A family of experiments to evaluate the understandability of TRiStar and i∗ for modeling Teleo-Reactive systems

José Miguel Morales , Elena Navarro , Pedro Sánchez , Diego Alonso

PII: S0164-1212(16)00004-2
DOI: 10.1016/j.jss.2015.12.056
Reference: JSS 9655

To appear in: *The Journal of Systems & Software*

Received date: 21 July 2015
Revised date: 1 December 2015
Accepted date: 31 December 2015

Highlights

- An *i** extension for Teleo-Reactive (TR) systems named TRiStar.
- A novel approach to modeling software requirements of TR systems using TRiStar.
- An empirical proof of the higher efficiency of TRiStar vs *i** for TR systems.
- An empirical proof of the higher effectiveness of TRiStar vs *i** for TR systems.

A family of experiments to evaluate the understandability of TRiStar and *i** for modeling Teleo-Reactive systems

José Miguel Morales
Systems and Electronic Engineering Division (DSIE),
Universidad Politécnica de Cartagena
Campus Muralla del Mar s/n, Cartagena, Spain
josemiguel.morales@upct.es

Elena Navarro
LoUISE Research Group, Computing Systems Department,
University of Castilla- La Mancha
Avda. España s/n, 02071 Albacete (Spain)
tel +34 967 599 200 ext. 2624, fax +34 967 599 343
Elena.Navarro@uclm.es

Pedro Sánchez
Systems and Electronic Engineering Division (DSIE),
Universidad Politécnica de Cartagena
Campus Muralla del Mar s/n, Cartagena, Spain
pedro.sanchez@upct.es

Diego Alonso,
Systems and Electronic Engineering Division (DSIE),
Universidad Politécnica de Cartagena
Campus Muralla del Mar s/n, Cartagena, Spain
diego.alonso@upct.es

Abstract. *The Teleo-Reactive approach facilitates reactive system development without losing sight of the system goals.*
Objective: To introduce TRiStar as an extension of i notation to specify Teleo-Reactive systems. To evaluate whether the notational extension is an improvement in terms of effectiveness and efficiency over the original language when it is used to specify Teleo-Reactive systems.*
Method: A family of experiments was carried out with final-year engineering students and experienced software development professionals in which the participants were asked to fill in a form designed to evaluate the efficiency and effectiveness of each of the languages.
Results: Both the statistical results of the experiments, analyzed separately, and the meta-analysis of the experiments as a whole, allow us to conclude that TRiStar notation is more effective and efficient than i as a requirements specification language for modeling Teleo-Reactive systems.*
Conclusion: The extensions made on i have led to TRiStar definition, a more effective and efficient goal-oriented notation than the original i* language.*

Keywords:
Teleo-Reactive; *i**; TRiStar; Requirements Engineering; Understandability

1 Introduction

The Teleo-Reactive paradigm (TR) [1] is a goal-oriented approach for modeling systems in which actions, outputs and states are computed as a response to a stimulus received from the system’s surroundings and from the system itself. Teleo means "to reach a goal". "Reactive" means "highly sensible to perceptions". As a consequence, the TR approach offers engineers a formal high-level goal-oriented way to develop reactive systems, allowing developers to define behaviour without losing sight of the goals and state changes ocurring in the environment.

A TR specification can be defined as a set of prioritized condition/action rules. The conditions are defined by inputs from sensors or from a model of the world created by the system. The actions change the world in some way from a physical or logical point of view (the model of the world). The condition of the rule with the highest priority represents the main goal of the system-to-be. At the same time, actions can be TR specifications, thus allowing the creation of hierarchical decompositions of the goals. The subgoals are therefore those objectives to be reached in each of the sub-specifications and are needed to fullfil the main goal. For more details on this topic, see Morales et al. [2], which gives a systematic review of works published between 1994 and 2011, as well as the extensions provided by Keith Clark with TeleoR [3].

Although the TR paradigm has proved useful when it comes to specifying reactive sytems [4][5], it is nonetheless true that developing TR systems is a hard and error-prone task. The main challenges involved have been identified in [6] and can be summarized as follows:

1. *Rule priorities*: a small change in priorities or order in the rules may lead to a very different system behaviour.
2. *Regression property*: a sound TR specification must guarantee that acomplishing a subgoal takes the system closer to reaching a higher priority goal, which in turn takes the system closer to the main goal. The demonstation of this property for a given system is not a trivial issue.
3. *Modularity and encapsulation*: in spite of the fact that the paradigm considers the use of subgoals (allowing a certain degree of encapsulation) the textual representation makes the understandability of the behaviour of the system particularly difficult at a single glance.
4. *Reuse*: as a result of the above, the creation of reusable components has not been a key issue in the evolution of the TR paradigm. The most remarkable exception can be found in [7], in which the authors propose a model-driven approach to obtain architectural components starting from a TR specification.

With the aim of overcoming these difficulties, it would be useful to find a Software Engineering approach to specify the requirements of TR systems. Morales et al. [6] argue that the most suitable Requirements Engineering technique for modeling TR systems is the Goal-Oriented approach, as both systems share the same foundations (*goals*). In the study cited, two techniques are proposed that use goal-oriented requirements languages to demonstrate that *i** [8] gives better results in terms of understandability. Starting from these results and going deeper into the study of the technique based on *i**, we detected a sort of weaknesses that, if fixed, would improve the understandability, efficiency and effectiveness of *i** as a specification language for TR systems. For this reason, and following the path used in other approaches, such as [9], we propose here an *i** extension that overcomes the limitations mentioned above. This extension, named TRiStar, was first presented in [10]. In the present paper we delve deeper into the definition of TRiStar and analyze the results by means of a family of experiments carried out to compare the efficiency and effectiveness of the original notation using *i** with the TRiStar extension. It is important to clarify that TRiStar extends the *i** notation but does not limit it in any way. Thus, all the expressiveness of the original language is available to deal with topics from the early stages of requirements engineering, such as uncertainty, conflicts among multiple agents or alternative ways of achieving the same goal. All these topics may be very useful when specifying complex TR systems in which several agents collaborate or compete with each other to achieve the goals in an application (see [3] for examples of such systems).

The Oxford English Dictionary defines the word “understandable” as “that can be understood; intelligible” [11]. The understandability of a given notation is therefore something inherently subjective and linked to the modeler’s capacity to understand such notation. In this vein, many studies, besides measuring what can be called "subjective understandability", have looked for other more objective ways of evaluating understandability by means of performance-based measures. For instance, Genero et al. [12] define the concepts used throughout this document as follows:

- *Understandability Time* (UT): The time needed to understand a TR diagram (expressed in minutes).
- *Understandability Effectiveness* (UEffec): The number of correct answers reflects how well the participants performed the required understandability tasks.
- *Understandability Efficiency* (UEffic): The number of correct answers divided by UT relates the understanding performance of the participants to their effort (in terms of time spent).

In this paper we introduce a family of experiments in which the above concepts have been evaluated for each of the notations introduced: *i** and TRiStar. The rest of the paper is organized as follows: Section 2 gives an overview of related works on the development of TR systems and goal-oriented requirements engineering techniques needed to understand the contents of this paper. Section 3 gives a brief introduction to *i** and its use for defining TR systems. Section 4 describes in detail the TRiStar extension, starting from the limitations detected in *i** notation. Section 5 details the family of experiments carried out, while Section 6 describes possible threats to the validity of the experiments. Finally, Section 7 summarizes our conclusions and some worthwhile future lines of research.

2 Related work

The TR paradigm has obtained many important results in distinct fields of research, perhaps with the most valuable outcomes in the Robotics and Artificial Intelligence domain. In [2] a detailed summary of the existing literature on the TR paradigm is given, including several contributions to TR formalism, platforms for TR program simulation and validation purposes, as well as methodologic and engineering concerns for creating TR programs or generating executable code.

Among the existing Requirements Engineering approaches [13][14], Goal Oriented Requirements Engineering (GORE) has been shown to be particularly helpful in many stages of the system development process [15]. In addition, Yu and Mylopoulos state in [16] that "*some researchers have considered goals to be an important construct in a number of different areas of RE.*". Those areas include, among others, requirements acquisition, clarifying requirements or driving design, which are very useful in the latter stages of requirements specification in TR systems. Morales et al. [6] state that GORE is the most straightforward choice for developing TR systems, as both paradigms share the same fundamental concept: 'goal'. The choice of the GORE paradigm to specify TR systems is not only based on this coincidence. The search for a graphical notation to help stakeholders to understand the specification of a TR system and avoid wrong interpretations was motivated by the desire to increase the abstraction level. TR systems need a notation which allows the concept of 'goal' to be represented in the most natural possible way and at the same time specifies the rules with the appropriate level of detail. GORE offers both these advantages. Other approaches, such as the rule-based approach [17], are not suitable as they stay at the same abstraction level as that of the TR program. In addition, the mapping between TRiStar and TR programs means that the corresponding code can be obtained directly, which obviously makes the work of the developers easier. The study in [6] compares the most common GORE languages (*i** [8] with KAOS [18]) and concludes that *i** is the best GORE language to specify TR systems. In spite of the advantages of using *i**, the notation has some weaknesses when it comes to specifying TR systems, and this is why we decided to create an extension that would overcome these limitations.

There are many examples in the literature of extensions to well known languages with the aim of adapting them to specific domains. In this context, CSRML [9] (Collaborative Systems Requirements Modeling Language) is a representative extension for *i**, targeting collaborative systems to create the well-known Computer Supported Cooperative Work (CSCW). These systems allow users to do collaborative tasks, communication and coordination, besides other tasks, on common software applications. However, the specification of such systems using traditional Requirements Engineering techniques is rather complicated, while an *i** extension provides the expressiveness needed to specify CSCW more simply.

In [19], the authors make a comparative analysis of the original *i** language with its two most widespread variants: Goal-oriented Requirement Language (GRL) [20] and the language used in the TROPOS methodology [21]. It also analyzes the following three *i** extensions:

- The REDEPEND tool [22], which extends *i** and allows new types of Means-End relationships using satisfaction arguments, Contribution relationships, and other minor differences. It provides systems engineers with *i** modeling and analysis functions, coupled with additional functionality and the reliability of Microsoft Visio. It provides a graphical palette from which systems engineers can drag-and-drop *i** concepts to develop Strategic Dependency and Rationale models.
- The Formal TROPOS Language. Formal Tropos adds to *i** temporal specification primitives [23]. It allows specifying cardinality constraints in the dependencies among intentional elements and also allows a new dependency type (prior-to) to be defined to specify temporal order between intentional elements.
- In [24] the authors propose new types of dependencies among actors and intentional elements: responsibility dependencies between an agent and a goal or a task; authority dependencies between two agents; audit dependencies between an agent and a goal or a task; and capability dependencies of an agent with respect to a goal or task.

On the other hand, controlled experiments to determine the understandability of a given notation or language is a widely accepted practice. Jamison and Teng [25] carried out an experiment to determine the perceived ease of use of several types of textual and graphical database representations. They concluded that graphical notations were more easily and efficiently accessed and the participants declared that graphical representations were much easier to understand.

Lee and Choi [26] compared a set of conceptual data-modeling languages to determine which gave more accurate and understandable models in the shortest time. The best results were obtained by the Extended Entity-Relationship Model (ERM) and the Object Modeling Technique. Bajaj [27] studied the influence of the number of metamodel concepts on the readability of schemes created using such metamodels. They carried out an experiment using many variants of the original ERM, each one with a different number of concepts in order to evaluate efficiency, effectiveness and learnability (defined as an improvement in efficiency and effectiveness over time). The results led the authors to conclude that the variants with most concepts allowed higher precision in the domain conceptualization and at the same time were easier to learn, although the time needed to process the schemes was increased significantly.

Many other approaches are based on ERM: in [12] a set of objective metrics were defined on ER diagrams and an experiment was performed to determine whether these metrics had any correlation with the "subjective understandability", efficiency and effectiveness of ER diagrams. Three of the proposed metrics (number of entity attributes, number of 1:1 relationships, and number of 1:N relationships) were significantly correlated with scheme understandability: the more attributes and relationships a diagram had, the less understandable it turned out to be.

A family of experiments was carried out in [28] to compare the understandability of *i** and CSRML when specifying Collaborative Systems in which the users could perform collaborative, communication and coordination tasks. Similarly to the system used in the present study, they used two replicas in which the subjects answered a set of questions related to the understandability of the two notations. The statistical analysis showed that the specifications made by CSRML scored higher than *i**, especially in collaboration aspects. The study concluded that in terms of understandability CSRML outdid *i** as a specification language for collaborative systems.

