TESOL Connections
Keeping English language professionals connected

Try, Fail, Succeed, Repeat: Taking Risks in STEM

by Katie Miller and Holly Sawyer

All educators can agree that fear of failing is an obstacle to learning. Nobody fears success. We're afraid to try because we're afraid to fail. Our multilingual students must overcome their affective filters to communicate in English. There are many kinds of fear our learners may face: Newcomers can be so overwhelmed in a new culture and school environment that they become "deer in the headlights." Kids can be mean, so why would anyone volunteer an answer only to be met with ridicule if wrong? Some students have unrealistic expectations for their learning, like the student who says they'll learn English in one year, or the one who's demoralized by one incorrect answer.

Our goal is to move students away from the "wrong is bad" trope. In our high school STEM and English Language Development classes, we incorporate many opportunities for students to fail on purpose in order to increase their confidence and allow them to ultimately succeed.

Why STEM?

English learners (ELs) are significantly underrepresented in STEM classes and careers (National Academy of Sciences et al., 2011; National Academies of Sciences et al., 2018). Despite the challenge of learning academic English alongside diverse content-specific knowledge, secondary ELs experience better outcomes with opportunities to learn academic content (Tolbert, 2018).

Math, engineering, and coding classes provide unique opportunities for ELs to engage in problem-based learning, where solutions often fail on an attempt. A regular engagement with failure and success cultivates a mindset, computational thinking, where problem-solving moves from an act to a practice. It works like this: When a human interacts with a computer, extreme precision is required and if an error ensues, it's likely from the human side. The human is responsible for communicating effectively. If one direction in the algorithm is off, then the entire program won't work. This generates many problems, which become expected. Their frequency brings a tolerance and acceptance of making errors. This valuable skill benefits our students as they engage with a new language and, undoubtedly, make errors. The risk of making errors can bring fear, but with a computer, the stakes are low or nonexistent: Computers won't pass judgement.

Testing an algorithm for errors is a necessity. We want students to expect to try, fail, succeed, and *repeat*. This also requires a mindset shift. Normally in education, an end product is completed and then evaluated; this is known as waterfall methodology. In contrast, agile methodology has students regularly test their program as it is being coded. (Read about both waterfall and agile methodologies in the free issue of *Hello World* magazine, "The Big Book of Computing Content.") Testing in agile methodology is like the approach's namesake: flexible, incremental, and iterative. Microerrors are caught and corrected frequently. This ongoing testing normalizes failure: "Here we go again, teacher." Students begin to expect to struggle, likely fail in a communication attempt, work through problems, and repeat the process all over again. We see persistence and resilience develop—all important in learning a new language.

Combating Math Phobia

Many people, of any age, often have a visceral reaction to math. Fear comes from not knowing how to approach a problem and in having to decide what possible tools you can use to solve it. We start with teaching formulas and processes with defined steps, such as how to simplify an equation. To help mitigate this fear, here are some strategies you can encourage students to try:

- **Answer Bank:** If they don't arrive at one of the answers on their first attempt, they try again. Students perceive the task as matching to something already there, providing a small sense of security.
- **Mini-Whiteboards:** Students who are paralyzed with fear and don't know what to do can start by copying the problem and discreetly flashing their first step attempt at the teacher during guided practice. Constant feedback encourages them to make corrections to their formula. Students can show each other their models where different, and creative solutions are anticipated.
- **Start From the Solution:** Take fear of the unknown out of the equation: "Spoiler alert! X equals 3!" Have students work through the process and show the steps to arrive at that correct answer. They will still demonstrate understanding of the skill, but the fear of not arriving at the correct destination is gone.
- **Calculator:** Many of our high school newcomers have had limited or interrupted formal education and lack math fact fluency. However, we don't want the fear of miscalculating *6 x 7* to stop them from attempting a solution in algebra class. Using a calculator frees up mental energy to apply formulas correctly or demonstrate they know which numbers to calculate (and how and why).

Engineering Design: Process vs. Product

We teach the engineering design process as an iterative process always focused on improvement: Define the problem, brainstorm ideas, explore solutions, generate a model, and test and evaluate.

