identify the i^* concepts that are easy for the students to understand and the guidelines that they can easily apply to produce effective models. In addition, potential ways to improve the teaching of i^* are suggested to be used by others.

The paper is structured as follows. Section 2 introduces the Amazon drone delivery coursework. Section 3 introduces the guidelines and conventions we used for teaching i^* modelling. Section 4 reports on the results of the use of these guidelines by students to complete the coursework. Finally, Section 5 summarises our findings and concludes the paper.

2 Coursework : Amazon Drone Delivery

The students were given a use case specification describing how a shopper may use the Amazon Prime Air service to receive a package delivered by a drone. They were also provided with more general background information on the new delivery system to provide them with the necessary context. They were then asked to produce an i^* SD model for a set of given actors, and to produce an SR model for the shopper actor. To help the students with an example, it was specified that the model should include the goal dependency the shopper depends on the drone to satisfy the goal package unlocked.

3 Guidelines and Conventions for Producing i^* Models

We taught the students a number of guidelines, developed by City University London [4], to help them produce i^* models more effectively. We describe the main guidelines in the following.

Guideline 1: Degree of delegation to determine the type of dependency. One of the most difficult decisions in SD modelling is to determine the type of dependency. The difficulty might be due to the confusion between the data flow and the responsibilities and expectations of the actors involved. We used a guideline based on the degree of delegation from the depender to the dependee in order to guide the choice of dependency [4]. This guideline specifies that goal and soft goal dependencies indicate greater degrees of delegation than task dependencies, and task dependencies indicate greater degrees of delegation than resource dependencies. For example, if the shopper depends on the drone to attain the goal package collected, then the shopper fully delegates to the drone the satisfaction of the goal. If the shopper depends on the drone to do the task collect package, there is less delegation: the shopper initiates the task, but depends on the drone to complete the task successfully. In other words, the shopper and drone collaborate to perform the task collect package. If the shopper depends on the drone only for getting the resource package available, then there is little delegation.

Guideline 2: Dependency directions. In order to help the students transition between SD models and SR models, we used the guideline that states: *IF Actor A is a depender in a dependency relationship in the SD model, THEN the depended-upon element is modelled in Actor A's SR mode*l [4]. For example, the task dependency guideline presented in [3], states that in a task-type dependency, the depender initiates and owns the task. This means that if Actor A is a depender

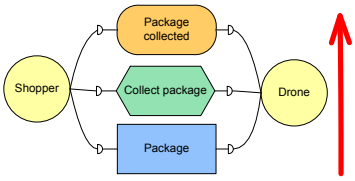

Fig. 1. Example of type of dependency

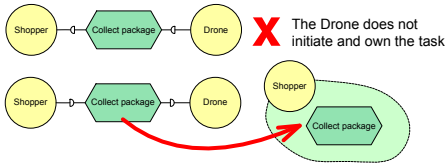

Fig. 2. Example of dependency directions

in a task relationship in the SD model, then the task element is modelled in Actor A's SR model. Figure 2 shows how the guideline makes it clear which SD elements appear in which SR model boundaries.

Guideline 3: Naming conventions. It is important to ensure the wording of the goals, soft goals, tasks and resources is consistent and understandable by other people. We used the naming conventions from [3], as shown in Table 1.

i^* element	Naming convention	Example
Goals	Past participle describing a desirable state	<desirable state>: package unlocked
Soft goals	Properties or constraints on a desirable state	<desirable state> <adjective | adverb>: order processed reliably
Tasks	Active verbs describing how something is done (bare infinitive from of verb)	<do task>: collect package
Resource	Noun describing resource	<resource>: order number

Table 1. Naming conventions for the i^* process elements

Guideline 4: Pre-defined operators for soft goals. To help the students think about how to specify soft goals, we introduced the goal operators used in the KAOS goal modelling approach [1]. van Lamsweerde identified that most goals on software-based systems are types of a small number of sets. In simple terms, the system or actor is trying to achieve, cease, maintain, prevent, or optimise something. Examples of soft goals are reliable delivery achieved and shopping convenience maximised.

Guideline 5: Task structure for SR models. There is a lack of explicit process guidance about how to start creating an SR model. Therefore, we applied the guideline from [4] that states: *IF there is a mandated solution, THEN describe the solution as a task, and ask why (goals and soft goals) and how (sub-tasks and resources).* As the solution was mandated in the coursework description, this approach was recommended to the students. Furthermore, through tutorial examples we encouraged a core task structure, based on the main 'super' task and decompositions of this task thereon. We discouraged fragmented models, and stated that whilst i^* implies no ordering, models are easier to read if a basic temporal order is applied from left to right in the diagram. A simple example is shown in Figure 3.