More recently, a controlled experiment was performed in [6] to determine the understandability of *i** versus KAOS as a language for specifying TR systems. The results showed that both languages obtained similar scores in terms of understandability, although *i** notation stood out slightly. The statistical analysis of the results led to the conclusion that *i** notation was more understandable than KAOS as a specification language for TR systems.

Following the strategy defined in [6], the aim of the present study is to statistically validate whether or not the notational extensions are an improvement of the original *i** notation by means of a family of experiments

3 Previous background: *i** for TR system requirements specifications

The *i** framework guides the stakeholders through the different phases of the software development process, namely from the early requirements analysis up to the detailed design. As already mentioned, *i** can also be employed to specify the requirements of TR systems.

The work by Morales et al. [6] introduces the language and gives a detailed description of the technique developed for specifying TR systems. Table 1 briefly summarizes the mapping from i* concepts to TR concepts, which constitutes the kernel of the technique described in the work.

Table 1. Mapping concepts between *i and TR**

i*	***TR***

Main Agent	System-to-be
Agent	Sensor
Goal	Goal
Task	Action
Resource	Percept
Resource Dependency	Condition

Figure 1 shows the application of this approach by using a simplified version of the *i** specification of one of the examples used in the family of experiments described in Section 5.2. The example consists of a drone that always goes back to its origin, no matter where it has taken off from.

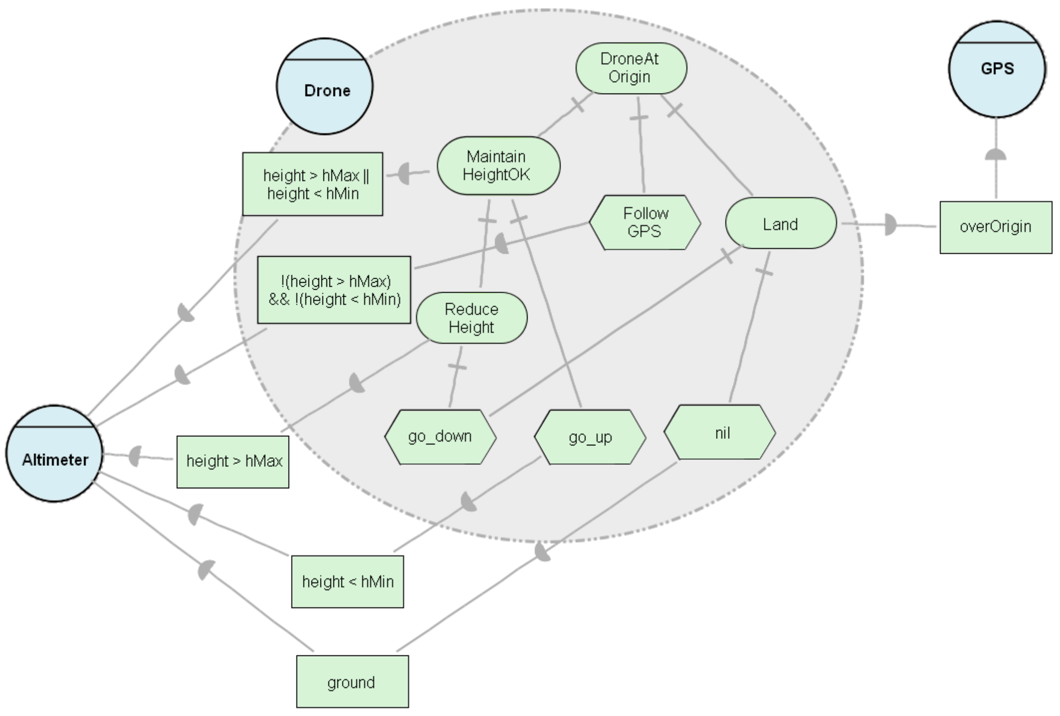


Figure 1. Drone application specification using *i**

The details of the application of the tecnique described in [6] to the *i** specification shown in Figure 1 are given below. The resulting TR program is shown in Code 1:

- Every *i** goal in Figure 1 becomes a TR goal (in bold text in Code 1, as for instance, *Land* or *MaintainHeightOK)*.
- Every *i** agent becomes a sensor or device, such as *GPS* or *altimeter*.
- Every *i** resource that has a dependency relationship with an agent becomes a condition monitored by the homologous sensor, such as *ground* with the *altimeter*. *i** goals or tasks that lack dependency relationships, such as *go_down*, are mapped to TR rules whose condition is always true (*True* → *goal/action*).
- Every *i** task, such as *followGPS*, becomes an action.

- *i** tasks and goals linked by a task-decomposition link to an *i** goal become rules of the same TR goal. For example, the *Land* goal is decomposed into the tasks *go_down* and *nil*. Therefore, two rules are created for the TR goal named *Land*: one whose action is *go_down* and another whose action is *nil*, as shown in Code 1.
- The relative position of the items in the i* specification states the priority of the rules in the TR program. For example, *Land* is drawn farther to the right than *followGPS* (see Figure 1). As can be seen in Code 1, the rule whose action is *Land* is over the rule whose action is *followGPS* because it has higher priority.

Code 1. TR program for Figure 1

```
DronAtOrigin:
        overOrigin → Land
        NOT(height > hMax) AND NOT(height < hMin) → followGPS
        height > hMax OR height < hMin → MaintainHeightOK
MaintainHeightOK:
        height < hMin → go_up
        height > hMax → ReduceHeight
ReduceHeight:
        True → go_down
Land:
        ground → nil
        True → go_down
```

3.1 Shortcomings of i* for TR systems

As shown in the previous section, it is possible to specify the requirements of a TR system using *i**. Although the validity of the proposed mapping between *i** and TR programs has been established in previous works [6] [10], in this last paper the authors pointed out some limitations found in applying such a technique and briefly presented an extension to *i** named TRiStar, which aims to overcome them. We firstly summarize these limitations by an illustrative example, while the following section describes the enhancements provided by TRiStar for specifying TR systems employing *i**.

- **S1**. Setting the priority by using the order in which tasks or goals refining a goal are positioned in the diagram constrains the likely position of subtasks or subgoals in it. Occasionally, this may result in messy diagrams hard to interpret. In addition, it is difficult to automatically process a diagram in which the relative position of two items has an important meaning. The example shown in **Figure 2**, a variation of Figure 1, will help us to explain this shortcoming. Task *go_down* must be placed far away from the *Land* goal because it needs to be on the left of *go_up*, as the priority of *go_down* when refining *MaintainHeightOK* is lower than that of *go_up*.

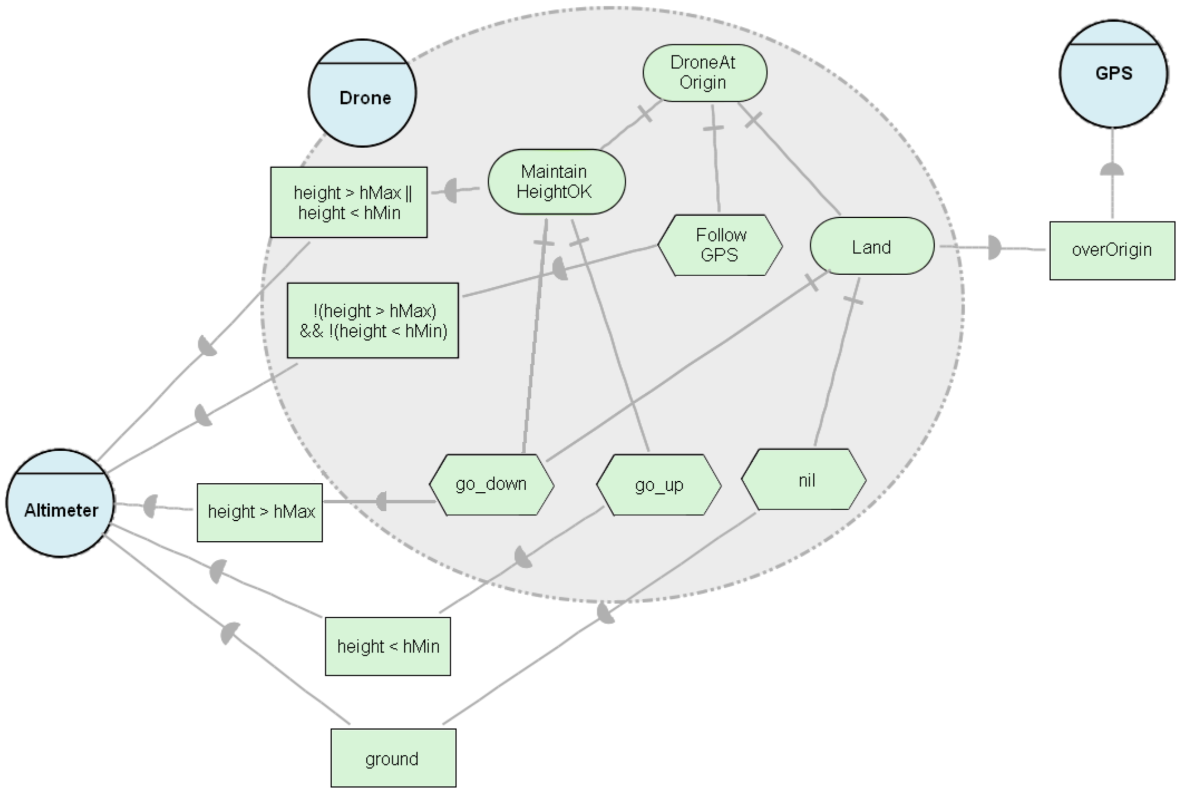


Figure 2. Ambiguous *i specification**

- **S2**. If the same task is involved in two or more goals, it may then depend on different resources when refining one of the goals. This may cause ambiguity when obtaining the conditions of the associated TR rule. Considering the *i** specification of the drone shown in Figure 2, it can be seen that it is is very similar to that of Figure 1. In this case we introduced a goal named *ReduceHeight* to avoid the ambiguity around the *go_down* task. If this artificial goal is not used (note that there is no goal *IncreaseHeight*, as there is no possible ambiguity with the *go_up* task), it is not possible to say whether *go_down* depends on *height > hMax* when refining *MaintainHeightOK* or when refining *Land*, or in both cases. Code 2 shows a TR program that fits this specification. Note the condition of the lowest rule in the subgoal *Land*; a drone programmed with Code 2 would crash when it flew over its origin. When *overOrigin* became true, the subgoal *Land* would take control, but as none of the conditions of the rules in *Land* are actually true, the drone would do nothing and thus would fall to the ground.

Code 2. A TR program fitting the specification in Figure 2

```
DronAtOrigin:
        overOrigin → Land
        NOT(height  >  hMax)  AND  NOT(height  <  hMin)  →
followGPS
        height > hMax OR height < hMin → MaintainHeightOK
MaintainHeightOK:
        height < hMin → go_up
        height > hMax → go_down
Land:
        ground → nil
        height > hMax → go_down
```

We got around this problem in the *i** version shown in Figure 1 by using the additional subgoal *ReduceHeight*. The resource *height > hMax* depends on this new subgoal, which is decomposed into only one task, freeing this task (*go_down*) from dependencies. With this alternative specification a correct TR program can be generated, but the extra item needed reduces the diagram's readability.

- **S3**. The conditions of TR rules are usually composed of logical combinations of percepts given by the sensors. In *i** there is no way to graphically represent a Boolean combination of some of the percepts provided by sensors. Retaking the example shown in Figure 1, in *i** there is no symbol to represent a dependency on *height > hMax* OR *height < hMin*, for instance. We got around this limitation by adding resources that are labeled with the Boolean expression we wanted to represent inside the boundary of the system. These expressions may become difficult to read in systems with a certain degree of complexity.

4 TRiStar Enhancements

To overcome the limitations identified in the previous section, an extension to *i** is proposed. The following three main new features compose this extension, named TRiStar:

- E1: Prioritized decomposition links.
- E2: Dependent decomposition links.
- E3: Logical resources.

In this section these new features are described in depth.

