**Portada**

Hello, my name is Núria Bruch and I am going to present the proposal of my thesis

**Outline**

These are the main parts of my presentation and I am going to start by introducing the area in which this project is framed.

**Introduction**

Introducing the area in which this project is framed.

**Dependency management (1)**

This area is dependency management.

So, many different aspects of dependency management have been studied during the past years.

Such as, how to detect a vulnerability in a library and how does it spread across the software ecosystem to which the library belongs.

Also, the trend of adoption of different versions of the same library as well as library recommendations have been researched too.

However, to study the dependencies in an open-source software ecosystem, it is necessary to model it.

**Dependency management (2)**

And how do we do that, currently? Well, most of the representations of the software ecosystems that I have been able to find, do it by the means of graphs. Such as…

**Dependency management (3)**

This one. As you can see here, the nodes of the graphs are the projects of the ecosystem, or the versions of such projects, and the edges indicate that one project uses another one. Such as in here.

This is what is called a package-based dependency graph. And it is used by most of the package managers to solve if there is a dependency between two projects or not.

**Research Question 1**

Now let’s talk about which are the questions that arise in this current situation.

**RQ1 No question**

Going back to the package-based dependency graph, you can see here that the dependencies between projects, are binary: either the dependency exists or doesn’t.

However, how could we know how much we depend on this library. Does it really exist a transitive dependency between this App and Lib 2? Or the parts of Lib 1 that use Lib 2 are not used by App?

All these questions lead us to our first research question.

**RQ1 Question**

How can we measure the degree of source-code library dependency between a project and the libraries?

To answer this question, we asked another question

**RQ1 Coupling metrics**

Which metrics do exist right now, that measure dependency between two elements?

And the answer is Coupling Metrics. These metrics are used to measure the coupling between classes or modules of the same project.

Therefore, the idea is to adapt a sub-set of the existing metrics to measure the coupling between projects.

**Research Question 2**

Let’s talk about the second research question. And let’s go back to our initial dependency graph.

**RQ2 No question**

Now imagine, that for some reason, Lib 1 is either going to become closed-source or has a known vulnerability that we do not want to affect the project. How can we measure, …

**RQ2 Question**

How much effort is it going to be needed to replace the library.

To answer this question, we need to first know, how do we measure effort.

**RQ2 COCHCOMO**

One of the most common answers to this question is COCOMO, which is a model to estimate the effort needed to develop new pieces of code. However, we don’t really want to develop new code, we want to change existing code.

By researching a bit more, I found another model, based on COCOMO, to estimate the effort needed to make changes in existing code.

This model is based on changes to requirements and which part of the code do these requirements affect.

The idea here is to adapt this model to use it with which part of the code is using the library that we want to replace and calculate the effort based on that.

**Expected Results**

So, which is the expected output of this project.

**Set of metrics (1)**

First, the goal is to come up with a set of metrics that can be applied to calculate the dependency or coupling between a project and the libraries. As well as a model to measure the effort needed to replace a library.

For each one of these metrics, we are going to provide…

**Set of metrics (2)**

The formal definition of the metrics as well as a validation criteria that the metric must fulfil.

**Proof-of-concept**

The second result of this project is going to be a PoC. A prototype to calculate the metrics to provide results that can be evaluated.

It is important to consider, that for these PoC, the package-based dependency graph is not enough to calculate the metrics. Therefore, we will need to use a call-level dependency graph.

We have found two papers that provide an approach to create such graph. Both could be used for the purpose of this PoC.

**Evaluation and validation**

Finally, with the definition of the metrics and the PoC, an evaluation and validation will be conducted.

There is no clear and unique methodology to validate metrics. Therefore, different approaches will be taken, in order to validate them as much as possible, considering the time limitation.

According to literature, metric validation has two different approaches, theoretical and empirical.

For the theoretical validation, for example, we will check that the coupling metrics fulfil the properties of coupling metrics defined in this paper.

Also, a sub-set of the validation criteria described in this paper, which is a literature review of the different approaches to validate metrics, will also be used.

Finally, we will conduct some controlled experiments and case studies as the empirical validation of the metrics.

**End**

This has been the presentation of my proposal, now if there are any questions, let me know.

**Research Method**

The main research method used in this project is going to be he Technical Action Research.

As I have already introduced earlier, controlled experiments and case studies will also be conducted to validate the proposed metrics.

**Choosing applicable metrics**

These are some of the questions that I have asked to decide whether it was interesting to use a metric or if it was even applicable to the case.