Guideline 6: Task decomposition to soft goal. To guide the modelling of task decompositions, we used the guideline that states: *a sub-soft goal of a task must be satisfied by the completion of the task, and is a post-condition.* Therefore,

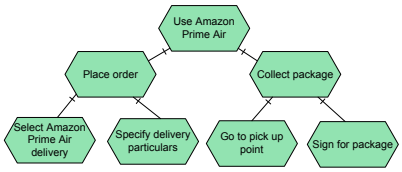

Fig. 3. Task structure with temporal ordering

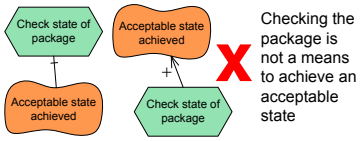

Fig. 4. Task decomposition to soft goal

we encouraged students to model desirable properties or states of a task using task decomposition links. For example, Figure 4 shows the difference between using a task decomposition link and a contributes-to soft goal link.

The need for tooling. As reported in [3], i^* modelling can be challenging without tool support. Therefore, we provided the students with the MS Visio-based REDEPEND i^* modelling tool [2]. 118 out of the 134 students used REDEPEND to produce their i^* SD and SR models. Although only indicative due to the low sample size, it is worth mentioning that of those students not using REDEPEND only 25% linked the SR model correctly, compared with 65% of those using REDEPEND. We expected the number to be higher for the REDEPEND users, as the tool checks the validity of links in real-time, but only if the macros are enabled!

4 Qualitative Evaluation

We reviewed the complete set of 134 coursework submissions and assessed the i^* SD and SR models based on a set of criteria related to the basic semantics of i^* modelling as well as the correct application of the guidelines described in Section 3, to which they have been introduced during the lectures and tutorials. 59 of the assignments are from undergraduate (UG) students and 75 from postgraduate (PG) students. Figure 5 summarises the criteria and results, which we describe in the following.

Correct SD elements. We checked if the students had understood the definitions of the goal, soft goal, task and resource elements and whether they had applied them correctly. We found that 61% of UGs and 72% of PGs made no errors specifying the 4 different element types. The most common mistake was misunderstanding the subtleties between some soft goals and goals. For example, expressing package delivered on time as a soft goal when it expresses a boolean condition akin to a goal. The next most common error was confusing tasks and goals, which seemed to be caused by the students not following the suggested wording conventions.

Correct SD links. We checked if the SD models were constructed correctly with respect to actors and linked dependencies. The outcome was encouraging, with 93% of UGs and 99% of PGs successfully constructing their SD models.

Correct SR elements. Interestingly, the students performed better at choosing the correct process elements for the SR models than for the SD models. 68% of UGs and 87% of PGs made no errors with the 4 types of process elements.

Correct SR links. We looked at the different i^* link types to check if they had been applied and connected properly. The results showed that the students did not construct SR models as successfully as they did SD models, which is

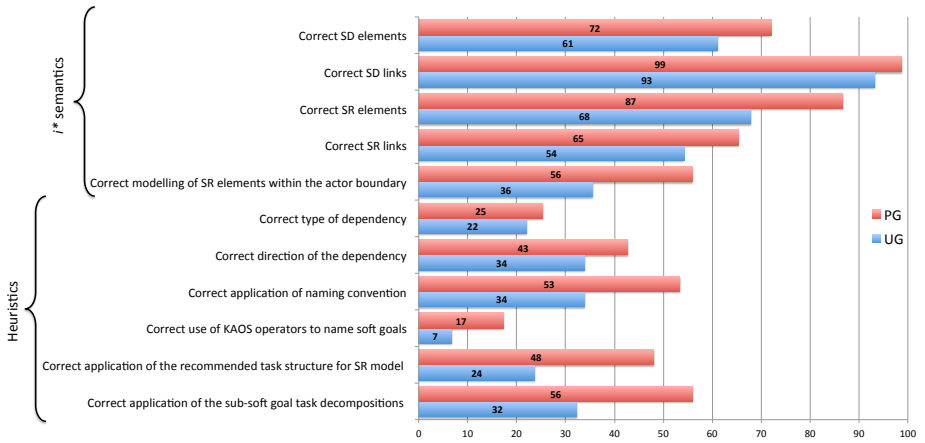

Fig. 5. Percentage of correct application of the guidelines to the coursework

not surprising due to the additional links and richer semantics. For the UGs, 54% applied the links correctly as compared with 65% for the PGs. Common mistakes were contributes-to soft goal links connected to non-soft goals, and task decomposition links used to decompose goals.

