The number one reliability driver of today's system problems

Dr. Samuel Keene, FIEEE
s.keene@ieee.org

Consistently, the largest Pareto significant software and system problem lies with "requirements" deficiencies. First, it must be recognized that the customer who wants the product has an immature or imperfect notion of what is truly desired. The customer knows the main points of the desired product but has not realized the "totality of requirements". Then, the customer issues these imperfect requirements for the developer to interpret. Limitations of the natural language clarity represent another hazard to the perfect understanding of requirements.

In the 1990's, two contractors independently coded safety-critical code for the Canadian Darlington nuclear reactor. This code forced the reactor to go into a safe mode when the reactor coolant fell below a specified level. The coolant level was in a constant state of variation. There was interpretation ambiguity facing the contractors as to whether this contingent action was to take place when the mean, mode, or the median coolant level was used to trigger the action. It was later discovered that both contractors made the same misinterpretation. This ambiguity was subsequently corrected by mathematically stating the control invoking condition. This made the triggering condition explicit without relying on individual's interpretation of a natural language statement.

Requirements are always a big challenge. Brendan Murphy reported that the preponderance of system reliability problems stem from "System Management" deficiencies [1]. These are deficiencies resulting from incomplete requirements or interface definition. He published this finding a decade ago based upon over 2,000 Digital systems operating in Europe. He has related to me in the past year that this observation still holds on other systems that he is now tracking in Europe. The additional component that this author adds to System Management focus it the managing of product changes or product evolution. Changes degrade the product architecture and increase the system complexity. Maintaining situational awareness and design focus while introducing product changes is a special challenge for requirements management.

Managing Requirements

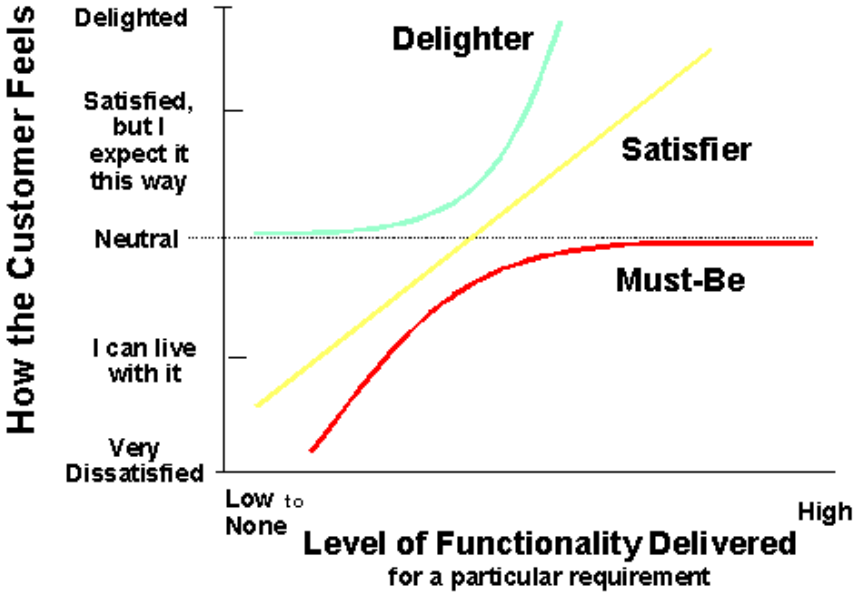
The Kano diagram depicts three levels of requirements:

- Delighters- which the customer did not expect but loves
- Satisfiers – which meet his specified needs
- Musts - often unspoken and not even realized until they are not met

The "delighters" are features that the customer did not specify, or even think of, but likes a lot. These can differentiate a product. A compass on rear view mirrors was an unexpected "delighter". Once introduced, these delighters cause the customer to demand them on future products. The compass became a specified "satisfier" to some customers.. Some believe that the "must have" requirements are those basic things needed to make the product work. This author feels that the "must have" are assumed capabilities. They

do not even have to be stated and are only noticed in their absence. Example would be a home sold in my area that did not have heat on the second floor. The buyer discovered this in the wintertime. He had assumed heat ducting would have been provided to all rooms. This lack was a surprise "dissatisifier".

Kano Diagram



The "must haves" requirements often lie in the infrastructure. Again these requirements are typically noticed in their absence. The author heard a recent example that likened oxygen to our infrastructure. We assume it is always present and always will be present. In its absence, we recognize quickly that it is a "must be" requirement.

There are two venues to collect initial requirements. First the customer/users can be directly asked. They identify their product needs. These typically center on their early thoughts on performance, quality and cost. These needs are given independently, whereas there are trade offs to be made, to reach an optimum balance. Typically these stated needs are the Kano "Satisfiers". The more they are satisfied, the better the customer feels about them, ie., the faster the response time of a device, the greater the satisfaction of the customer.

The more proactive developer will also actively seek "Context Data" about the customer needs. These are the indirect comments provided by the potential customers. These comments might be mined from help desk comments made on the present product. It might be product complaints that have been received. This is the data that can be mined to find "unstated requirements". These requirements have the potential to "delight" the customer, and successfully differentiate your product and raise the perceived quality of

your delivered product. Quality is the customer's perception of your product benchmarked against the customer expectation.

A leading computer manufacturer routinely invited some of its leading edge customers to participate in week long seminars. These customers were put up in a hotel and spanned different customer areas, e.g., food industry, automotive, consumer products. The host computer company provided facilitators to promote discussion. The facilitators would put out seed topics for general discussion such as "Managing accounts receivables". The listen for "best practices" or "challenges" these companies were facing. The facilitators would keep the ideas flowing and note themselves development opportunities whereby their products can better serve their customers. At the same time all the participants had the opportunity to learn from each other. It was an energized, "win-win" interchange for all.

I recently asked a successful realtor in San Diego, what she did to be so successful. Her response, "I listen to what the customers don't say". She was collecting "context data" to particularly find the customer delighters'.

So our challenge is to engage the customer to mutually help discover and communicate product requirements. Requirements are a discovery challenge for the customer as well as the developer. The customer is the "C" – the end customer, and the "c" – all the down stream functions in the developers organization that interact and help deliver the final product. The "c' includes the documentation group, service and test personnel and so forth. Six Sigma does provide a number of tools for cross group requirements discovery, ranking, cross-functional discussion and requirements capture. That topic will be elaborated on in a sequel article to this one..

1. B. Murphy, and T. Gent, "Measuring System and Software Reliability Using an Automated Data Collection Process", Quality and Reliability Engineering International, CCC 0748-8017/95/050341-13pp., 1995.