- **E1**: **Prioritized decomposition links**. To avoid relying on the relative position of the diagram elements when information about their priority is needed (shortcoming S1), a new decomposition link has been defined. This new type of decomposition link provides the priority of the rule whose subgoal or task is at the end of the link by changing its own representation. The standard *i** decomposition link has a short perpendicular line at the end that is closer to the goal being decomposed. The new decomposition links have as many of these short lines as needed to show the priority of the subgoal or subtask. One line means the lowest priority. The more lines a decomposition link has, the higher priority its related subgoal or subtask has. Now, the position of these lines on the diagram does not have any intended meaning. Figure 3 shows a diagram in which these new decomposition links can be seen. Note that although the *highest_priority_subgoal* is in the middle of the subgoals, it has the highest priority as its decomposition link has three short perpendicular lines. In the TR program corresponding to this specification, the rule containing *highest_priority_subgoal* would be the uppermost rule in *Goal*, then *medium_priority_subgoal* would be next, and finally *lowest_priority_subgoal*.

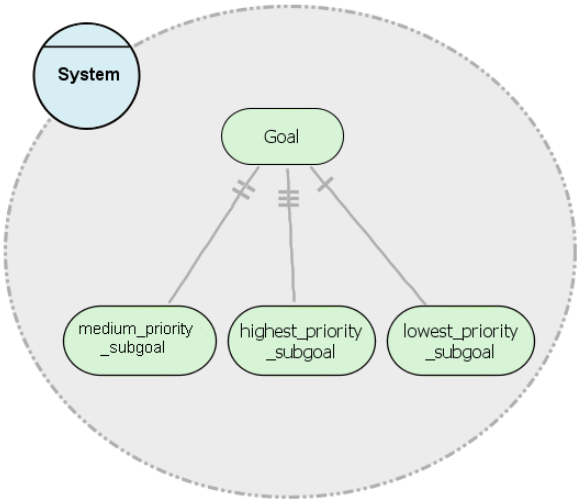


Figure 3. Prioritized decomposition links

In order to avoid scalability problems when a goal is refined into many tasks or subgoals, the short perpendicular lines can be substituted by a circle with the priority specified in its interior, with '1' being the lowest priority. Although this notation facilitates the insertion of new subtasks or subgoals and avoids the excessive cluttering that can be generated by the addition of many perpendicular lines, we recommend the use of short lines to maintain the similarity with the original *i** notation.

- **E2**. Dependent decomposition links have been introduced to avoid linking dependencies directly to subgoals or tasks (limitation S2). It is worth remembering that the condition of a rule in a TR program cannot be generated from a dependency on a task or subgoal alone, but the relationship between the task and the goal it refines is also needed. This relationship is obviously represented by the decomposition link that connects them and explains why a dependency link between the decomposition link and the resource has been introduced. For example, as Figure 4 shows, the decomposition link between *Goal* and *Subgoal* depends on *resource1*, which is in *Sensor1*'s boundary. Similarly, the decomposition link between *Goal* and *Task* depends on *resource2*, which is in *Sensor2*'s boundary.

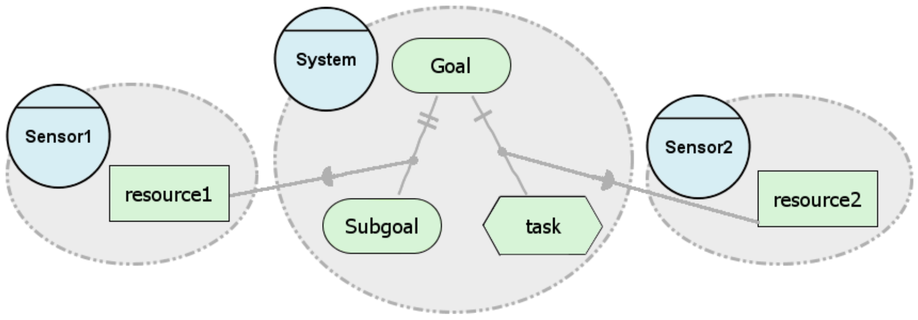


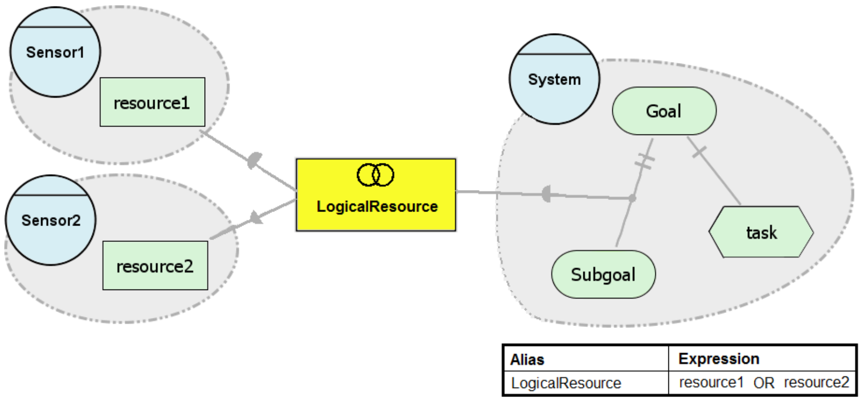
Figure 4. Dependent decomposition links

Note that the new prioritized decomposition link has been used in the example. As the link between *Goal* and *Subgoal* has a higher priority than the other, the first rule in the TR program is the one whose condition is *resource2*. **Code 3** shows the TR program generated from this specification.

Code 3. TR program for Figure 4

```
Goal:
          resource1 → Subgoal
          resource2 → task
```

- **E3**. To overcome limitation S3, we introduced a specialization of *i** resources to represent the logical combinations of percepts. These specialized resources are related to all the percepts they involve by using directed dependency links. In addition, the *logical resource* is given a name, which acts as an alias for such combinations of percepts. A table is provided to link every name with its logical expression. Figure 5 shows an example of this new kind of resource.



Alias	Expression
LogicalResource	resource1 OR resource2

Figure 5. Use of logical resources

The decomposition between *Goal* and *Subgoal* depends on a logical resource which is the result of an OR operation between *resource1* and *resource2*. As the logical resource uses the percepts *resource1* and *resource2* as operands, dependency links are established from the logical resource to its two operands. The expression represented by *LogicalResource* can be seen in the table just under *System*'s boundary. **Code 4** shows the TR program that corresponds to this specification:

Code 4. TR program for Figure 5

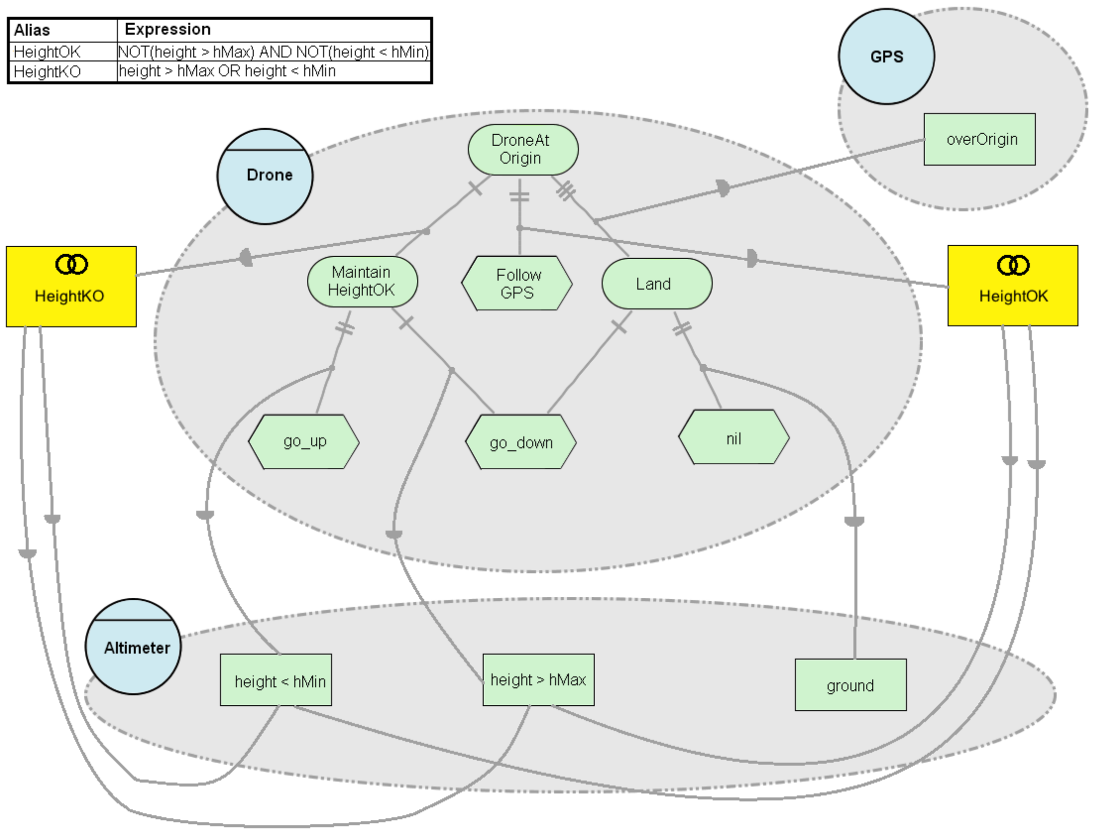
```
Goal:
      resource1 OR resource2 → Subgoal
      True → task
```

Lastly, although it cannot be considered an extension to *i**, the dependencies between a percept and the sensor that generates it are represented by inserting the resource inside the agent's boundary, as already shown in **Figure 4** and **Figure 5**. In this way, only a dependency link is needed to represent the condition of a rule, unlike plain *i** specifications, which require two such links.

The mapping from a TRiStar specification to a TR program is very similar to that of *i**. In fact, Table 1 still remains valid. There are however some differences:

1. The main TRiStar agent is transformed into the TR system-to-be. The main TRiStar agent is the one that has the goal that the final system wants to achieve in its boundary.
2. TRiStar goals become TR goals.
3. TRiStar tasks are specified as TR atomic actions.
4. TRiStar resources (except logical resources) become percepts generated by sensors.
5. A logical resource will be translated into the expressions found in the table associated to its alias.
6. Considering that a TR rule is defined as *condition → goal/action*, every TRiStar resource having a decomposition link as a dependee is transformed into a TR rule whose condition is that resource and its action is the task or goal that is at the end of the decomposition link. A decomposition link not depending on any resource is transformed into a rule of the form *True → goal/action*.
7. Since a TR goal is defined as a set of prioritized TR rules, a TRiStar goal being refined into goals and tasks through task-decomposition links is transformed into a TR goal formed by as many rules as TRiStar tasks or goals refine the original *i** goal.
8. Rule priority, given by the order of the rules in TR programs, is specified in TriStar diagrams by using prioritized decomposition links. So, the tasks or goals placed at the end of the highest priority decomposition link will be translated into the action of the highest priority rule in the TR program. The resource on which the highest priority decomposition link depends will be transformed into the condition of that rule.

Figure 6 shows the specification of the same drone as that in Figure 1 but using TRiStar.



Alias	Expression
HeightOK	NOT(height > hMax) AND NOT(height < hMin)
HeightKO	height > hMax OR height < hMin

Figure 6. Drone specification using TRiStar.

All the proposed extensions have been employed in this example:

- Prioritized decomposition links allow positioning *go_down* near both *MaintainHeightOK* and *Land*, which helps keep the diagram organized and uncluttered, with no crossing lines.
- Dependent decomposition links enable the artificially created subgoal *ReduceHeight* to be removed. The resource *height* > *hMax* depends on the link between *MaintainHeightOK* and *go_down* and not on the link from *Land*, so that the ambiguity of the rule condition is eliminated.
- Two logical resources have been introduced: *HeightOK* and *HeightKO*, whose expressions can be found in the table in Figure 6. The aliases make it easier to understand the conditions that apply to the rules involved.