Whether using LEGO blocks or computer-aided design programs to model designs, it's often easier for students to completely take apart, erase, or start over if something goes wrong. They're not starting from scratch, but from experience. When we build wooden structures like towers or cranes that are designed to hold weight, we will test until they literally break apart (see Figure 1). We don't know how much weight they can bear, or know the limit of the designs until we push past those limits. Students take pride in constructing their design, but, ultimately, their beautiful structure will be smashed to bits.

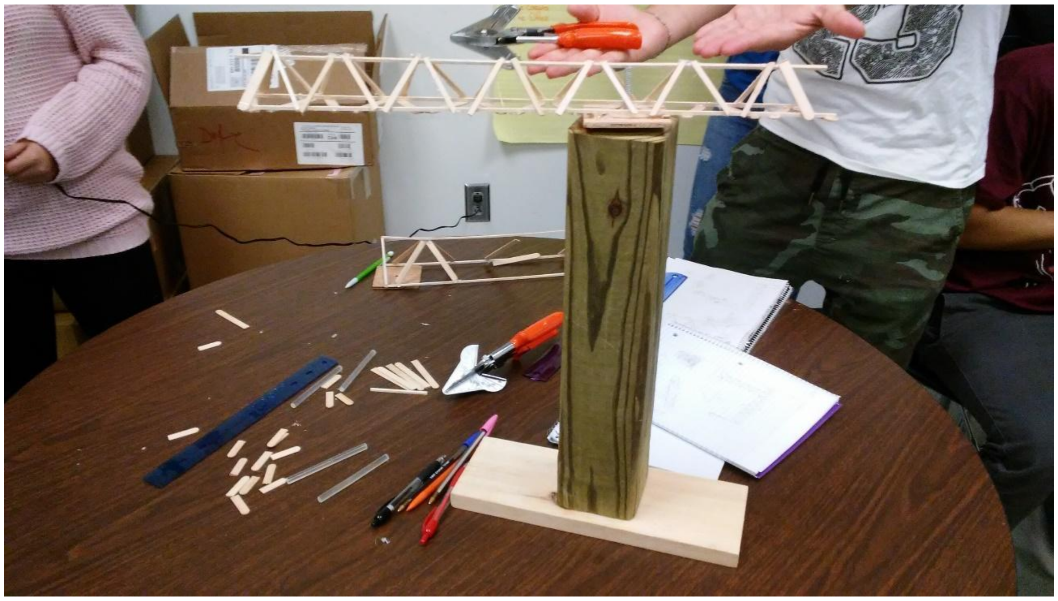
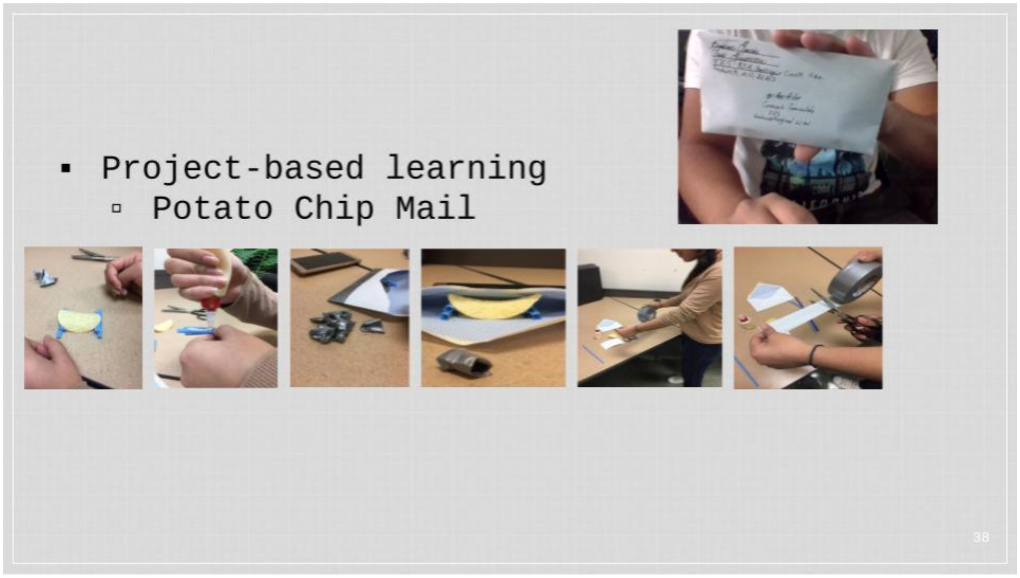


Figure 1. Building structures to test limits.

