Editor: Neil Maiden ■ City University, London ■ n.a.m.maiden@city.ac.uk

Theater Improvisers Know the Requirements Game

Martin Mahaux and Neil Maiden

Improvisational theater, or *improv*, is when actors simultaneously write, direct, and perform theater in front of an audience. Actors build on and act out ideas to interpret a theme given to them in real time. No actor knows what the others are thinking, but they act as if they're in the same world, imagining what others are doing, seeing, and hearing. Each responds to the other actors with new propositions that take the show forward, no matter how bizarre the direction might seem. These propositions build the performance piece by piece.

So what does improv theater have to do with requirements? In simple terms, improv supports team-based innovation. To achieve this, it provides techniques for improving stakeholder communication, increasing mutual understanding, and generating ideas that we can express as requirements. In this column, we propose improv theater as a training aid in requirements projects.

Improving stakeholder communication

Workshops for discovering requirements and their solutions occur in many development projects. However, running effective workshops can be difficult. Problems such as poor communication, lack of empathy with other stakeholders, and poor workshop facilitation can lead to stakeholder frustration and a sense that they're wasting time. In theater, improv coaches have faced similar problems for decades, if not centuries. Fortunately for us in software development, these coaches have developed and refined techniques that help participants listen to each other, empathize with others' positions, accept others' viewpoints, and build on each others' ideas—for example, not blocking, negating, or denying what someone else says, and making choices that take a scene forward. Analysts and stakeholders can benefit from using these techniques in their requirements projects and workshops.

Increasing innovation

Requirements engineering is a creative process in which stakeholders and analysts work together to create ideas for new software systems, which they express as requirements. Creativity is indispensable if software systems are to deliver progress and competitive advantage. Previously, we've reported on workshops that support creative requirements processes;[1] running these workshops, however, can be expensive and time consuming. We want to enable innovative thinking about requirements throughout a project. This is where improv training can help. It can help stakeholders become better innovative team players by encouraging them to reflect on their own strengths and weaknesses, learn about others' characteristics, and evolve their behavior accordingly during the project.

So what does improv theater give you?

Improv helps participants develop different soft skills through practice and play. One such skill is communication, which includes listening actively to everything that other people say and do and making yourself understood to others in the same way. Learning to use body language in presentations is

0740-7459/08/$25.00 © 2008 IEEE

also important. Another skill is to foster creative thinking by developing trust between participants and exploiting other people's willingness to be creative. Participants must also be able to think on their feet by quickly assimilating, sorting, and analyzing information so that they can react to others. Learning negotiation and role-playing skills lets participants react positively to each other's disagreements, thus producing an outcome the whole group finds acceptable. Practitioners also learn to be open-minded in response to unexpected situations, build up their self-confidence through humor, and deliver results under pressure in risk-free training sessions.

So, how does improv help you develop these skills? It prescribes innovative team-playing rules for actors that, together, form a framework for communication and innovation (http://improvencyclopedia.org/references//Rules_of_Improv.html). Here are some simple examples:

- Don't block, negate, or deny what another actor says in a scene. For example, if one actor says to another "Hi, John!" the responding actor can't reply by saying that he's not John.
- Don't ask questions—an actor can always answer in an imaginary world, but you should bring new information to the scene rather than ask someone else to. For instance, the actor playing John shouldn't ask "Hello, what are you doing?" Instead, he might say "Hey Janet, I'm happy to watch TV with you!"
- Make choices that take the scene forward. When John proposes going to the football match, Janet agrees, and they both go directly to the next scene, at the football match.
- Make assumptions about your character and other characters clear to all actors in the scene, and endow other characters with new characteristics. For example, at the football match, John endows Janet with superhuman powers, and Janet joins the football game, scoring 23 goals.
- Treat information that you share with other actors as gifts. If you receive something, give something back. Having received superhuman powers from John, Janet responds by turning John into a frog and refusing to kiss him.
- Maintain 360-degree awareness based on whatever communication channel is available. Janet observes that John (now a frog) is smiling, concludes that he's too happy as a frog, and turns him into a computer!
- Always maximize the scene's value for other actors. In our example, the actors recognize problems arising from competition between themselves in the scene. So, Janet turns herself into a computer as well, and a Bluetooth-based love story ensues.
- Be trustworthy in your actions, and trust your fellow actors' actions. John recognizes Janet turning him into a frog as a means to improve the story rather than as an attack.
- Have fun constructing the scene, even if the scene itself isn't humorous. Fun is a prerequisite for innovation, so the participants should enjoy themselves to motivate further learning.

Improv's Origins

From http://en.wikipedia.org/wiki/Improvisational_theatre: Improvised performance is as old as performance itself. From the 16th to the 18th centuries, *commedia dell'arte* performers improvised in the streets of Italy, and in the 1890s, theatrical theorists and directors such as Konstantin Stanislavski and Jacques Copeau, founders of two major directions in acting theory, both heavily used improvisation in acting training and rehearsal. … Modern theatrical improvisation is generally accepted to have taken form in the classroom with the theater games of Viola Spolin in the 1940s and Keith Johnstone in the 1950s. These rehearsal-room activities evolved quickly to an independent art form worthy of presentation before a paying audience. Further reading suggestions can be found at www.improvcomedy.org/books.html.

Now that we've introduced some improv rules, let's look at how to apply them to requirements engineering.

How to deliver improv training in requirements projects

In recent years, exchanges between the improv theater and software development communities have created a pool of coaches who can tailor training programs to specific needs. The first author (Martin Mahaux) has considerable experience developing training programs composed of two three-hour sessions separated by a week. After being introduced to improv rules, participants take part in warm-up exercises to build trust. Follow-on creativity exercises promote more open-ended problem solving that builds on others' ideas. Only then do the improv sessions themselves occur.

Debriefing is crucial to analyze what happened in each session. Did the participants feel good about being there and with each other? What was easy? Difficult? Coaches use improv rules to constructively explain how the exercise succeeded or not. So, participants learn through direct experience rather than observation.

Successes and challenges

Requirements projects in the financial, retail, and consulting sectors have recently benefited from improv training. Individual feedback is often enthusiastic. Participants reported having better understood cultural problems in their projects, changed how their teams worked, and obtained insights into their strengths and weaknesses in requirements work.

Requirements communication and innovation remain two key challenges in software projects, and we need new techniques to improve both. Improv theater offers fresh insights and new methods that demonstrate real benefits in training for requirements projects, and we hope to report more about this in the future.

Reference

1. N. Maiden, S. Robertson, and A. Gizikis, "Provoking Creativity: Imagine What Your Requirements Could Be Like," *IEEE Software*, vol. 21, no. 5, 2004, pp. 68–75.

Martin Mahaux is a business consultant on information and communication technology at inno.com. Contact him at martin.mahaux@inno.com.

Neil Maiden is a professor of systems engineering at City University, London. Contact him at n.a.m.maiden@city.ac.uk.