 11 to 15 m-files.
- ☐ 16 to 20 m-files.
- ☐ 21 to 50 m-files.
- ☐ more than 51 m-files.

26. For which purpose(s) do you use MATLAB?

27. I normally work with a team of developers rather than alone, when I am developing a MATLAB system.

- ◯ Yes
- ◯ No

28. How many developers (besides yourself) make up the developing team:

29. How would you classify the nature of your work when using MATLAB:

- ☐ Industry
- ☐ Research
- ☐ Teaching
- ☐ Other:

30. I normally deal with MATLAB m-files with...

- ◯ 1 function.
- ◯ 2 to 5 functions.
- ◯ 5 to 10 functions.
- ◯ more than 11 functions.

31. How many toolboxes you tend to use?
    - ◯ 1 to 5
    - ◯ 6 to 10
    - ◯ 11 to 20
    - ◯ more than 21
32. Which toolboxes do you use the most?

33. Using the United Nations - International Standard Industrial Classification, where do you care out your work?
    - ◯ Agriculture, forestry and fishing
    - ◯ Mining and quarrying
    - ◯ Manufacturing
    - ◯ Electricity, gas, steam and air conditioning supply
    - ◯ Water supply; sewerage, waste management and remediation
    - ◯ Construction
    - ◯ Wholesale and retail trade; repair of motor vehicles and motorcycles
    - ◯ Transportation and storage
    - ◯ Accommodation and food service activities
    - ◯ Information and communication
    - ◯ Financial and insurance activities
    - ◯ Real estate activities
    - ◯ Professional, scientific and technical activities
    - ◯ Administrative and support service activities
    - ◯ Public administration and defence; compulsory social security
    - ◯ Education
    - ◯ Human health and social work activities
    - ◯ Arts, entertainment and recreation

◯ Other service activities

◯ Activities of households as employers; undifferentiated goods- and services-producing activities of households for own use

◯ Activities of extraterritorial organizations and bodies

34. I use MATLAB to perform this kind of work:

☐ Signal Processing

☐ Image Processing

☐ Data Analysis

☐ Production Analytics

☐ Machine Learning

☐ Simulation

☐ Other: _____

35. Which workspace do you use when programming MATLAB?

☐ MathWorks

☐ GNU Octave

☐ Scilab

36. Are you familiar with other programming languages?

◯ Yes

◯ No

37. Which languages are you familiar with?

☐ C/C++

☐ Python

☐ Java

☐ Others: _____

If you wish to get the preliminary results of this study, just leave your email or LinkedIn URL in this box.

_____