It’s important to separate the design from the student: The former fails, but the latter grows. A student can get an A for a project even if the design fails. By completing an engineering design journal for each project, we can see how the students thought through every aspect of their design, including criteria, constraints, materials, and math calculations.

- *Fun Classroom Challenge*: Use plastic straws, tape, and cardboard to modify an envelope to carry a single potato chip in the mail. The chips hardly ever come back intact after a journey through interoffice mail (see Figure 2).


- Project-based learning
  - Potato Chip Mail

38

Figure 2. Project-based learning potato chip mailer.

We also build Rube Goldberg machines (see Figure 3). Students must stop and restart their machine dozens of times before it works successfully. Though students do get frustrated when their machine doesn't work as planned, more often than not, they laugh. By normalizing the expectation of failure, you help them understand that they will have to continuously tweak their design and try again.

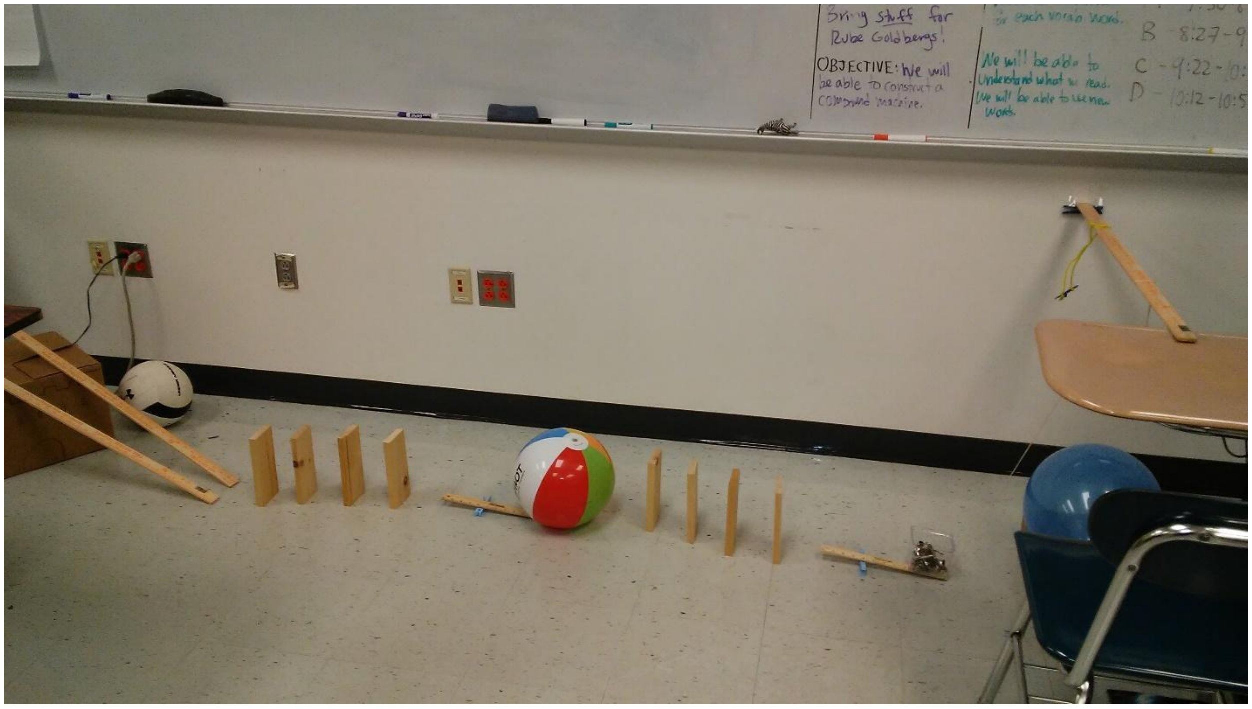


Figure 3. Rube Goldberg project.

Strategies to Fail Forward

1. Model Expectation of Failure

Even teachers fail! But we can demonstrate how to keep our cool and bounce back.

