

Fractured Fables: The Wolf and the n Pigs: updated

Once upon a time there lived three pigs and a Big Bad Wolf. Pig1 constructs a house of straw; BB Wolf blows it down and eats pig1. Pig 2 constructs a house of sticks; BB Wolf blows it down and eats pig2. Pig3 constructs a house of bricks, which BB Wolf cannot blow down. Pig3 takes revenge on Wolf and kills him by boiling him in water, and Pig3 lives happily ever after.

This is an old "classic" fable, and we tell it to young children? So what does this "parable" have to do with computing? Well it says something about building properly using big building blocks.

The straw house could correspond to building in a low level language, close to the machine, with its many commands (load, store, shift, branch, goto, etc). Sticks correspond to the higher level commands such as if, while, for, etc. Bricks correspond to even higher level constructs such as functions and routines.

In computing we can go to more levels, both higher and lower levels. A lower level corresponds to building out of zeros and ones. The fable could be extended to involve a Pig0 who builds out of sand, jelly, sugar cubes or worse.

At the other extreme we can go to higher levels, building out of modules, classes, libraries, components. The corresponding Pig(n) would build using walls, rooms, and ultimately prefabricated homes.

In a possible sequel fable, BB Wolf gets reincarnated bigger and badder, as alien or computer, but Pig(n+1) is also reincarnated (by induction, yet), with new grail, silver bullet or other wonderful thing.

Movie rights are reserved.

Turtle and Rabbit Race

Two individuals, named Turtle and Rabbit, were applying for the same job. They were given the same programming assignment, due at the end of the day.

Rabbit went immediately to the computer, began programming, and declared he was 80 percent done by noon.

Turtle initially spent some time thinking, was slow to get to the computer, and did not have much code done by noon. She felt that she was 50 percent done.

Rabbit was so confident that he even took a nap after lunch. Then he began debugging, and by the afternoon coffee break declared to be 95 percent done.

Turtle continued programming after lunch, and just barely began testing by the afternoon tea-time break.

After the afternoon break they both continued testing. Rabbit often had to make significant changes to his code; Turtle had few and small changes to make.

At the end of the day both submitted their work to a test suite; Turtle passed all the tests, but Rabbit failed many of them.

Turtle was hired, but Rabbit continued, trying to complete the project, apparently far into the night chasing a few elusive bugs. He never did finish.

Moral of the story: Quick to code; slow to finish.