Code 5. TR Program for Figure 6

```
DronAtOrigin:
        atOrigin → Land
        NOT(height > hMax) AND NOT(height < hMin) → followGPS
        height > hMax OR height < hMin → MaintainHeightOK
MaintainHeightOK:
        height < hMin → go_up
        height > hMax → go_down
Land:
        ground → nil
        True → go_down
```

Code 5 shows the TR program obtained by applying the mapping rules described in Section 4 to the TRiStar specification depicted in Figure 6:

- Every TRiStar goal in Figure 6 becomes a TR goal (in bold text in Code 5, as for instance, *Land* or *MaintainHeightOK).*
- Every TRiStar agent becomes a sensor or device, such as *GPS* or *altimeter*. The resources within their boundary are mapped to the percepts provided by each of them. See *ground* inside *Altimeter's* boundary, for example.
- A decomposition link depending on a resource, logical or not, becomes a rule whose condition is the percept represented by the resource and its action is the goal or task at the end of the decomposition link. See for example in Figure 6 the link between *DronAtOrigin* and *Land*, which depends on *atOrigin*. It is mapped to the first rule in goal *DronAtOrigin* as can be seen in Code 5.
- Logical resources are mapped to the conditions corresponding to their aliases in the table. For instance, *HeightKO* is mapped to `height > hMax OR height < hMin`.
- Decomposition links that lack dependency relationships, such as that between *Land* and *go_down*, are mapped to TR rules whose condition is always true (*True → goal/action*).
- Just as in the *i** case, every TRiStar task, such as *followGPS*, becomes an action.
- TRiStar tasks and goals linked by a task-decomposition link to a TRiStar goal become rules of the same TR goal. For example, the goal *Land* is decomposed into the tasks *go_down* and *nil*. Then, a TR goal named *Land* appears with two rules: one whose action is *go_down* and another whose action is *nil*, as shown in Code 5.
- The number of short perpendicular lines in the decomposition links states the priority of the rules in the TR program. For example, the decomposition link from *DronAtOrigin* to *Land*

has three of these perpendicular lines, while the decomposition link between DronAtOrigin and *followGPS* has only two (see Figure 6). As can be seen in Code 5, the rule whose action is *Land* appears before the rule whose action is *followGPS*.

5 The family of experiments

In order to assess the *understandability* of both the newly created TRiStar extension and *i** when modeling the software requirements of TR systems, a family of experiments (see Figure 7) performed to compare both of them based on the guidelines described by Kitchenham et al. [29]. In this section we will describe the context, the design and how the experiments were conducted. All the three members of the family were designed in a similar way, so that only one description is given.

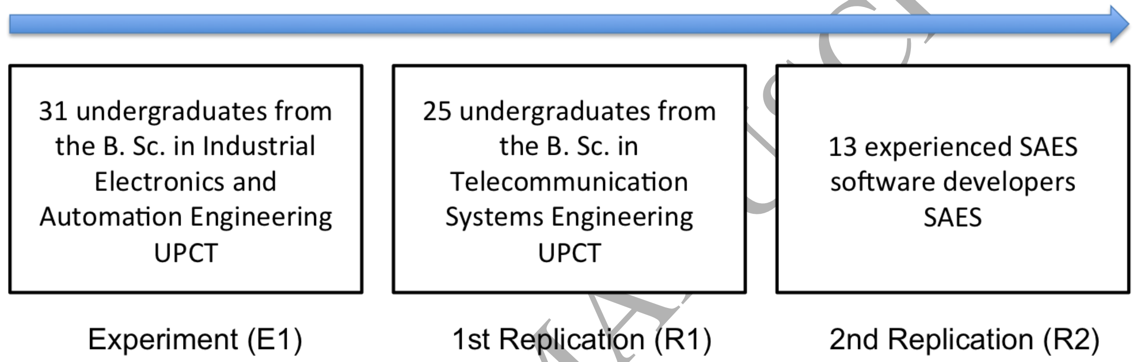

31 undergraduates from the B. Sc. in Industrial Electronics and Automation Engineering UPCT
25 undergraduates from the B. Sc. in Telecommunication Systems Engineering UPCT
13 experienced SAES software developers SAES
Experiment (E1)
1st Replication (R1)
2nd Replication (R2)

Figure 7. Chronology of the family of experiments

5.1 Experimental Context

The main goal of this family of experiments was to study the requirements specifications of TR systems using both *i** and TRiStar and evaluate their effectiveness and efficiency from the perspective of requirements engineering researchers, using undergraduate B. Sc. students and experimented software developers as subjects. To achieve this goal, the null hypotheses shown in Table 2 were defined using the Goal Question Metric template [30].

Table 2. Main features of the family of experiments

Null-Hypotheses	$H_{UEffee0A}$: *i** has the same average score for understandability effectiveness as TRiStar when specifying TR requirements. $H_{\mathbf{UEffec1A}}$: ¬$H_{\mathbf{UEffec0A}}$ $H_{UEffec0B}$: The understandability effectiveness average score is the same regardless of the domain used in the experiment. $H_{UEffec1B}$: ¬$H_{UEffec0B}$ $H_{UEffec0AB}$: *i** has the same understandability effectiveness average score as TRiStar when specifying TR systems requirements, regardless of the domain used in the experiment and viceversa. $H_{UEffec1AB}$: ¬$H_{UEffec0AB}$ $H_{UEffic0A}$: *i** has the same average score for understandability efficiency as TRiStar when specifying TR requirements. $H_{\mathbf{UEffic1A}}$: ¬$H_{\mathbf{UEffic0A}}$ $H_{UEffic0B}$: The understandability efficiency average score is the same regardless of the domain used in the experiment. $H_{UEffic1B}$: ¬$H_{UEffic0B}$ $H_{UEffic0AB}$: *i** has the same understandability efficiency average score as TRiStar when specifying TR systems requirements, regardless of the domain used in the experiment

	and viceversa. $H_{UEffic1AB}$: $\neg H_{UEffic0AB}$		
Dependent variables	Understandability effectiveness of requirements modeling languages, measured by UEffec Understandability efficiency of requirements modeling languages, measured by UEffic		
Independent variables	The *system* the models specify and the *language* used to specify these models		
Location	ETSII at UPCT (Cartagena, Spain)	ETSIT at UPCT (Cartagena, Spain)	SAES Facilities (Cartagena, Spain)
Date	February 2015	February 2015	February 2015
Subjects	31 undergraduates of the B.Sc. in Industrial Electronics and Automation Engineering (16 Group 1; 15 Group 2)	25 undergraduates of the B.Sc. in Telecommunication Systems Engineering (13 Group 1; 12 Group 2)	13 experienced software development professionals (6 Group 1; 7 Group 2)

As Table 2 shows, the subjects in the experiments were engineering students and software development professionals. All were familiar with requirements engineering but none had previously used either *i** or any other GORE language and none had any previous experience of TR systems.

The *Sociedad Anónima de Electrónica Submarina* (SAES) collaborated in this study and allowed almost all their software engineers to be subjects for the second replication. SAES is a Spanish company specializing in underwater acoustics and develops undersea security and environmental protection systems. The company has more than 25 years of experience in developing advanced technology in the fields of Sonar, Acoustic Signal Processing, Underwater Signature Measurement and Management, Simulation and Training. Highly skilled and experienced engineers and scientists in various disciplines make SAES an innovative and competitive company in both national and international markets.

5.2 Experimental Design

All the experiments in this family were aimed at evaluating the understandability of the requirements specification of two different TR systems specified by both *i** and TRiStar. The first system consisted of a drone which was able of deliver a package to a destination and go back to its origin, always keeping at a safe height. GPS informs the drone when it is flying over its origin, over the destination and gives it directions to reach both places. Weight is monitored so that the drone knows whether it is loaded or not and an altimeter is in charge of updating height information. The actions the drone is able to carry out are limited to going up, going down, following GPS directions and releasing the load.

The second system was a variation of one of the systems used in [6], which was a soccer robot which plays in defensive positions. When the robot considers the danger is over, it goes back to its own goal. The robot can find the ball and knows who is controlling the ball: i.e. himself, an opponent or a teammate. The robot can identify other members of its own team, in fact, its main goal is to keep the ball in his team’s possession. To do this, the robot can turn, move forward, dribble and kick the ball.

The subjects in all the tests were divided into two groups, Group 1 and Group 2, each group using one of the languages. Table 3 summarizes these decisions:

Table 3. Experimental design

		System	
		Drone	**Football player**
Language	**TRiStar**	Group 1	Group 2
	i*	Group 2	Group 1

Dividing the subjects into 4 different groups starting from the combination of the two independent variables makes up a 2x2 factorial design with confounded interaction [31] and thanks to this combination system-language among the groups, the *learning effect* is cancelled. Every subject answered a brief questionnaire on both models. The questionnaire (see Appendix) consisted of some TR program fragments from the presented models using the appropriate mapping. In every fragment there was an element missing and the subject was asked to fill in the blanks. They were also asked to record the time they need to answer the questions. With this information, effectivenes (UEffec) was calculated as the number of correct answers divided by the total number of questions. Efficiency (UEffic) was calculated as UEffec divided by the number of minutes required to fill in the questionnaire. In the final question the subjects were also asked which language they thought was most understandable in specifying TR systems.

Since all the participants had previous experience in requirements engineering but not in GORE or TR systems some filtering criteria were laid down to eliminate any subjects whose previous experience would give them an advantage that could adulterate the results. Those that matched any of the following criteria were discarded:

- Those more than 5 years older than the group’s average age.
- Previous experience in GORE languages.
- NO previous experience in requirements engineering
- Previous experience in TR systems.

Finally, each subject was interviewed on his opinion of the questions and the answers were recorded for subsequent analysis.

5.3 Test Procedure

The tests were carried out in three different sessions: one for the original test and two more for the replicas. The first session took place in the Industrial Engineering School of the University of Cartagena and the second in the Telecommunications Engineering Faculty of the same university. The third session took place in the SAES facility in Cartagena.

The same procedure was used for all three sessions. An instructor initially briefed the subjects on TR systems, *i** and TRiStar, and how to represent TR systems requirements in both notations. The examples used in the experiment, the drone and the football player, were also described. The time needed for the complete briefing was about 20 minutes. Before giving the models to the subjects, the following information was obtained:

- For subjects in groups G1 and G2:
  - Gender (Male/Female)
  - Age
  - Qualifications
  - Average score
  - Have you had any previous experience of working with goal-oriented requirements engineering?
  - Have you had any previous experience of working with any other requirements engineering technique?
  - Have you had any previous experience of working with teleo-reactive systems?
- For subjects in group G3:
  - Gender (Male/Female)
  - Age
  - Years of experience in software development
  - Have you had any previous experience of working with goal-oriented requirements engineering?
  - Have you had any previous experience of working with any other requirements engineering technique?
  - Have you had any previous experience of working with teleo-reactive systems?

The subjects were asked to record their exact start and end times from an online clock projected on a screen.

5.4 Analysis of the Results

Error! Reference source not found. shows the participants' subjective preference in the form of the combined answers for the three experiments to the question "In your opinion, which language is more understandable?".

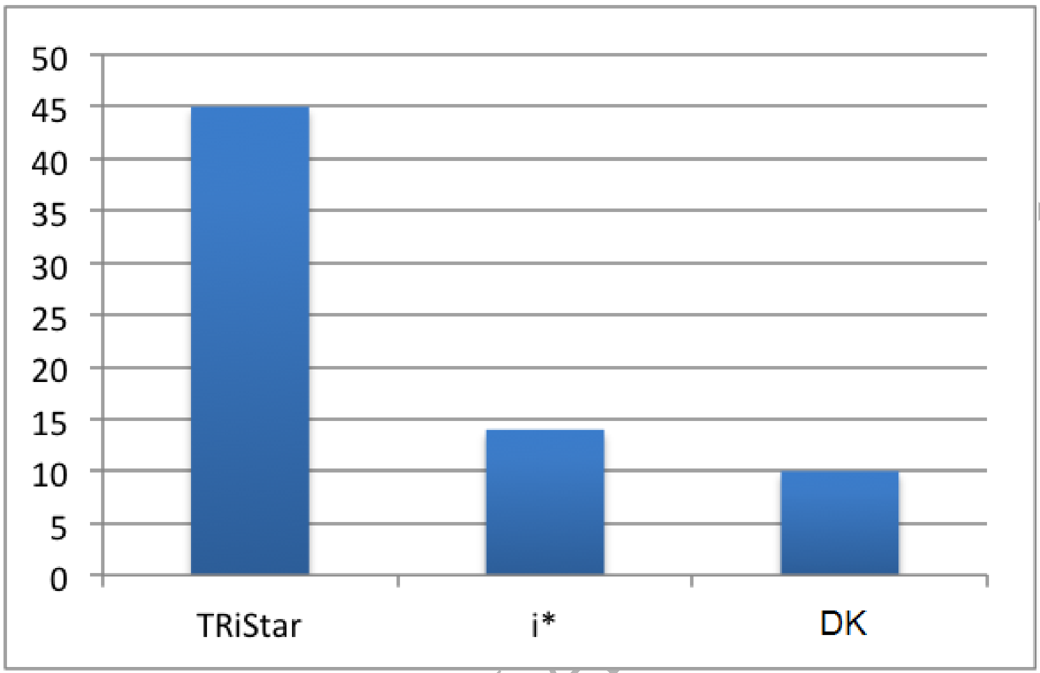


Figure 8. Subjective Understandability.

