T. Hirashima et al. (Eds.) (2011). Proceedings of the 19th International Conference on Computers in Education. Chiang Mai, Thailand: Asia-Pacific Society for Computers in Education

The level up procedure: How to measure learning gains without pre- and post-testing

Kurt VANLEHN[a], Winslow BURLESON[a], Helen CHAVEZ ECHEAGARY[a], Robert CHRISTOPHERSON[a], Javier GONZALEZ SANCHEZ[a], Yoalli HIDALGO PONTET[a], Katarzyna MULDNER[a] & Lishan ZHANG[a]
[a]*CIDSE, Arizona State University, USA*

Abstract: The level up procedure is a method for evaluating the learning gains of educational software, and tutoring systems in particular, that includes some form of embedded assessment. The instruction is arranged in levels that take only a few minutes to master, and students level up when the software indicates they have achieved mastery. This paper reports some methodological lessons learned from applying this procedure in studies of a tutoring system that taught high school students how to model dynamic systems.

Keywords: Experimental methodology, tutoring systems, embedded assessment

1. Methodological problem

A common methodological problem in AI and Education research is to determine the learning gains caused by one or more types of instruction. There are three common study designs for measuring learning gains.

The *fixed tasks* design has all students work on the same set of training tasks, then take a post-test. They may also take a pre-test before the training. A student's learning is measured as a function of their post-test score, perhaps adjusted by a pre-test score or other measure of prior competence. Typically, students vary in how long it takes them to complete the training tasks. This can be considered a secondary measure that is related to learning rate. Other behavior during training is sometimes measured, such as errors, hints or latencies. However, the fixed tasks design is a difficult to use in a setting where all the students are present for the same period of time. For instance, if the study runs during a single class period, then some students may not finish by the end of the period, and others may finish it long before the end of the period.

In order to fit the study into a fixed period of time, one may use the *fixed time* design, which has all students work on training tasks for the same period of time, then take a timed post-test. Students may also take a pre-test before training. As with the fixed tasks design, the primary measure of learning is the post-test score, perhaps adjusted by a pre-test score. Here, the secondary measure is number of tasks completed, which is again related to learning rate. Again, other measures of behavior may be collected as well.

The third design, called *mastery learning*, requires students to keep doing training tasks until they can pass a mastery test [1]. A post-test is typically not used because it would be redundant with the mastery test. The main measures of learning are the number of tasks and the amount of time required to reach mastery.

There is a pleasing symmetry about these three designs. Given the three variables of training tasks, training time and competence acquired, each design controls one variable and measures the other two.

However, all three designs require testing, which raises many issues. For instance, the tests may not be aligned with the training, or students may be fatigued or disengaged by the time they reach the test. Moreover, the time required for testing subtracts from the time available for training. The methodological problem addressed here is to formulate a design that does not require testing.

2. Proposed solution: The level up procedure

The inspiration for the proposed solution come from computer games, which often have many levels of difficulty, and players "level up" only when they have demonstrated competence in their current level. Although this design element is included for many reasons (e.g., keeping players in the optimal game play corridor), it also serves as an assessment of player competence.

This procedure can be used to evaluate non-game instruction if (1) the instruction can be divided up into levels where the training inside a level is homogenous, and (2) there is some kind of assessment embedded in the instruction so that a student can level up without having to take a test. We call this the level up procedure.

For example, if the instruction is a tutoring system with student modeling capabilities [2], then its assessment of the student's mastery can be used to determine when to level up. Even when the tutoring system does not have student modeling capabilities, it usually does give feedback and hints on steps or answers, so one simple policy is to level up whenever the student can complete at least one problem correctly without using any hints. If guessing can sometimes be successful, then it would be wise to require that more than one problem be answered correctly without hints. In short, the level up procedure can be easily used with most tutoring systems including ones without a sophisticated embedded assessment.

If progress through levels is gated by the student's mastery, then one can use two designs for evaluation studies. One study design is to have students work for a fixed period of time and use their final level as a measure of learning. Another study design is to have students finish all the levels, and use the time required as a measure of learning.

In the abstract, the level up procedure is hardly innovative, as it is a simple combination of mastery learning [3] with embedded assessments. Indeed, it is used by the tutoring systems of Carnegie Learning, ALEKS, Pearson and others. However, their levels correspond to instructional modules or units, and leveling up only occurs after hours of problem solving on a level.

We wondered if the same basic procedure could be used in the context of a single-session training experiment, where students work through many levels in a single 2-hour period. As far as we have been able to determine, this design has not appeared before in the learning sciences literature, so we feel a little less guilty about coining "level up" as a new name for it. The rest of this paper describes two studies that used the level up procedure, and what we learned about it from that experience.

3. The project and the task domain

The level up procedure was used to study students' behavior while using a tutoring system. We will first describe the task domain, then the tutoring system, and finally the studies. The results of the studies will be presented elsewhere. This paper concerns methodology.

