the Rational edge
e-zine for the rational community

Explaining the UML

by **Joe Marasco**
Senior Vice President
Rational Software

The one true test of your understanding of any concept comes when you must explain it to someone unskilled in the art. For those of us who deal daily in technology, the most maddening variant of this challenge is to transmit your understanding to "civilians," that is, other intelligent people who have little or no background in technology. The reason this is so difficult is that you cannot fall back on technical jargon -- that shorthand that permits high-bandwidth communication with peers, while at the same time presenting a barrier to those unfamiliar with the lingo.

In fact, I have found that software people have difficulty explaining the nuances of their craft to other engineering professionals. On a recent trip to China, I needed to explain the UML (Unified Modeling Language) and its significance to technical managers who were not software professionals themselves. I had not anticipated that this would be a problem, but when I first mentioned "UML," I got nothing but blank stares. Before I could advance, I needed to get them grounded in UML. But how?

What follows is the ten-minute presentation I improvised and subsequently polished. When we're done, there's a neat irony that wraps up the tale.

What Is the UML, and Why Is It Important?

Let us begin with a simple example. If I write on the whiteboard:

1 + 1 =

anywhere in the world, people understand what I am trying to say. In fact, at this point, someone in the audience always volunteers "2"! When that happens, I complete the equation:

1 + 1 = 2

and explain that, not only are we understood around the world, but also we usually get the right answer, too.

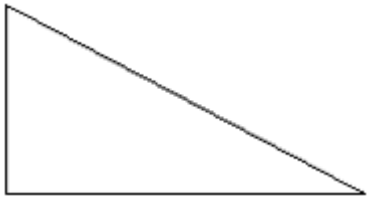
This is a good example of a *universal notation*, that is, the number system. People all over the world use it to communicate with each other. An English speaker can write it down, and a person speaking Mandarin in China can understand it.

Although this example seems trivial at first sight, it really does reveal an amazing fact: Numbers are universal, and certain symbols such as + and = have the same meaning all over the world.

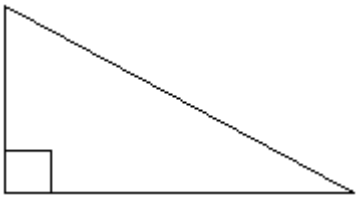
The other really nice thing about this example is that anyone who has a first-grade education can understand and appreciate it. It has the unfortunate disadvantage of appearing to be more trivial than it really is.

A Second, Less Trivial, Example

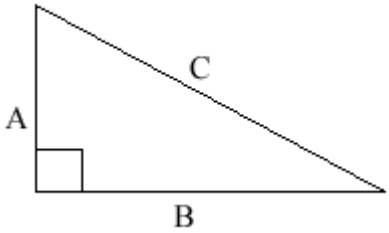
At this point I acknowledge that perhaps this first example is a little too simple. So I then draw a triangle on the whiteboard that looks like this:



I then point out that the triangle takes on additional meaning when I complete the diagram with the following addition:



Now this triangle is unambiguously a *right triangle*, because the little square doohickey is a worldwide convention meaning "right angle." Furthermore, I can now label the sides of the triangle A, B, and C:



And, immediately, we can write down that

$$A^2 + B^2 = C^2$$

Now this has a few very endearing properties. First, it is once again an example of a universal notation. Right angles, right triangles, and the symbols representing them are the same all over the world; someone from ancient Egypt could in principle reason about right triangles with a modern Peruvian by drawing such diagrams. What's more, once the diagram for the right triangle has been written down, the relationship of A, B, and C is defined. A, B, and C can no longer have completely arbitrary values; once any two of them are specified, the third is determined as well. The diagram implies the Pythagorean Theorem. One could even go so far as to say that the diagram has some "semantics," that there is a well-understood relationship between the picture and the values implied by the letters.

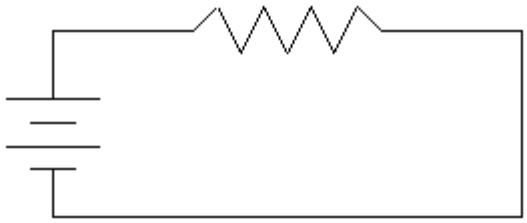
What is truly amazing about this example is that anyone with a high school education can understand it. If the person has seen any geometry at all, they have seen triangles and right triangles, and if they remember anything at all from their geometry, it is good old Pythagoras.

So now we have a diagram with semantics, and we have moved up a level of abstraction at the "accessibility cost" of moving from the first grade to the high school freshman level of mathematics. Also, at this point, people are definitely intrigued as to where I am going with all this. So I try to bait the hook with a very tasty worm.

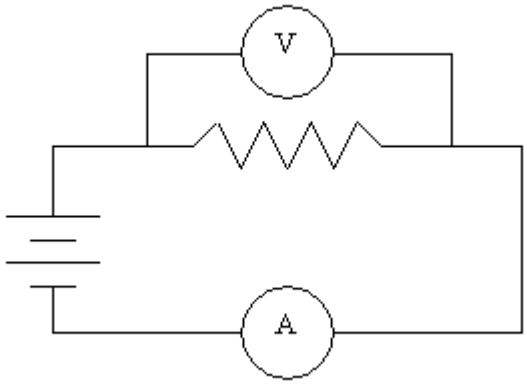
The Third Example

So far, these examples demonstrate the utility of a universal notation. The problem is, they are both from the world of mathematics; although math has concrete manifestations, it is intrinsically abstract. Are there any examples *not* from mathematics?

