# Why is Antifragile more relevant for quality than Robustness

When traditionally thinking about high quality systems we think mostly about Robustness. The degree to which a component or system can function correctly in the presence of invalid inputs or stressful environmental conditions. This condition can be fulfilled with a fairly static environment, you can create robustness by limiting and highly control the changes done to your system.



Today’s condition for many businesses multiple trends combined, technology change rate, the highly integration between the business and IT, to serve the business, services on top of other services that are under constant change. Control is not done in isolation and the adaptability to change has increased, all of this sums up to higher complexity.

We usually talk about the role within system development cycle that has highest rank of design; an Architect. This might make us think of the role similar to someone that perform physical architecture, like building a house or a bridge. If we continue comparing these two roles similarities are very few.

A construction site will have some unknown, when appearing, these have some slight impact on the constructions and need to be taken into consideration. Thus very seldom once we build our foundation on Concrete Version 3.7, and our Walls Version 8.9 we understand that to be compatible with our new Windows 10.0 we need to upgrade Concrete Version 5.8 and this is totally different from 3.7 we have deployed, but our competitors are successfully using Concrete 5.8 that came with a new Carport Capability, and unless we use this we can stop this project all together.

The dynamics, change rate, always in design phase that our system are under, and needs to adapt, makes the role of a System Architect very different from physical construction. Maybe System Wizards that magically can resolve upcoming problems in what sometimes appears unrealistic ways is a better description for the modern version of the role. And our systems needs to be adaptable in ways we could not foresee with events that seems to have happened out of the blue, or as black swans (they exist but are very rare to come across).

Robustness does not say anything about in what ways our system can adapt or change with inputs. Changes that was in some ways, low probabilities but that once they appear becomes almost mandatory. To become robust one has mitigated risks based on knowledge of historical data, though rare, unlikely event cannot be found in the typical space of historical data. Though they can have high systematic risks towards the solution that is currently under construction or deployed. So to mitigate systematic risks, to work in a changed environment we must perform another type of stressors.

Nassim Nicholas Taleb came up with the word Antifragile, it is the opposite of Fragile, but in contrast to Robustness it is better to handle stressors, randomness, disorder and events sprung out of uncertainties. So how do we help the System Wizards winning the magic duels and have systems to be awesome in solving customer problems yet without taking Quality debt?

By studying the ideas behind Antifragile and accepting the complexity space that today’s system development is all about, we can think broader. A combination of tool support for computer capacity and human power that apply variability and stressors that strengthen the system makes it more antifragile.

This never ends, we cannot complete a set of defined historical set of stressors (automated checks) that apply to the future, but variability must be a key part of the applied test strategy.

A strong Test Strategy include variability as a key part with a combination of tools and human power. This needs to work together with an appropriate infrastructure, the support of automation for the purpose of variability rather than replacement of human power. That way we can support creation of Antifragile Systems that in better ways handle the context of today’s software development challenges. If going even further we can apply test strategies that apply both prio to and after production.

***Short about the Author****[Anna Almén](https://www.linkedin.com/pub/anna-sundin-alm%C3%A9n/8a/5b5/973) has long experience in the financial domain with both development and test perspectives. At AddQ she provides a strong holistic knowledge on the development life cycle and to build in quality assurance in the process. She has worked with defining and successfully implementing Test Strategies.*

*You can hear more from Anna talking about test strategies at our breakfast seminar 16:th of june. You’ll find more information* [*here*](http://www.addq.se/utbildning/frukostseminarium/).