2. Unplugged First, Then Plugged Lesson Sequence

Students first work with the concept in a real-world, unplugged context. They then transfer the concept to their coding task. This embeds additional opportunities to try, fail—and succeed! It also animates them with a fun, highly engaging real-world activity before they sit down and extend the concept in technology.

- *Fun Classroom Challenge*: Write directions between two locations in the school for a classmate to follow. (E.g., "Move one step forward. Turn left. Move five steps forward.") Upon arrival, the classmates take a selfie in the location. Did they arrive where they were supposed to? Then, plug in and program a robot to move to a destination.

- ➢ *Fun Classroom Challenge*: To prepare students to code conditional statements, create and play a red light–green light game using an “If ___, then___; otherwise ___” sentence frame. Look out! Students have substantial fun sabotaging their friends with very personal statements based on what they know about each other. (E.g., “If you have a bird as a pet, then walk two steps forward.”) They write, then listen, react, and laugh. When it is later time to code conditional statements, they draw from this familiar language experience.

3. Teach the Language of Problem-Solving

Language and content go hand in hand: Knowing the language of the STEM area helps students understand the process, complete the steps of the algorithm, and organize thoughts for reflection. Begin by teaching the language to ask for help; provide sentence stems for asking for assistance from the teacher and peers, and for how to offer assistance to a confused classmate, because these can be used at any time. Following is some common language used in STEM that will help students solve their problems:

- *Coding*: conditionals with if/then, dependent clause, imperatives
- *Engineering*: text structures such as cause/effect, problem/solution, compare/contrast
- *Math*: multiple meaning words, such as *step* (on the stairs) and *step* (one operation in a process)

4. Collaboration

Students benefit from interactive peer support, or peer programming. This enables them to get/give more immediate feedback from peers (not just the teacher). We always have students work in pairs or small groups. It’s helpful when one student can give an oral direction and the other can enter the program or manipulate the design.

5. Manage Socioemotional Expectations

Emphasize progress and growth, not perfection.

- As students leave their comfort zones, teach them words to identify and express emotions, with attention to gradiency: “Are you annoyed, mad, or furious?”
- Take frequent pulse or vibes checks; for example, have students hold up fingers on a scale of 1 to 5 to answer “How are you right now?”
- Use humor to put anxious students at ease: “I will call 911 if you are injured by solving this equation.”

Try, fail, succeed, repeat: Teachers of ELs have a real opportunity to engage in this very process themselves. Try! Try to incorporate STEM content and be determined to learn alongside your students. Fail. But be agile and flexibly adjust your teaching practice. Succeed! Celebrate the learning of language and STEM, integrated. And, repeat.

References

Bell, Pete. (2022). Testing the ~~Fear~~ Love of Failure. In *The Big Book of Computing Content* (2nd ed.; pp. 136–137). Raspberry Pi Foundation.

National Academy of Sciences, National Academy of Engineering, & Institute of Medicine. (2011). *Expanding underrepresented minority participation: America's science and technology talent at the crossroads*. The National Academies Press. https://doi.org/10.17226/12984

National Academies of Sciences, Engineering, and Medicine. (2018). *English learners in STEM subjects: Transforming classrooms, schools, and lives*. The National Academies Press. https://doi.org/10.17226/25182

Tolbert, S. (2018). *EL STEM committee report: Secondary science education for English learners*. The National Academies Press.

***Katie Miller** is a National Board Certified Teacher who has taught ESOL and STEM in secondary and adult education programs. She was most recently a high school EL Department chair and is now an EL Fellow with the U.S. Department of State in Kyrgyzstan. Katie holds a BSLA in French from Georgetown University and an MA in applied linguistics and TESOL from Old Dominion University. She is a past president of Maryland TESOL (2020–2021).*

***Holly Sawyer** is a National Board Certified Teacher who currently teaches English Language Development in secondary school in Chesapeake, Virginia, USA and has also specialized in early childhood ESL. During pandemic teaching, she discovered coding as a vehicle for language learning, integrates it in her instruction, and documents her students' success on her blog. Holly is a Spanish learner with a BA in foreign languages and a MA in applied linguistics from Old Dominion University.*