Understanding ‘software quality’ is way more complex than can be given by a simple definition. Many people have thought about what quality means and almost all have come to different conclusions. According to Robert Glass’ *Defining Quality Intuitively* article in IEEE Software, he explains that quality is dependent on its products purpose and therefore defining an all-encompassing definition would be futile due to the main objectives to any end-goal of a project (Glass 104-107). Although there are some similar attributes that different products will share in terms of quality, there are also some that completely have nothing to do with each other.

That being said, the characteristics that make up quality, although may share the same purpose, implementing the correct ones is crucial for making a successful product. So, which are the proper attributes? Well, that depends on the product. Glass uses McDonald’s as an example of explanation here by stating that their “hamburgers match user requirements and affordability in a timely manner… but no one claims that McDonald’s is quality dining” (Glass 104). In this context, a quality assured product must meet affordability, timely, and user requirements but not so much a good quality dining experience. However, the expensive steak house downtown would challenge this assumption of satisfied quality assurance.

This same framework applies to software quality as well, and truly *is* important for the success of these types of products. Imagine a phone app that had to be constructed with the same quality assurance guidelines as the software used in your DVR at home. Just to pick out a few calamities, why would the DVR need a functional GPS locator? Or, in the case of multi-platform use, would AT&T-owned Direct-TV care if their equipment recorded on a Verizon box, but you could only use the phone app on iOS and not an Android operating system? That would be pretty detrimental to that app creator’s bottom-line, and not only would AT&T not care but would most-likely object to its software working on a rival’s platform. These are just a few examples off the top of my head, but it proves that having a general solution for quality would waste a lot of time and money. And anyone who has taken any kind of software engineering or computer science classes know – if you waste time or money, you better start looking for a new career.

So, what’s to stop us from committing this dreaded sin of wasted effort? If there is no single explanation on how to define good quality, how can anything be made or accomplished without flaw? The simple answer is testing. Testing is where all of the design, requirements, manufacturing, and engineering are all combined and checked for defects. Good testing starts at the beginning stages of a product or project and is meticulously woven into every aspect of what a good quality one is, then uses that information to weed out the poor-quality ones. Never is this more influential and apparent then in software. Afterall, not all baseball gloves are made similar, but every Windows operating system operates the same. By finding defective software, developers and engineers can make proper changes and adjustments which will in-turn produce good quality assured products.

**Reference(s)**

R. Glass, “Defining Quality Intuitively,” IEEE Software, May/June 1998, pp 103-107.