# Looks vs Brains

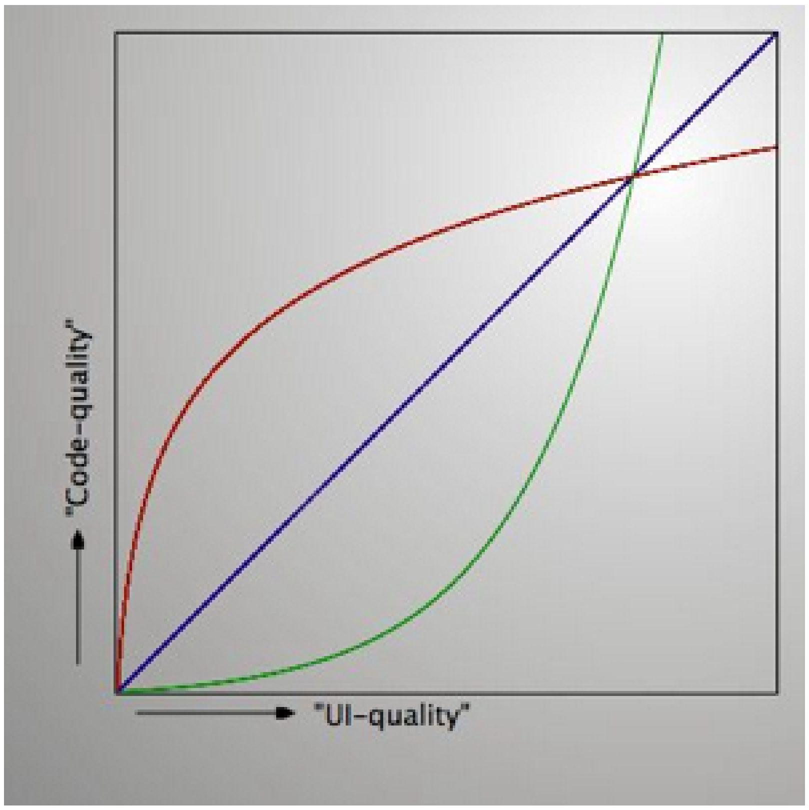
During their lives, people evolve, that's a given. For me, they start out as little ugly creatures, which most of us seem to like for some reason, and their brains don't seem to do very much either yet. (but who knows, really?) So, low score in looks and brains.

The first 25 years seem to deal mostly with getting the "looks" right. A minority succeeds. A substantially larger part of the population <strong>think</strong> they have succeeded.

In the next 25 years the brains (should) get some more attention. Possibly the earlier mentioned ratio of actual versus perceived success is even worse here. If there is a third or even a fourth quarter for people, looks are pretty much left behind, while in some cases the brains still can produce some occasional surprise.

The above is an intro to what I was really wondering about: How does a software package evolve with respect to "looks" and "brains" in its life? (As always i'm mostly focussing on the typical open source package and how that comes to be)

Being the person that I am, that brings roughly this image into my head:



Inevitably i tried to catch the question in some sort of equation. Here's roughly the train of thought:

- as the analogy is a "natural" thing some sort of "natural logarithmic" component came to mind
- work being mostly done in (large) groups which means that "things inforce things" (bad or good) some exponential part should be in there;
- it would be nice if we could have a simple number to characterize the curve, like a quality factor Q or something.

In the picture above the blue line would be the ideal(?) but rather boring mix where the looks and brains are always completely in balance. (The lifeline starts in the origin following the line with age). The red line would be a "code-quality" oriented progression where the green line focusses on looks. The rough idea is to get from the origin to some point on the blue line as far away from the origin as possible. By varying that Q factor that point can be placed anywhere on the blue line. The formula i came up with was this:

$$x = e^{\frac{y}{Q}}$$

which corresponds to the red line. (the green line is the reciprocal where x and y switched places). The interesting part is to play with the value of Q, look at the graph and reason about the software product. Or vice versa, reason about **your** project and determine it's Q factor.