As can be seen in Figure 8, the answers show that TRiStar is more understandable than *i**. 45 subjects declared that they found TRiStar more understandable, vs less than 15 who preferred *i** or the 10 people who did not give a clear answer (Don't Know). As regards the effictiveness and efficiency aspects; the factorial design of the experiments in this family makes them particularly appropriate for a two way ANOVA test in order to analyze the results. The three main assumptions for this test are the following:

- Independence of observations.
- The distribution of the residuals must be normal.
- Homocedasticity: homogeneity of variances.

In the following subsections the original experiment and its replications will be analyzed to check firstly whether these assumptions are achieved or not. In those cases in which the assumptions are achieved, the results of the ANOVA tests will be presented and anlyzed. The way in which the data was obtained guarantees the independence of the observations, so that only normal distribution and homocedasticity need be proven. The results were analyzed by IBM SPS Statistics v. 22.

5.4.1 Original Experiment (E1)

The answers of 5 participants in this test were discarded from the sample either because they did not comply with one of the criteria in Section 5.2 or they had not completed the questionnaires. The total number of remaining subjects in the sample was 31. According to the central limit theory [32], the normality of the sample may be assumed.

UEffec: as can be seen in column "Sig." in Table 4, Levene's test [33] for homogeneity of variances provides a p-value of 0.493, allowing us to assume the homocedasticity of the sample. This test was designed to fit the two-way ANOVA test to be performed: language + system + language * system, each of these elements corresponding to one of the three null hypotheses to be evaluated ($H_{UEffec0A}$, $H_{UEffec0B}$ and $H_{UEffec0AB}$).

Table 4. Levene's test for UEffec in E1

F	df1	df2	Sig.
0,814	3	58	0,493

The results provided by the ANOVA test are shown in Table 5.

Table 5. ANOVA results for UEffec in E1

Source	Type III Sum of squares	df	Mean Square	F	Sig.
Model	33.907[a]	4	8.477	415.250	0.000
Language	0.085	1	0.085	4.154	0.047
System	0.016	1	0.016	0.797	0.376
Language * system	0.012	1	0.012	0.573	0.453
Error	0.980	58	0.020		
Total	34.887	62			
a. R Squared= 0.972 (Adjusted R Squared = 0.970)					

As the p-value obtained for language is 0.047 (see column "Sig.") and therefore less than $\alpha = 0.05$ $H_{UEffee0A}$ can be rejected and it can be concluded that there is a statistically significant difference between the UEffec results obtained from *i** and those obtained from TRiStar. On the other hand, as the p-values for system and language*system are much bigger than α, neither

$H_{UEffee0B}$ nor $H_{UEffee0AB}$ can be rejected. We can thus be sure that language influences UEffec, but neither the system nor the combination of language and system does so.

To calculate the confidence interval of the mean differences between *i** and TRiStar: [-0.15986, -0.00168], as all the values in the interval are less than 0, we can say with a 95% confidence level that the effectiveness of TRiStar is higher than that of *i** when modeling TR systems.

Table 6 shows the homocedasticity of the sample for UEffic as it provides a p-value of 0.869.

Table 6. Levene's test for UEffic in E1

F	df1	df2	Sig.
0.238	3	58	0.869

Table 7 shows the results of the ANOVA test. As with UEffec, $H_{UEffie0A}$ may be rejected but $H_{UEffie0B}$ or $H_{UEffie0AB}$ may not, given the p-values obtained for language (0.029), system (0.309) and language*system (0.091). Then, as in the case of effectiveness, we can conclude that the language used does affect the efficiency, but the system or the combination of language and system does not.

Table 7. ANOVA results for UEffic in E1.

Source	Type III Sum of squares	df	Mean Square	F	Sig.
Model	1.388[a]	4	0.347	47.809	0.000
Language	0.036	1	0.036	5.008	0.029
System	0.008	1	0.008	1.052	0.309
Language * system	0.021	1	0.021	2.953	0.091
Error	0.421	58	0.007		
Total	1.809	62			
a. R Squared = 0.767 (Adjusted R Squared = 0.751)					

Once we know that language does affect UEffic, we will obtain the confidence interval of the mean differences between *i** and TRiStar in order to determine which language obtains the best results. The calculated interval is [-0.09175, -0.00373] and we can conclude at a 95% confidence level that TRiStar is more efficient than *i** when specifying TR systems requirements.

5.4.2 First Replication (R1)

For the first replication of the experiment, after discarding 6 students that did not comply with the criteria in Section 5.2, we had a sample of 25 subjects, whose normality had to be shown as the sample size was less than 30. As the ANOVA test is robust before moderated normality deviations, a graphical proof of the distribution was enough. Figure 9 contains a box graph showing the normality of the UEffec distribution:

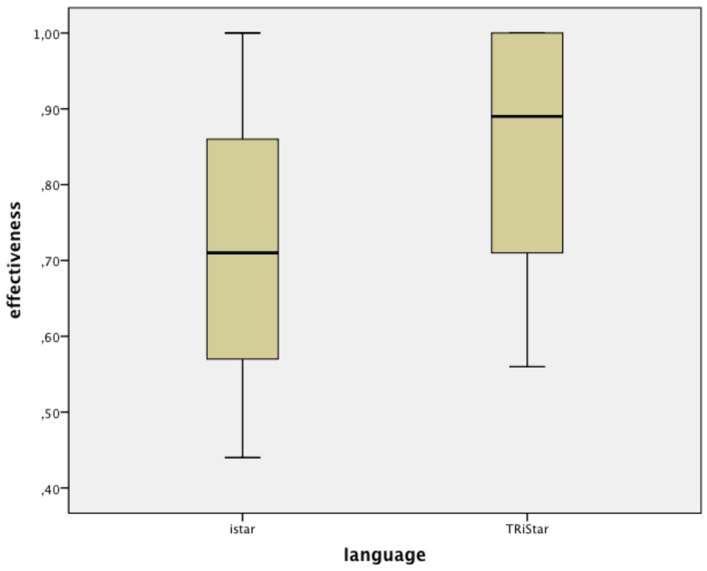


Figure 9. Normal distribution of UEffec in R1

As in the case of E1, a Levene's test was performed to check the homocedasticity of the samples. This test is also designed to ensure the homogeneity of variances for language, system and the combination of both (language * system). The results are shown in Table 8 in which a p-value of 0.922 can be seen to prove the homogeneity of the error variances.

Table 8. Levene's test for UEffec in R1

F	df1	df2	Sig.
0,16	3	46	0.922

After checking the assumptions, the ANOVA test was performed and the results are shown in Table 9. In this case, the p-values displayed in column "Sig." for language (0.017), system (0.437) and language*system (0.101) support the same conclusions as in E1: only language affects the effectiveness of the specification.

Table 9. ANOVA results for UEffec in R1.

Source	Type III Sum of	df	Mean Square	F	Sig.

	squares				
Model	29.151[a]	4	7.288	254.738	0.000
Language	0.177	1	0.177	6.188	0.017
System	0.018	1	0.018	0.616	0.437
Language * system	0.080	1	0.080	2.801	0.101
Error	1.316	46	0.029		
Total	30.467	50			
a. R Squared = 0.957 (Adjusted R Squared = 0.953)					

To show that TRiStar provided a better UEffec value, the confidence interval of the mean differences between *i** and TRiStar was calculated: [-0.2152, -0.02]. As all the values in the interval were lower than 0, TRiStar obtained the best language effectiveness values. In other words, TRiStar is more effective when specifying TR systems requirements.

Figure 10 shows the normality of the UEffic samples:

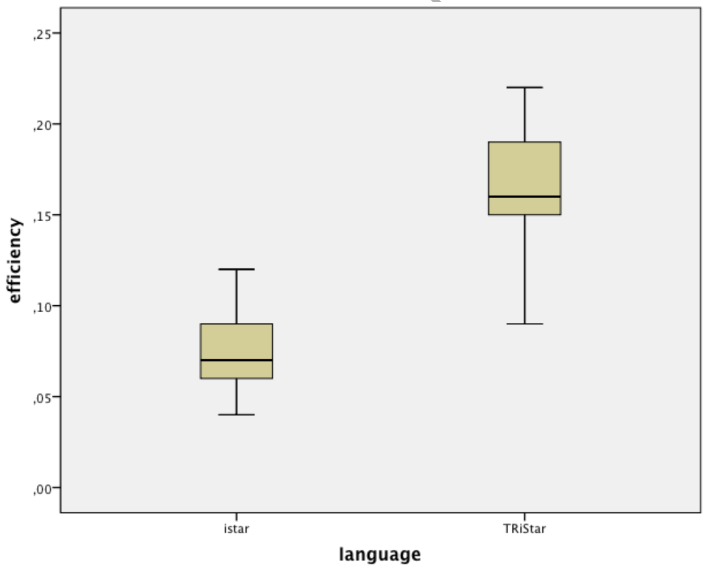


Figure 10. Normal distribution of UEffic in R1

Homocedasticity was proven again using Levene's test (see Table 10). The small p-value (0.086) obtained was still higher than 0.05 and therefore the null hypothesis of the homogeneity of variances could be assumed.

Table 10. Levene test for UEffic in R1

F	df1	df2	Sig.
2.336	3	46	0.086

Table 11 summarizes the results of the ANOVA test to analyze UEffic in this experiment:

Table 11. ANOVA results for UEffic in R1.

Source	Type III Sum of squares	df	Mean Square	F	Sig.
Model	0.789[a]	4	0.197	230.751	0.000
Language	0.084	1	0.084	98.500	5.17×10^{-13}
System	0.000	1	0.000	0.198	0.658
Language * system	0.002	1	0.002	1.933	0.171
Error	0.039	46	0.001		
Total	0.829	50			

a. R Squared = 0.953 (Adjusted R Squared = 0.948)

The p-value for language is 5.17×10^{-13} so that $H_{UEffie0A}$ can be rejected. The p-values for system (0.658) and language*system (0.171) do not allow us to reject $H_{UEffie0B}$ or $H_{UEffie0AB}$ thus reaching the same conclusion as in E1: language affects the efficiency of the specifications but system or the combination of both do not.

As in the previous cases, the confidence interval of the mean differences was calculated, giving [-0.09878, -0.06522]. As the whole interval was formed by negative values, we could conclude that TRiStar was more efficient than *i** in specifying TR systems.

5.4.3 Second Replication (R2)

The small sample of the second replication (13 subjects) forced us to check the normality of the distribution. This was not possible for effictiveness because the sample hugely deviated from normality, so we could not use an ANOVA test. As the use of non-parametric tests is recommended for this type of sample, we chose the Kruskal-Wallis test to check the equality of the distributions among the categories of the samples. As this test only allows one factor to be checked at a time, two tests were necessary: one for language and one for system.

Kruskal-Wallis does not need the normality assumption but it does need the homocedasticity condition. To prove this, we performed a Levene’s test for language that provided a p-value of

0.079 and another for system, giving a p-value of 0.359. These results are summarized in Table 12.

The Kruskal-Wallis test provided a p-value for language of 0.046 with a significance level of 0.05, which indicated that language did affect the UEffec distribution. We obtained a p-value of 0.217 for system, which prevented us from concluding that UEffec was affected by the system. Therefore, if only language affects the UEffec distribution and taking into account the distribution shown in Figure 11, we can state that TRiStar is more effective at specifying TR systems.

Table 12. Results for UEffec in R2.

	Levene's Test	Kruskal-Wallis
Language	0.079	0.046
System	0.359	0.217

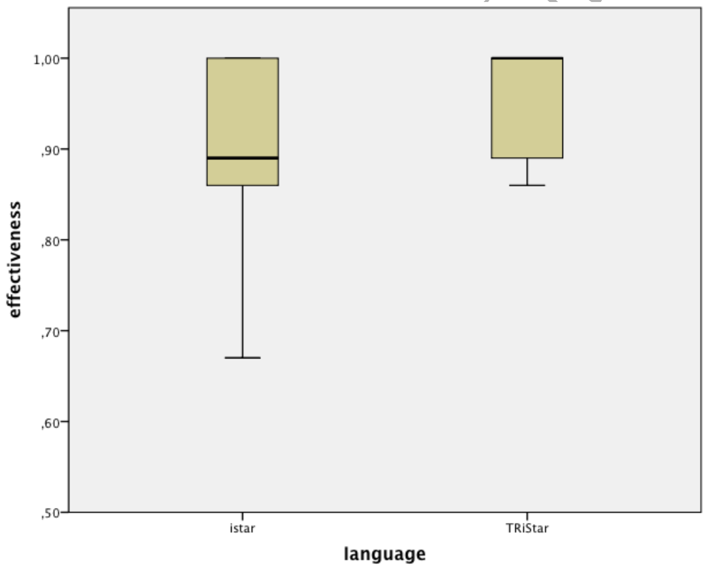


Figure 11. Distribution of UEffec in R2

Figure 12 shows the normality of the UEffec distribution. Homocedasticity was checked by Levene's test and the result is shown in Table 13. As its p-value is 0.214, the homogeneity of variances can be assumed.