The students' task was like concept mapping [4]. They read a text describing a system and then drew a node-link diagram that modeled it. The nodes represented variables or factors, and the links represent direct influence or causality. Unlike a conventional concept

map, our nodes represented quantities and had algebraic calculations inside them. For instance, Figure 1 shows a model of a barrel with a hole near its bottom so that the amount of water emptied from the barrel per minute is 5% of the amount of water left in the barrel. The math inside the node named "emptying" is "emptying = barrel * drain rate." Such node-link diagrams are called stock and flow diagrams, and they are a standard notation for system dynamics models.

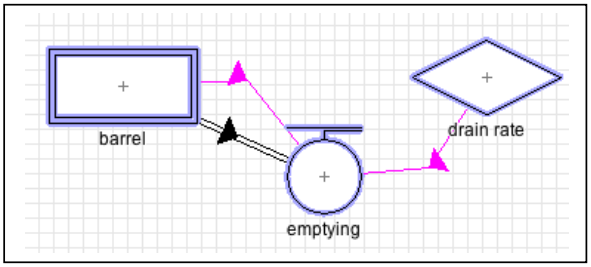

Figure 1: Model of proportional outflow

Our instruction comprised 11 levels, where each level introduced just one new concept. For instance, Figure 1 comes from level 8, where the concept of a proportional outflow is introduced. Each level consisted of a short reading, an example of a system and its model, and a sequence of three modeling problems to solve.

The tutoring system was answer-based [5]. That is, it only gave feedback when student submitted a model (the "answer"). Students had to submit a correct model in order to go on to the next problem. Instruction was available from a constantly present, level-specific reading, and from three hints per problem. The strongest hint (called the bottom-out hint) revealed all the details required to build a correct model. The bottom-out hint was necessary because the tutor required students to correctly solve a problem before going on to the next problem. The instructions emphasized that students should only ask for hints when they were truly stuck, and they should self-explain the hints so that they wouldn't get stuck again.

We chose this task for several reasons. First, modeling is a part of the Arizona high school science standards. Second, it is not a familiar task for students, so we could safely omit a pre-test. Third and most importantly, the task is not intrinsically motivating, which was necessary for our purposes, as we wanted to study self-regulatory behavior in tutoring systems with the aim of discovering ways to improve it.

Although we wanted to study self-regulatory behavior in a classroom setting, our task was not normally taught in schools, so we ran our studies in summer camps for high school students. The summer camps were intended to be a fun introduction to engineering, so students did not receive grades. In order to simulate the extrinsic motivation normally provided by grades in classrooms, we awarded points during the instruction, and at the end of the experiment, we gave prizes (e.g., headphones) to students who earned enough points. The exact method for awarding points will be described in a moment, but was strongly related to the number of levels a student achieved.

Because the students were available for only one short period, we chose an experimental design where they would all work for the same period of time (three hours, with a water and cookies break midway through). The main dependent variable was a measure of their competence, namely, the number of points they earned. This aligned the competence measure and the extrinsic reward, just as grades do. Our goal was to use this measure of competence to distinguish fast learners from slow learners, then see what kinds of self-regulatory behaviors were exhibited by each type of learner.

4. Lesson 1: When to level up

This paper is intended to report methodological "lessons learned" from implementing the level up procedure during the 2009 summer camps. The first lesson relates to determining exactly when to level up. In a typical game, players level up when they exhibit a certain

degree of competence, such as defeating a certain number of enemies. With tutoring systems instead of games, such a procedure might be too simple. First, tutoring systems give hints, so solving a certain number of problems could not be used as a criterion for leveling up unless one somehow accounted for the use of hints in solving the problems. Second, tutoring systems can be more boring than games, so giving students some control over leveling up might increase their motivation [6].

In study 1, students could level up any time and the points they got depended on the hints they took. In particular, the points awarded per problem were (4-H)*L where H was the number of hints received and L was the level. Thus, a student who saw only the weakest hint while solving a level 2 problem would get (4-1)*2=6 points. This scheme was too complicated. During focus groups after the experiment, students said that they did not understand the point scheme, so they just avoided using hints. In fact, the log files showed that they almost never used hints, floundered frequently, and leveled up slowly. For instance, after 35 minutes, 18 of 27 students were still stuck on level 1.

Study 2 simplified the point system. Students were required to solve at least two problems per level. On the first problem of each level, they could use as many hints as they wanted. On the subsequent problems of the level, they could choose to level up after completing the problem, but they got points only if they completed the problem without hints. Students now said that they understood the scheme. Moreover, they now used hints more frequently, and this caused more learning and faster leveling up. However, we were surprise to see that students only chose to level up after solving a problem without hints.