Correct modelling of SR elements within the actor boundary. We checked to see if the elements within the actor boundary specified only what the actor could accomplish themselves. We found that only 36% of UGs correctly identified which elements should be contained with the Shopper boundary, with the corresponding percentage for PGs at 56%. There was a strong tendency for the students to model certain tasks and goals owned by the drone e.g. the task unlock package.

Correct type of dependency. We reviewed the SD models to check if the correct dependency types had been used and whether duplication of dependencies was avoided. The results were disappointing, as only 22% of UGs and 25% of PGs correctly applied the delegation guideline. The most common error was duplicating dependencies, usually by specifying resource dependencies as well as the higher level goals and tasks to which they relate.

Correct direction of the dependency. As with delegation, this guideline was not applied successfully the majority of the time. The guideline was applied correctly by 34% of UGs and 43% of PGs. It was common for students to express that the depender actor depends on the dependee to undertake a task. For example, Drone depends on Shopper to collect package, rather than Shopper depends on Drone to collect package. In reality, Shopper collects the package, whilst Drone provides the package.

Correct application of naming convention. Looking at both the SD and SR models, we found that the naming conventions were only applied correctly by 34% of UGs and 53% of PGs. Many students failed to use the past participle when expressing goals and soft goals.

Correct use of KAOS operators to name soft goals. This was the least successfully applied guideline, with only 7% of UGs and 17% of PGs using the operators on all of the soft goals in their i^* models. It was clear that whilst these formal operators may help specify certain types of soft goals, they are not appropriate for all.

Correct application of the recommended task structure for SR model. This was the most subjective of the criteria, but nonetheless we made a boolean assessment of the criterion based on the SR model structure we were trying to mandate (see Figure 3). We found that 24% of UGs correctly followed our guidelines, with a more encouraging number of PGs at 48%.
Correct application of the sub-soft goal task decompositions. We checked whether the 'qualities of the task' had been modelled using task decomposition links as we had advised. For UGs, 32% correctly applied the guideline, whilst this percentage was higher for PGs at 56%.

5 Conclusion

In this paper we discussed the effectiveness of existing guidelines for teaching i^* modelling. While the main elements and links of the models appeared easy to grasp, applying most of the guidelines had only qualified success.

For teaching the basic concepts and conventions of i^*, we found the most effective method was using trivial real-world examples, e.g., organising a Halloween party. Therefore, we would recommend using a non-software engineering example to convey the basics of i^*, and then move onto more complex examples.

In terms of the guidelines, given that there is a heavy payload learning the i^* elements and semantics, introducing guidance on modelling too soon appears to overwhelm many of the students. However, the students who grasped the semantics quickly managed to apply the guideline correctly and create some very impressive i^* models. Overall, this was more apparent with the PG students. Whilst identifying process elements for the SD model through the delegation principles, it was evident through our teaching that the students may have benefitted from making a draft SR model at the same time. Rather than follow the sequential development of SD and SR models, we believe that an iterative process where both models are developed concurrently can be more efficient.

We believe that some of the students found the suggested naming conventions counterintuitive, e.g., using the past participle for goals while specifying a system to be. Therefore, teachers should make it clear that goals are about desirable states of the system to be. Also, we still feel that these conventions are helpful and worth teaching, as many of the students use the wording to understand the differences between tasks and goals, and goals and soft goals. Finally, visual tools can provide assistance and improve substantially the models produced by the students.

References

1. van Lamsweerde, A.: Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley (2009)
2. Lockerbie, J., Maiden, N.A.M.: REDEPEND: tool support for i* modelling in large-scale industrial projects. In: Proc. of CAiSE. pp. 69–72 (2008)
3. Maiden, N., Jones, S., Ncube, C., Lockerbie, J.: Using i^* in requirements projects: some experiences and lessons. Social modeling for requirements engineering. MIT Press, Cambridge pp. 155–185 (2011)
4. Maiden, N.A.M.: IN3015 and INM311 Requirements Engineering Supermodule (PRD1 2013/14). City University London (2013/14)