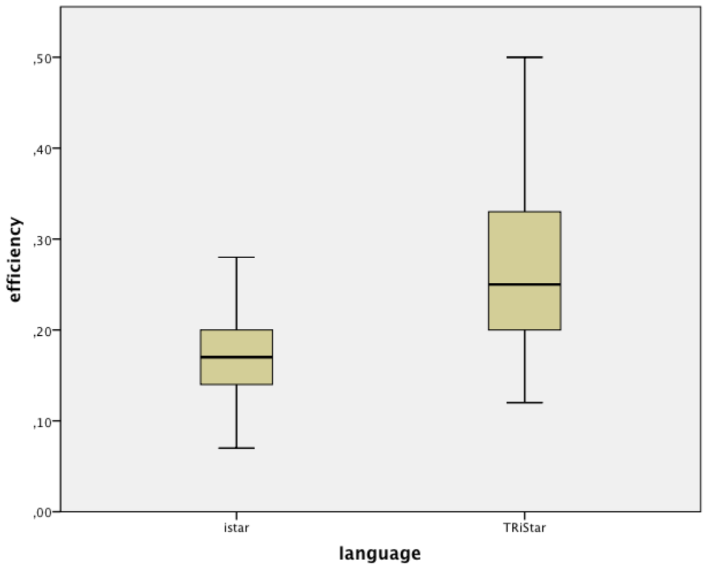


Figure 12. Normal distribution of UEffic in R2

Table 13. Levene's test for UEffic in R2

F	df1	df2	Sig.
1.616	3	22	0.214

After checking all the conditions, the two-way ANOVA test was performed. The results are shown in Table 14:

Table 14. ANOVA results for UEffic in R2.

Source	Type III Sum of squares	df	Mean Square	F	Sig.
Model	1.398[a]	4	0.349	39.589	0.000
Language	0.072	1	0.072	8.181	0.009
System	0.000	1	0.000	0.013	0.909
Language * system	0.000	1	0.000	0.017	0.899
Error	0.194	22	0.009		
Total	1.592	26			
a. R Squared = 0.878 (Adjusted R Squared = 0.856)					

Language obtained a p-value of 0.009 so we could reject $H_{UEffic0A}$. System and language*system obtained p-values well over 0.05, preventing us from rejecting $H_{UEffie0B}$ or $H_{UEffie0AB}$. From these results ir can be concluded that, as in the previous cases, language does affect efficiency when specifying TR systems but the selected system or the combination of language and system do not.

In order to determine the language which obtains the best results in terms of UEffic, the confidence interval at 95% of the mean differences was calculated and the result was [-0.17945, -0.03132]. As all the values in the interval were less than 0, we could assume that TRiStar is more efficient at specifying TR systems.

5.5 Meta-analysis

After analyzing the isolated results of every experiment in the family, we performed a global analysis of all the experiments. First, we performed a similar study to those performed for every isolated experiment but using all the data from the original experiment and the two replications. This meant performing a two-way ANOVA test both for UEffec and UEffic, keeping the null hypotheses given in Table 2.

The sample size (31 + 25 + 13 = 69) was big enough to satisfy the normality assumption. A similar Levene’s test to those described in the previous section (language + system + language * system) was applied to the data to prove homocedasticity. Table 15 shows the results of the test for UEffec and Table 16 for UEffic.

Table 15. Levene’s test for global UEffec

F	df1	df2	Sig.
0.088	3	134	0.966

Table 16. Levene’s test for global UEffic

F	df1	df2	Sig.
0.838	3	134	0.475

In both cases homogeneity of variances could be assumed, as the calculated p-values were well over 0.05.

After checking the assumptions, a two-way ANOVA test for all the samples used was performed. Table 17 summarizes the results for UEffec and Table 18 for UEffic:

Table 17. ANOVA results for global UEffec.

Source	Type III Sum of squares	df	Mean Square	F	Sig.

Model	90.900[a]	4	22.725	991.645	0.000
Language	0.296	1	0.296	12.896	4.61×10^{-4}
Domain	0.047	1	0.047	2.044	0.155
Language * Domain	0.003	1	0.003	0.151	0.699
Error	3.071	134	0.023		
Total	93.971	138			
a. R Squared = 0.967 (Adjusted R Squared = 0.966)					

With these results $H_{UEffce0A}$ could be rejected, thanks to the calculated p-value of 4.61×10^{-4} for language. However, $H_{UEffee0B}$ and $H_{UEffee0AB}$ could not be rejected as the obtained p-values for system and language*system were much higher than 0.05. The calculated confidence interval was [-0.14303, -0.04102], which proved that there was enough statistical evidence to affirm that TRiStar is more effective than *i** when specifying the requirements of TR systems.

Table 18. ANOVA results for global UEffic.

Source	Type III Sum of squares	df	Mean Square	F	Sig.
Model	3.350[a]	4	0.838	127.574	0.000
Language	0.175	1	0.175	26.617	8.73×10^{-7}
Domain	0.004	1	0.004	0.656	0.419
Language * Domain	0.019	1	0.019	2.834	0.095
Error	0.880	134	0.007		
Total	4.230	138			
a. R Squared = 0.792 (Adjusted R Squared = 0.786)					

The results for efficiency were similar to those for effectiveness: $H_{UEffie0A}$ could be rejected but $H_{UEffie0B}$ and $H_{UEffce0AB}$ must be accepted. The p-value for language was 8.73×10^{-7} but those of systems and language*system were well over 0.05. The confidence interval for the mean differences between *i** and TRiStar was [-0.09844, -0.04359]. Therefore, taking into account the aggregate results for the family of experiments, we had enough statistical evidence to state that TRiStar is more efficient than *i** when specifying requirements for TR systems.

We used BioStat's Comprehensive Meta-Analysis [34] for the meta-analysis. We first obtained the Global Effect Size of the family of experiments and then used it to decide the specific meta-analysis method to use. The Global Effect Sizes for UEffec and UEffic are shown in Table 19 and Table 20, respectively.

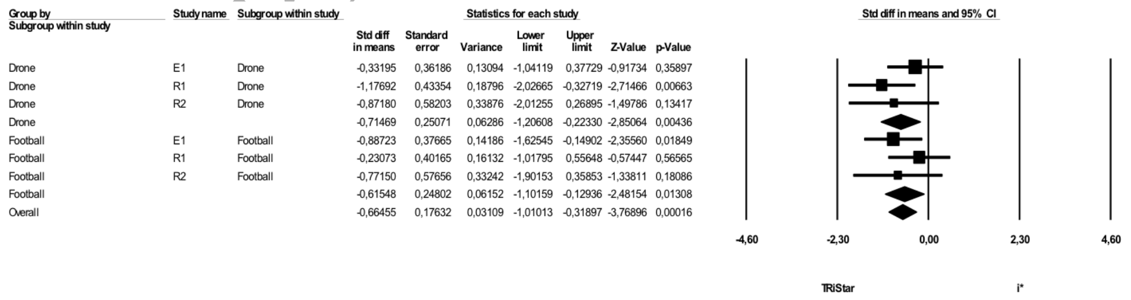
Table 19. Global Effect Size for UEffec.

		i*			TRiStar					
Study	**System**	**Mean**	**SD**	**N**	**Mean**	**SD**	**N**	**Hedges' g**	**Std. Err.**	**Effect Size**
E1	Drone	0.7653	0.10895	15	0.8075	0.14201	16	-0.3232	0.3524	Small
E1	Football	0.7575	0.13685	16	0.8767	0.13162	15	-0.8640	0.3668	Medium
R1	Drone	0.64	0.16657	12	0.8392	0.17168	13	-1.1381	0.4192	Large
R1	Football	0.7577	0.15996	13	0.7967	0.17839	12	-0.2231	0.3884	Small
R2	Drone	0.8586	0.1224	7	0.945	0.06025	6	-0.8109	0.5414	Medium
R2	Football	0.93	0.07668	6	0.98	0.05292	7	-0.7176	0.5363	Medium
Global Effect Size								-0.6411	0.1697	Medium

Table 20. Global Effect Size for UEffic.

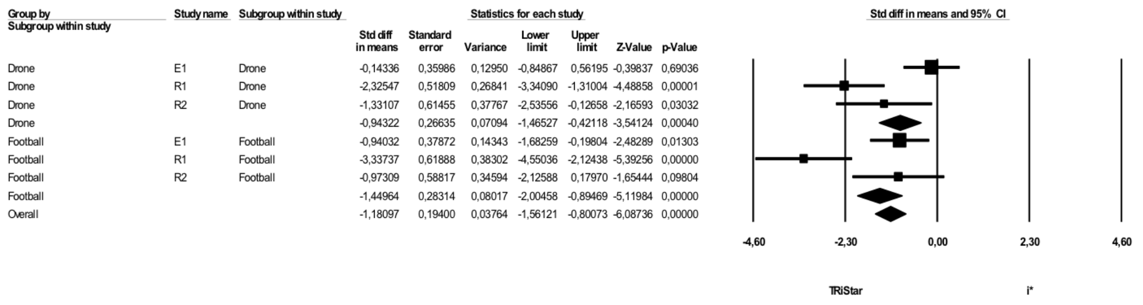
		i*			TRiStar					
Study	**System**	**Mean**	**SD**	**N**	**Mean**	**SD**	**N**	**Hedges' g**	**Std. Err.**	**Effect Size**
E1	Drone	0.13	0.08341	15	0.1413	0.07429	16	-0.1396	0.3504	Small
E1	Football	0.115	0.07607	16	0.2007	0.10491	15	-0.9157	0.3688	Medium
R1	Drone	0.0817	0.01642	12	0.1523	0.03898	13	-2.2488	0.5010	Large
R1	Football	0.0738	0.02256	13	0.1675	0.03306	12	-3.2273	0.5984	Large
R2	Drone	0.1729	0.06473	7	0.2833	0.10053	6	-1.2382	0.5716	Large
R2	Football	0.1733	0.06055	6	0.2743	0.12921	7	-0.9052	0.5471	Medium
Global Effect Size								-1.1389	0.1867	Large

With these values and following Dieste's directions [35] Weighted Mean Difference (WMD) method was chosen, as it gets the best score in reliability and statistical power for both UEffec and UEffic. FiguresFigure 13 and Figure 14 summarize the WMD results for both variables.



Group by Subgroup within study	Study name	Subgroup within study	Std diff in means	Standard error	Variance	Lower limit	Upper limit	Z-Value	p-Value
Drone	E1	Drone	-0,33195	0,36186	0,13094	-1,04119	0,37729	-0,91734	0,35897
Drone	R1	Drone	-1,17692	0,43354	0,18796	-2,02665	-0,32719	-2,71466	0,00663
Drone	R2	Drone	-0,87180	0,58203	0,33876	-2,01255	0,26895	-1,49786	0,13417
Drone			-0,71469	0,25071	0,06286	-1,20608	-0,22330	-2,85064	0,00436
Football	E1	Football	-0,88723	0,37665	0,14186	-1,62545	-0,14902	-2,35560	0,01849
Football	R1	Football	-0,23073	0,40165	0,16132	-1,01795	0,55648	-0,57447	0,56565
Football	R2	Football	-0,77150	0,57656	0,33242	-1,90153	0,35853	-1,33811	0,18086
Football			-0,61548	0,24802	0,06152	-1,10159	-0,12936	-2,48154	0,01308
Overall			-0,66455	0,17632	0,03109	-1,01013	-0,31897	-3,76896	0,00016

Figure 13. UEffec WMD meta-analysis



Group by Subgroup within study	Study name	Subgroup within study	Std diff in means	Standard error	Variance	Lower limit	Upper limit	Z-Value	p-Value
Drone	E1	Drone	-0,14336	0,35986	0,12950	-0,84867	0,56195	-0,39837	0,69036
Drone	R1	Drone	-2,32547	0,51809	0,26841	-3,34090	-1,31004	-4,48858	0,00001
Drone	R2	Drone	-1,33107	0,61455	0,37767	-2,53556	-0,12658	-2,16593	0,03032
Drone			-0,94322	0,26635	0,07094	-1,46527	-0,42118	-3,54124	0,00040
Football	E1	Football	-0,94032	0,37872	0,14343	-1,68259	-0,19804	-2,48289	0,01303
Football	R1	Football	-3,33737	0,61888	0,38302	-4,55036	-2,12438	-5,39256	0,00000
Football	R2	Football	-0,97309	0,58817	0,34594	-2,12588	0,17970	-1,65444	0,09804
Football			-1,44964	0,28314	0,08017	-2,00458	-0,89469	-5,11984	0,00000
Overall			-1,18097	0,19400	0,03764	-1,56121	-0,80073	-6,08736	0,00000

Figure 14. UEffic WMD meta-analysis

Calculated p-values (0.00016 for UEffec and $< 1 \times 10^{-5}$ for UEffic) allow us to reject the null hypothesis and say that both effectiveness and efficiency of TRiStar and *i** are different. In addition, the cell in the "Overall" row and "Std. diff. in means" column of both tables show the WMD values for effectiveness (-0.66455) and efficiency (-1.18097). As both values are less than 0 we can state that TRiStar provides better effectiveness and efficiency when specifying TR systems requirements.