This suggests using a more game-like procedure: *After the first problem of the level, students automatically level up when they have solved a problem without using any hints.* This would have been simpler and yet would have worked equally well for encouraging appropriate use of hints. Although it might hurt motivation in some contexts, it would not hurt motivation in our context. In our summer camp setting, students were seated at carrels in a large lab and allowed to talk as much as they wanted. Some students loudly announced their leveling up, creating a lively competitive atmosphere, just as one used to find in gaming parlors. Although competitions have their drawbacks [7, 8], it is rare to find this level of engagement on a task that is difficult and not intrinsically motivating. Thus, the first lesson learned is to use the italicized rule above for leveling up.

5. Lesson 2: Isomorphic problems invite copying

When designing the levels for the level up procedure, one naturally tries to teach one concept or skill per level. Thus, there is a tendency to make all the problems on a level have structurally similar solutions. That is, they are isomorphic. In our task domain, where the solutions are node-link graphs with equations inside the nodes, the solutions were graphically isomorphic: they had the same directed graph structure, with only the node names and positions being different. The strong similarity among the problems of a level was due to the simplicity of the concepts and their gradual introduction, one per level.

Unfortunately, this meant that students could copy models instead of learning how to generate them from basic principles. Each level began with a short text that taught the new concept and gave an example of a model using it. Some students quickly learned that they didn't need to read the text, because all the models on a level would be look exactly like the example. For instance, given Figure 1, which appears in the instructions for level 8, and a problem on level 8 that specified three nodes (e.g., rabbit population, rabbit deaths per month, rabbit death rate), some students merely substituted "rabbit population" for "barrel", "rabbit deaths per month" for "emptying" and "rabbit death rate" for "drain rate." They

approached this as a matching game, with feedback on their guesses provided by the tutoring system. They could rapidly solve the problems without using any hints.

Such copying behavior is a type of gaming [9, 10] that is associated with poor learning [11]. Although one could perhaps hide the instructions that contain the source (the model being copied in our case) while students are trying to solve problems, this might harm students who were trying to learn and were not trying to game.

Instead of trying to prevent such copying, it might be better to design the instruction so that copying yielded the desired type of learning. There is a rich literature on how to design example/problem combinations in order to increase learning [12]. In studies conducted in the 2010 summer camps, we integrated all the examples into one instructional module that was presented prior to any levels. Problems slowly increased in complexity and yet no adjacent problems were isomorphic. Copying dramatically decreased.

6. Conclusions

To summarize, lesson 1 was that leveling up should be controlled by a simple rule: After the first problem of a level where students should use hints freely, students automatically level up as soon as they solve a problem without using hints. Lesson 2 was that levels should not be composed of isomorphic problems, as this encourages mindless copying. Instead, instruction and examples should precede the levels, and levels should contain non-isomorphic problems. Lastly, a third lesson is that when the level up procedure was used in our pseudo-class, summer camp context, a lively competition emerged that was highly motivating to some students.

Acknowledgement: This research is supported by NSF DRL-0910221.

References

1. Anderson, J.R., et al., *Cognitive Tutors: Lessons Learned.* The Journal of the Learning Sciences, 1995. **4**(2): p. 167-207.
2. VanLehn, K., *Intelligent tutoring systems for continuous, embedded assessment*, in *The future of assessment: Shaping teaching and learning*, C.A. Dwyer, Editor. 2008, Lawrence Erlbaum Assoc: New York, NY. p. 113-138.
3. Bloom, B.S., *Time and Learning.* American Psychologist, 1974. **29**(9): p. 682-688.
4. Leelawong, K. and G. Biswas, *Designing learning by teaching agents: The Betty's Brain system.* International Journal of Artificial Intelligence and Education, 2008. **18**(3): p. 181-208.
5. VanLehn, K., *The behavior of tutoring systems.* International Journal of Artificial Intelligence and Education, 2006. **16**: p. 227-265.
6. Lepper, M.R. and M. Woolverton, *The wisdom of practice: Lessons learned from the study of highly effective tutors*, in *Improving academic achievement: Impact of psychological factors on education*, J. Aronson, Editor. 2002, Academic: New York. p. 135-158.
7. Senko, C., C.S. Hulleman, and J.M. Harackiewicz, *Achievement goal theory at the crossroads: Old controversies, new challenges, new directions.* Educational Psychologist, 2011. **46**(1): p. 26-47.
8. Ames, C., *Achievement attributions and self-instructions under competitive and individualistic goal structures.* Journal of Educational Psychology, 1984. **76**(3): p. 478-487.
9. Muldner, K. and C. Conati, *Scaffolding meta-cognitive skills for effective analogical problem solving via tailored example selection.* International Journal of Artificial Intelligence and Education, 2010: p. in press.
10. Baker, R.S.J.d., et al., *Why students engage in "gaming the system" behavior in interactive learning environments.* Journal of interactive Learning Research, 2008. **19**(2): p. 185-224.
11. VanLehn, K., *Analogy events: How examples are used during problem solving.* Cognitive Science, 1998. **22**(3): p. 347-388.
12. Renkl, A., *Worked-out examples: Instructional explanations support learning by self-explanations.* Learning and Instruction, 2002. **12**: p. 529-556.