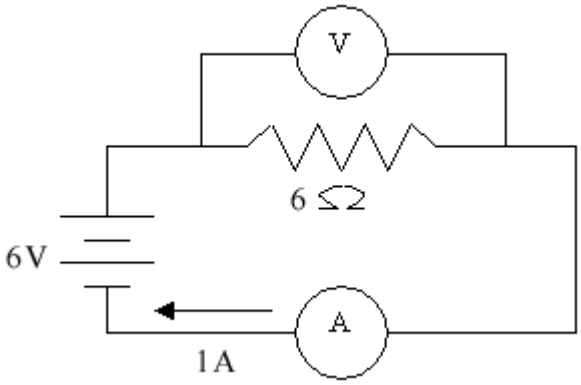
We then draw the following diagram on the whiteboard:



What is stunning about this picture is that as soon as I complete the drawing and say the words, "Here I have a simple circuit with a battery and a resistor," heads begin to bob. Of course, this is probably the simplest electrical circuit you could draw, but no matter. Just as the audience will applaud for itself when it recognizes the opening notes of Beethoven's Fifth Symphony, it will feel good about recognizing something technical. Without giving them too much time to think about it, I quickly add the symbols for a voltmeter and an ammeter.



And, in a final bold stroke, I note that if the battery is 6 volts and the resistor 6 ohms, then one ampere of current flows in the circuit:



Please excuse my rendering of the ohm symbol; it is important for the effect to use the symbol, not the word "ohm."

Now, people know what a 6 volt battery is; they can buy one in the store. And most people will have a recollection, however vague, that resistors are measured, or come, in units of ohms. So when you finally draw the "1 A" on the diagram, indicating that one ampere of current flows in the circuit (note that we even indicate the direction of flow!), people are totally convinced they know what you are talking about, even if they never could remember Ohm's Law.

This is a very good time to mention that a Swedish student and an Australian hobbyist can communicate about this circuit without knowing each other's language. Once again, an international standard notation has come to the rescue. Only this time it is not purely mathematical; the objects in the diagram have real physical instantiations. Moreover, semantics is in play: not only is Ohm's Law implied, but also implied is the direction of current flow that comes from our notions of the positive and negative terminals of the battery, represented by the long and short horizontal lines. I typically spend a few moments on the richness of the information communicated in this simple diagram, and remark how hard it would be to do any electrical engineering at all if we didn't have this notation that is the same all over the world.

Incidentally, we have moved the accessibility threshold up to anyone having had one year of introductory physics.

And Now for the Relevance to Software...

Now is the time to summarize that we have seen how progress is made in all fields by having a common notation that can be used to express concepts, and how diagrams begin to take on precision and meaning once we attach semantics to the pictures. The most useful of these notations are understood the world over.

But before 1996 there was no common notation for software. Before the UML became an international standard, two software engineers, *even if they spoke the same language*, had no way to talk about their software. There were no conventions that were universally accepted around the world for describing software. No wonder progress was slow!

With the advent of the UML, however, software engineers have a common graphic vocabulary for talking about software. They can draw progressively complex diagrams to represent their software, just the way electrical engineers can draw progressively complex diagrams to represent their circuits. Things like nested diagrams become possible, so different levels of abstraction can now be expressed.

Rational's contribution in this area is huge. By formulating the UML and bringing the world's most influential companies -- IBM, Microsoft, HP, Oracle, etc.-- to agree on it was a major step. Getting an international standards body -- the Object Management Group -- to ratify it as a standard was the formal process that irreversibly cast the die. Everyone agreed to end the Tower of Babel approach and also agreed about how to talk about software.

The significance of the UML is now established, and we can move on.

Final Thoughts

Of course, the UML itself is an example of "technical jargon." It is now the way software professionals talk to each other about software. As each example of a notation becomes deeper and denser, it can become an esoteric and subtle way of expressing ideas and designs that are very rich and complex. Yet, at the outset, this (and any) notation, at its highest level of abstraction, is useful for communicating between professionals and "civilians." That is because its fundamental elements can still be used to transmit fundamental ideas. A truly great notation "nests" and has many levels of abstraction; the highest levels facilitate communication between people who are "farthest apart" in terms of background and context, whereas the lowest levels (with the most technical detail) aid communication between people who are "closest together" in terms of their understanding of the domain -- the technical specialists.

What has been interesting about our journey this month is that we have used analogy to explain a technical notation. We have avoided the "self reference" trap -- i.e., we have explained the UML without describing the UML itself. We have explained the jargon without using the jargon. Although this seems like a subterfuge at first ("Hey, wait a minute; I never even got to see a UML diagram!") it is, in fact, a requirement that you be able to explain it without using it.

Otherwise, those "civilians" are going to block the first time you draw one. With this introductory context, however, I believe that the first UML diagram you do draw will be much better received. They will hark back to "1+1," Pythagoras, and Ohm's Law, and know that you are doing the same thing for software constructs.

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!