5.6 Observational Findings

This section deals with the conclusions extracted from the observations made during the experiments. Most questions asked by the participants were related to the representation of priority in *i**. They could all remember how priority was represented in TRiStar but some had forgotten how thus was done in *i**, although both techniques had been explained at the same time. This suggests that the participants found the prioritized decomposition links in TRiStar more intuitive and when they saw an *i** diagram with no prioritized decomposition links they could not figure out a way of representing priority without them.

The results obtained in the second replication, with a sample of experienced software developers, were better for effectiveness and efficiency than those obtained from the students. However, the relationship between both languages is similar: effectiveness and efficiency are better in TRiStar. The experience of the software developers probably helped them to learn new notations. In addition, SAES developers are used to dealing with much more complex problems than those given in the experiment. Most of them stated that they preferred using graphical notations instead of directly reading TR program rules.

TRiStar obtained better results in effectiveness but the efficiency results were much better than those for *i**. This suggests that although *i** is still an appropriate language for representing TR systems, TRiStar does the job better and faster.

The question in the questionnaire which obtained most incorrect answers in every experiment was one included in the drone example. In fact, none of the subjects who specified the drone with *i** answered this question correctly. Those who specified the drone with TRiStar had better results, but there were still a lot of wrong answers. This question was related to representing rules whose condition is always true. These results suggest that dependent decompostion links help to link conditions to rules, even though the representation of unconditioned rules in TRiStar could be improved.

6 Threats to the Validity of the Family of Experiments

In order to reduce research and publication bias, as recommended in [36], the raw experimental data can be consulted in http://xurl.es/RawData. This section deals with some issues that could have threatened the validity of the experiment, in line with the recommendations of Wohlin et al [37].

6.1 Validity of the conclusions

The statistical indicators obtained from both the individual experiments and the meta-analysis are well above a 95% confidence level, which allows us to reject the initial null hypotheses.

6.2 Internal validity

As detailed in the previous section, we showed that all the results of the individual experiments satisfied the requirements of the selected statistical methods (ANOVA and Kruskal-Wallis tests). The questionnaires were reviewed by several experts in the development of TR systems and the use of *i** to minimize the risk of incorrect questions.

None of the experiments lasted more than one hour, including the initial briefing by the instructor, to avoid the subjects becoming fatigued. Besides, the students that participated in the experiments were given an extra half point towards their final exam, while in the second replication, the professionalism of the subjects ensured their motivation.

6.3 Construct validity

The method employed to obtain the data from the experiments was a questionnaire similar to those used in other studies, e.g. [6] and [28], which reduced the threats to the construct validity. Understandability efficiency and effectiveness were also measured in a similar way to the above-cited studies: efficiency was obtained by dividing the number of correct answers by the total number of answers, while effectiveness was calculated as efficiency divided by the time in minutes taken by each participant to complete the questionnaire, as described in ISO/IEC 25000:2014.

6.4 External validity

According to [38], the differences between final-year students and software professionals when performing relatively small judgement tasks are minor. Since the questions in the questionnaire for both students and software professionals were not excessively complex, the mixture of students and professionals in the experiment did not involve a threat to it. This view is supported by the the good results obtained for the efficiency parameter, as well as the few questions raised by the participants on the experiments.

Regarding the nature of the proposed problems, we can affirm that the examples employed in the experiments were realistic, since both are part of already existing systems.

7 Conclusions and Further Work

In [6] we showed that the understandability of *i** notation was better than that of KAOS for specifying the requirements of teleo-reactive systems. From these results we developed TRiStar, an extension designed to overcome some shortcomings we identified in *i**, which is briefly introduced in [10] and fully described in the present paper. With the aim of validating the proposal, we conducted a family of experiments to compare the efficiency and effectiveness of the understandability of *i** versus TRiStar for specifying the requirements of teleo-reactive systems.

Subjectively, the vast majority of the participants stated that they found the TRiStar specifications more understandable than those of *i**. Regarding efficiency and effectiveness, the statistical results are conclusive; on one hand, the results of the analysis of the original experiment and the two replicas, and on the other, the results of the meta-analysis of the aggregate data considered as a single experiment, provide enough statistical certainty to reach the following conclusion: both the efficiency and effectiveness of TRiStar are higher than that of *i** diagrams for specifying the requirements of teleo-reactive systems.

In future research work we plan to extend TRiStar in order to cope with the new extensions proposed by Prof. Keith Clark in TeleoR [3]. We would also like to complete the requirements specification process for teleo-reactive systems by defining a method of guiding the process, starting from natural language specifications.

In the sequel to this research, we intend to make a study of the advantages of TRiStar for the requirements specification of TR systems as compared with a direct approach to TR programs. Starting from a textual description of a reactive system, the results obtained with TRiStar will be compared to those obtained by writing the TR programs directly. Among other objectives, this study will focus on detecting coupling problems among agents, detecting cohesion problems among goals, implementation effectiveness and early error detection.

Lastly, we would also like to develop a graphical tool to help developers depict TRiStar diagrams. This tool would include functionalities such as subgoal expand/collapse, which would be helpful in improving the scalability of the models. This tool will also allow the generation of the TR program which corresponds to the specified diagram.

Acknowledgements

This work was partially supported by the insPIre (ref. TIN2012-34003), cDrone (ref. TIN2013-45920-R) and ViSelTR (ref. TIN2012-39279) projects of the Spanish Government. Diego Alonso wishes to thank the Spanish *Ministerio de Educación, Cultura y Deporte, Subprograma Estatal de Movilidad, Plan Estatal de Investigación Científica y Técnica y de Innovación 2013-2016* for grant CAS14/00238. We specially acknowledge the support provided by the SAES Company and its employees, as well as by the students of the Politechnic University of Cartagena who participated in the study. Lastly, we wish to thank Esther Corbalán for her support and valuable contribution in carrying out the statistical analysis.

References

[1] Nilsson, N. J. (1993). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, *1*(1), 139–158. Retrieved from http://dl.acm.org/citation.cfm?id=1618595.1618602

[2] Morales, J., Sánchez, P., Alonso, D. (2012). A systematic literature review of the Teleo-Reactive paradigm. *Artificial Intelligence Review*, Springer Netherlands, 2012, (pp. 1-20).

[3] Clark, K. L., Robinson, P. J. (to appear 2015). Programming Robotic Agents: A TR Multi-Tasking Approach. Springer.

[4] Rajan, K., Py, F., McGann, C. (2010). Adaptive control of AUVs using onboard planning and execution. *Sea Technology*, April 2010, (pp. 51-55).

[5] Gubisch, G., Steinbauer, G., Weiglhofer, M., Wotawa, F. (2008). A Teleo-Reactive Architecture for Fast, Reactive and Robust Control of Mobile Robots. *IEA/AIE '08 Proceedings of the 21st international conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence*

[6] Morales, J. M., Navarro, E., Sánchez, P., Alonso, D. 2015. *A controlled experiment to evaluate the understandability of KAOS and i* for modeling Teleo-Reactive systems*. Journal of Systems and Software 100 (pp. 1-14).

[7] Sánchez, P., Alonso, D., Morales, J. M., Navarro, P. J. (2012). From Teleo-Reactive Specifications to Architectural Components: A Model-Driven Approach. *Journal of Systems and Software 85 (11),* (pp. 2504 – 2518).

[8] Yu, E. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97)* Jan. 6-8, 1997, Washington D.C., USA. (pp. 226-235).

[9] Teruel, M. A., Navarro, E., López-Jaquero, V., Montero, F., González, P. (2011). CSRML: A Goal-Oriented Approach to Model Requirements for Collaborative Systems. *Conceptual*

Modeling – ER 2011, Lecture Notes in Computer Science Volume 6998, 2011, (pp. 33 – 46).

[10] Morales, J. M., Navarro, E., Sánchez, P., Alonso, D. 2015. *TRiStar: An i* extension for Teleo-Reactive systems requirements specifications*. Proceedings of the ACM Symposium on Applied Computing, April 13-17, 2015, Salamanca, Spain. http://dx.doi.org/10.1145/2695664.2695703

[11] "underˈstandable, adj." OED Online. Oxford University Press, March 2015. Web. 20 March 2015.

[12] Genero, M. et al. 2008. Defining and validating metrics for assessing the understandability of entity–relationship diagrams. Data & Knowledge Engineering Volume 64, Issue 3, March 2008, (pp. 534–557)

[13] Sommerville, I., Sawyer, P., Viller, S. (1998). Viewpoints for requirements elicitation: a practical approach, *IEEE International Conference on Requirements Engineering (ICRE'98),* Colorado Springs, Colorado, 1998.

[14] Chernak, Y. (2009). Building Foundation for Structured Requirements. Aspect-oriented Requirements Engineering Explained- Part 1, in: Requirements Networking Group (RQNG), 2009.

[15] Lamsweerde, A. (2009). Requirements Engineering: from goals to UML models to software specifications. John Wiley & Sons Ltd, England

[16] Yu, E., & Mylopoulos, J. (1998, June). Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality* (Vol. 15).

[17] Tsalgatidou, A., Karakostas, V., & Loucopoulos, P. (1990, January). Rule-based requirements specification and validation. *In Advanced Information Systems Engineering* (pp. 251-263). Springer Berlin Heidelberg.

[18] Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. *5th IEEE International Symposium on Requirements Engineering (RE'01)* (pp. 249–262). Washington DC, USA: IEEE Comput. Soc. doi:10.1109/ISRE.2001.948567

[19] Ayala, C. P., Cares, C., Carvallo, J. P., Grau, G., Haya, M., Salazar, G., ... & Quer, C. (2005). A Comparative Analysis of i*-Based Agent-Oriented Modeling Languages. *In SEKE* (Vol. 5, pp. 43-50).

[20] Amyot, D., & Mussbacher, G. (2003). URN: Towards a new standard for the visual description of requirements. *In Telecommunications and beyond: The Broader Applicability of SDL and MSC* (pp. 21-37). Springer Berlin Heidelberg.

[21] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3), 203-236.

[22] Lockerbie, J., & Maiden, N. A. (2008). REDEPEND: Tool Support for i* Modelling in Large-scale Industrial Projects. *In CAiSE Forum* (Vol. 344, pp. 69-72).

[23] Fuxman, A., Liu, L., Mylopoulos, J., Pistore, M., Roveri, M., & Traverso, P. (2004). Specifying and analyzing early requirements in Tropos. *Requirements Engineering*, 9(2), 132-150.

[24] A. Sutcliffe, S. Minocha. "Linking Business Modelling to SocioTechnical System Design". *Proceedings of Advanced Information Systems Engineering, 11th International Conference CAiSE'99*, Heidelberg, Germany, June 14-18, 1999, pp. 73-87.

[25] Jamison, W. and Teng, J.T.C. (1993): Effects of Graphical Versus Textual Representation of Database Structure on Query Performance. *Journal of Database Management* 4 (1): (pp. 16–23).

[26] Lee, H. and Choi, B.G. (1998): A Comparative Study of Conceptual Data Modeling Techniques. *Journal of Database Management* 9(2): (pp. 26-35).

[27] Bajaj, A. (2004): The effect of the number of concepts on the readability of schemas: an empirical study with data models. Requirements Engineering 9: (pp. 261-270).

[28] Teruel, M. A., Navarro, E., López-Jaquero, V., Montero, F., Jaen, J., & González, P. (2012). Analyzing the understandability of Requirements Engineering languages for CSCW systems: A family of experiments. *Information and Software Technology*, 54(11), 1215–1228. doi:10.1016/j.infsof.2012.06.001

[29] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8), (pp. 721–734). doi:10.1109/TSE.2002.1027796.

