# TFG-Localization Based Systems and Intelligent Spaces

## Instructions for the assignment CA3

The third Continuous Assessment Test is devoted to the **implementation** of the application. In this CAT, you will develop an application that fulfils the user requirements considering the design decisions taken in the previous CAT.

The deliverable for this CAT should include at least the following:

* Product: Submission of the code used for your application, together with the resources (installation manual, usernames and passwords, virtual machine, ...) that can be used to test your application.
* Project report: Description of the implementation of your application as it will be featured in your project report. This description should provide a concise overview of the relevant aspects of your application (implementation decisions, high-level code organization, challenges addressed during the development, testing strategies, ...).
* Update to the project's schedule: A critical discussion of the status of the project with respect to the planned schedule. If there are significant deviations from the original schedule, you should propose actions to get the project back on track.

At the end of this CAT, you should aim for a working implementation of the core features of you application, with only minor extensions/optional features pending to be completed in the final submission.

## Learning resources CA2

### Contents and resources

Below are some guidelines for both parts of the CA. Obviously, its use will depend on the nature of the work, i.e. whether or not it includes these two parts in their entirety.

This phase is very important, because it lays the foundations for what the entire work will be. A very current habit in Engineering is "thinking with ten fingers." This is a serious problem in engineering due to the ease with which it is possible to run code and see if it really works. Although it may seem incredible, "wasting" an hour doing a good design (thinking, making diagrams, pseudocode, etc.) allows, in the long term, to reduce the number of unexpected problems (and, therefore, the development time is shorter and, the quality, higher).

A design must start from the objectives of the Final Work and define specifications (i.e. what must be met?). These specifications will be used to start developing the project.

To carry out the design of any product (Final Work included) there are several techniques:

* If the end users of the product are known, it is advisable to conduct an interview or survey to find out their needs.
* Above all, have a good knowledge of the area. If you do not have it, you must purchase it.
* Make a blog diagram of the Final Project (this also helps to structure the explanation of the work in the report).

### Product implementation

The implementation must ensure compliance with the specifications described in the design phase. Aspects such as modelling, encapsulation and documentation of the code (or hardware development) are key to guarantee high quality.

A design phase that is halfway between the previous phase and the implementation phase is the selection of the tools (software libraries or algorithms), the programming language (or languages) and the hardware to be used. It is important to prepare a good analysis of the available technologies (as has been done in the State of the art, which is in this phase), since it is not necessary to reinvent the wheel. That is, if there is already open source and/or APIs that do precisely what is needed at work and can also be used, it will be enough to learn how to use it. The time is gold!

The following are fairly standardized programming habits:

* Notation: There are several systems for declaring names of variables, constants, and classes. Among the best known is the Hungarian notation. However, the notations are not always followed 100%, but there are quite popular habits, such as: writing the classes with the first letter of each word in capital letters and without separators (e.g. Figure, SintacticLabeler, etc.). ); write variable names similar to classes but starting with a lowercase letter (e.g. playerAgeB, patientName, etc.); write the constants in capital letters and with an underscore as a separator between words (e.g. MAX\_PLAYERS).
* Documentation tools: there are various techniques to clearly and exhaustively document the code, such as Doxygen (http://www.doxygen.org) or JavaDoc (included in JDK). This is especially useful, because it may happen that in the future someone else wants to use the code, or even that the author of it wants to reuse it a few years later... if the code is not well documented, it is very difficult to understand and maintain it.
* In the event that the FP requires the development of code, and this consumes a large amount of memory and/or CPU resources, the use of "profilers" is recommended to guarantee the correct functioning and efficiency of the implemented code.