[30] Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric Approach. In *Encyclopedia of Software Engineering* (Vol. 2, pp. 528–532). Wiley. Retrieved from http://wwwagse-old.informatik.uni-kl.de/pubs/repository/basili94b/encyclo.gqm.pdf

[31] Winer, B. J., Brown, D. R., Michels, K. M. (1991). Statistical Principles in Experimental Design (3rd ed.) (p. 928). *McGraw-Hill Humanities/Social Sciences/Languages*.

[32] Grinstead, C.M., Snell, J.L. Introduction to Probability, American Mathematical Society, 2006.

[33] Levene, Howard (1960). Ingram Olkin, Harold Hotelling, et alia, ed. *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Stanford University Press. (pp. 278–292).

[34] Biostat Inc, Comprehensive Meta-Analysis, 2006. <http://www.meta-analysis.com> (accessed June 2015).

[35] O. Dieste, E. Fernández, R. García Martínez, N. Juristo, Comparative analysis of meta-analysis methods: when to use which? *15th International Conference on Evaluation & Assessment in Software Engineering* (EASE'11), IET, Durham, UK, 2011, (pp. 36–45).

[36] M. Jørgensen et al., Incorrect results in software engineering experiments: How to improve research practices, *The Journal of Systems and Software* (2015), http://dx.doi.org/10.1016/j.jss.2015.03.065

[37] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. (2000) Experimentation in Software Engineering: An Introduction. *first ed., Kluwer Academic Publishers*, Norwell, USA, 2000.

[38] Höst, M., Regnell, B., Wohlin, C. Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment, *Empirical Software Engineering* 5 (3) (2000) (pp. 201–214).

José Miguel Morales is an Assistant Professor and a Ph.D. student in computer sci- ence at the Universidad Politécnica de Cartagena and a member of the university's DSIE (Division of Systems and Electronic Engineering) research group. His research interests include real time systems, and specification and design of teleo-reactive systems. Morales has a master's in information technology engineering from the Universidad de Murcia.

Elena Navarro is an Associate Professor of Computer Science at the University of Castilla-La Mancha (Spain). Prior to this position, she worked as a researcher at the Informatics Laboratory of the Agricultural University of Athens (Greece) and as a staff member of the Regional Government of Murcia, at the Instituto Murciano de Investigación y Desarrollo Agroalimentario. She got her bachelor degree and Ph.D. at the University of Castilla-La Mancha, and her master degree at the University of Murcia (Spain). She is currently an active collaborator of the LoUISE group of the University of Castilla-La Mancha. Her current research interests are Requirements Engineering, Software Architecture, Model-Driven Development, and Architectural Knowledge.

Pedro Sánchez received his Ph.D. degree in computer science from the Technical University of Valencia, Spain, in 2000. Since 1996, he has participated in different projects focused on software engineering and conceptual modeling applied to the development of reactive systems. In 2000, he joined the Systems and Electronic Engineering Division (DSIE) at the Technical University of Cartagena. He is currently an Associate Professor at the Technical University of Cartagena in the field of computer science. His current research interests include software engineering for implementing teleo-reactive systems.

Diego Alonso is currently an Associate Professor of Computer Science at the Universidad Politécnica de Cartagena (Spain) and a member of the DSIE (Division of Systems and Electronic Engineering) research group. He received a M. Sc. Degree in Industrial Engineering from the Universidad Politécnica de Valencia (Spain), and a Ph.D. with "Doctor Europaeus" Mention form the Universidad Politécnica de Cartagena. His research interests focus on the application of the model-driven engineering approach to the development of component-based reactive systems with real-time constraints, mainly in the field of robotics.

Appendix 1: Experimental Material - An Example of an Understanding Task (Test for group 2)

Gender (Male/Female)

Age

Qualification

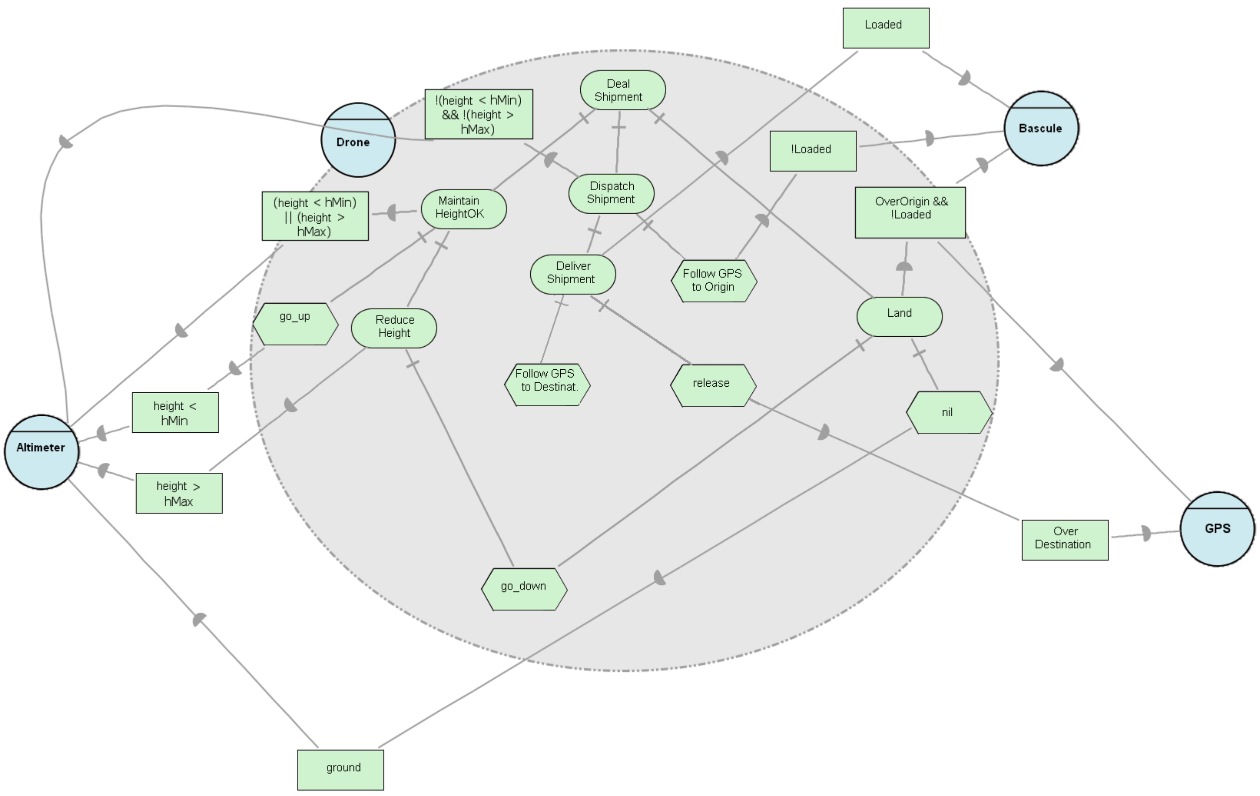
Average score

Have you had any previous experience of working with goal-oriented requirements engineering?

Have you had any previous experience of working with any other requirements engineering technique?

Have you had any previous experience of working with teleo-reactive systems?

[FILL IN AT THE END] In your opinion, which notation has better understandability?


Loaded
Deal Shipment
!(height < hMin) && !(height > hMax)
Drone
Bascule
!Loaded
Dispatch Shipment
(height < hMin) || (height > hMax)
Maintain HeightOK
OverOrigin && !Loaded
Deliver Shipment
Follow GPS to Origin
go_up
Reduce Height
Land
Follow GPS to Destinat.
release
nil
height < hMin
Altimeter
height > hMax
Over Destination
GPS
go_down
ground

STARTING TIME:

Fill in the blanks so that the obtained TR program agrees with the one that would be obtained from the previous specification.

1.- DealShipment:

__________ → Land

__________ → DispatchShipment

__________ → MaintainHeightOK

2.- Land:

Ground → ______

___________ → go_down

3.- Choose the correct (a) or (b):

(a).- DispatchShipment:

Loaded → DeliverShipment

NOT(Loaded) → followGPSToOrigin

(b).- DispatchShipment:

NOT(Loaded) → followGPSToOrigin

Loaded → DeliverShipment

4.- DeliverShipment:

_______ → release

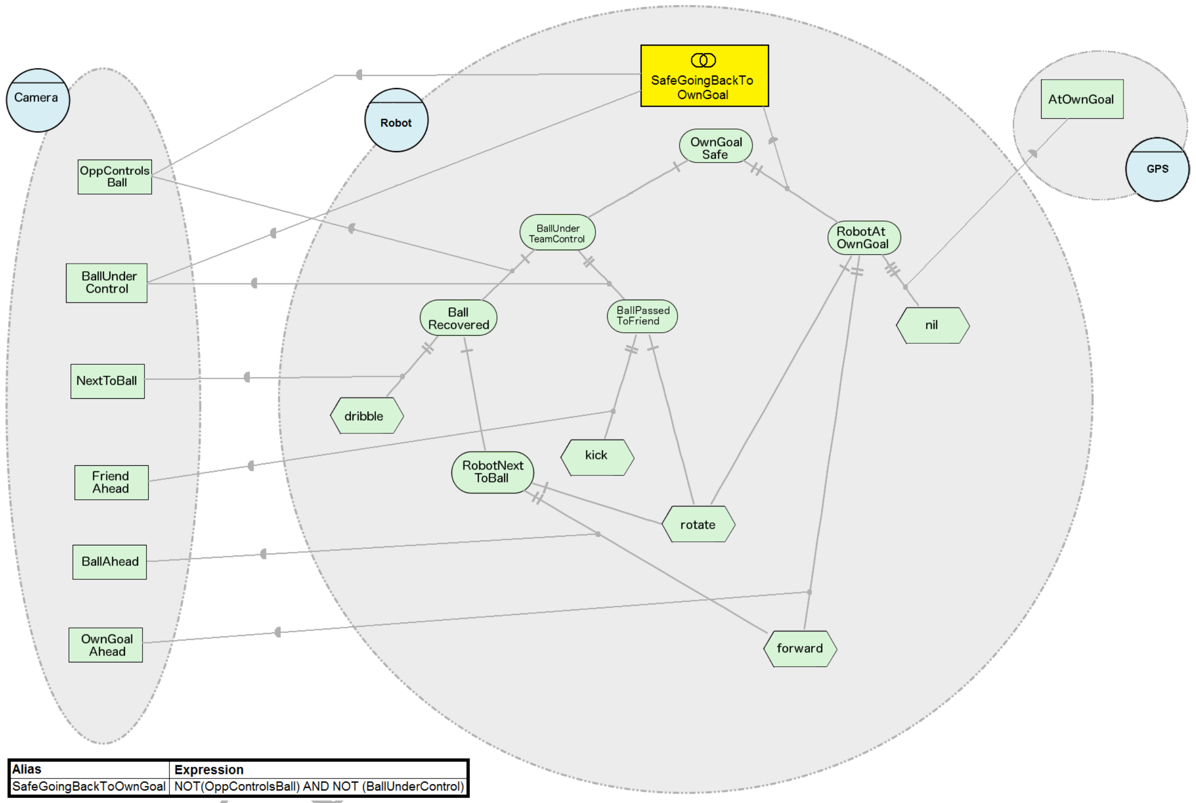
true → followGPSToDestination

5.- MaintainHeightOK:

height > hMax -> _______

________________ -> go_up

ENDING TIME:



Alias	Expression
SafeGoingBackToOwnGoal	NOT(OppControlsBall) AND NOT (BallUnderControl)

STARTING TIME:

Fill in the blanks so that the obtained TR program agrees with the one that would be obtained from the previous specification.

1.- RobotNextToBall:

BallAhead → ___________

True → rotate

2.- OwnGoalSafe:

_______________________________________ → RobotAtOwnGoal

True → _________________________

3.- RobotAtOwnGoal:

____________________ → nil

OwnGoalAhead → __________________

True → rotate

4.- BallPassedToFriend:

FriendAhead → kick

_________________ → rotate

5.- Choose the correct (a) or (b):

(a).- BallUnderTeamControl:

BallUnderControl → BallPassedToFriend

OppControlsBall → BallRecovered

(b).- BallUnderTeamControl:

OppControlsBall → BallRecovered

BallUnderControl → BallPassedToFriend

ENDING TIME:

Graphical abstract