HOW DO I LINK PROCESS CAPABILITY INDICES TO NUMBER (PPM) DEFECTS?

AMAZING processes are processes that are kept AMAZING

Workbook

Presented by

Dr Chris Jackson

Reliability engineer

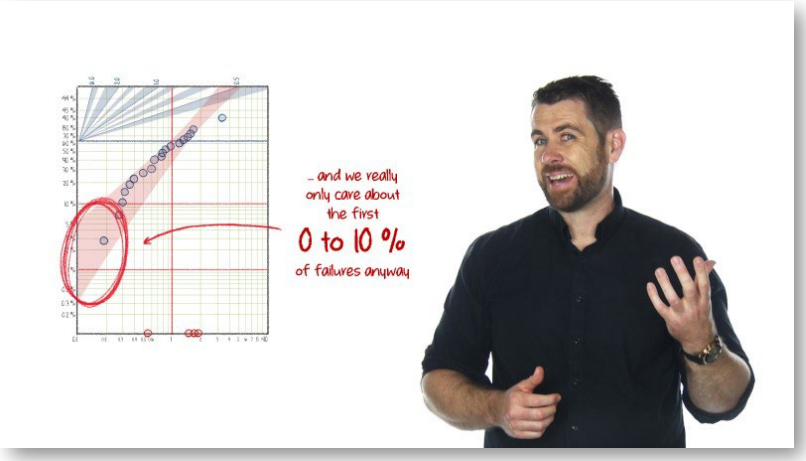
acuitas

reliability through insight

Who is your teacher? … Dr Chris Jackson

chris.jackson@acuitas.com

Dr Jackson holds a Ph.D. and MSc in Reliability Engineering from the University of Maryland's Center of Risk and Reliability. He also holds a BE in Mechanical Engineering, a Masters of Military Science from the Australian National University (ANU) and has substantial experience in the field of leadership, management, risk, reliability and maintainability. He is a Certified Reliability Engineer (CRE) through the American Society of Quality (ASQ) and a Chartered Professional Engineer (CPEng) through Engineers Australia.



Dr Jackson is the cofounder of Acuitas Reliability (www.acuitas.com) and www.is4.com online learning, was the inaugural director of the University of California, Los Angeles (UCLA's) Center of Reliability and Resilience Engineering and the founder of its Center for the Safety and Reliability of Autonomous Systems (SARAS). He has had an extensive career as a Senior Reliability Engineer in the Australian Defence Force, and is the Director of Acuitas Reliability Pty Ltd.

He has supported many complex and material systems develop their reliability performance and assisted in providing reliability management frameworks that support business outcomes (that is, make more money). He has been a reliability management consultant to many companies across industries ranging from medical devices to small satellites. He has also led initiatives in complementary fields such as systems engineering and performance-based management frameworks (PBMFs). He is the author of two reliability engineering books, co-author of another, and has authored several journal articles and conference papers.

Table of Contents

PROCESS PERFORMANCE GOALS

So what should we be 'aiming' for when it comes to our **processes**? Where do we need to be?

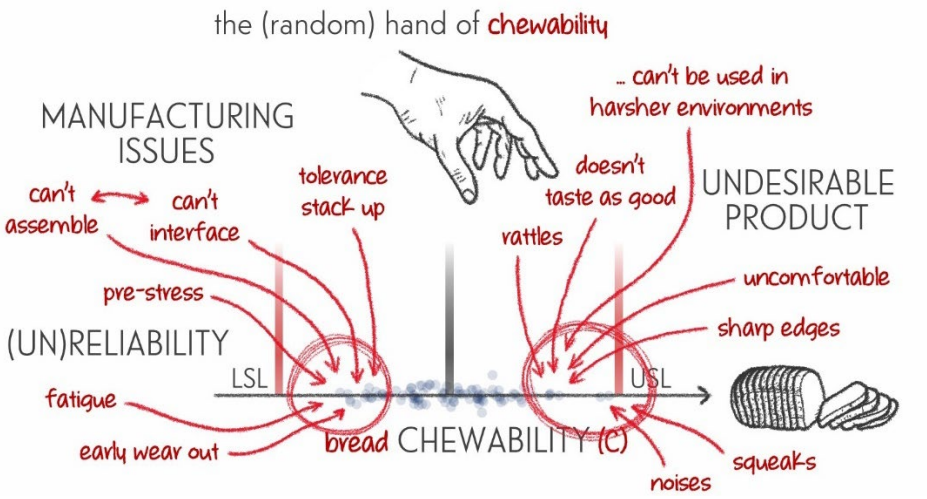
We have previously talked about how there is no such thing as 'good' and 'bad' when it comes to **processes**. There is no (or shouldn't be a) binary approach. We can use things like **capability indices** to help us come up with a number, allowing us to compare one **process** to another.

So the reason we need to come up with **performance** goals for EACH CHARACTERISTIC is to be able to compare the RELATIVE CAPABILITY of each **performance attribute**. Those **processes** that meet or exceed the aim are the TRIVIAL THOUSANDS we don't need to focus on. The remaining **processes** that DON'T meet the aim are those VITAL FEW that we need to improve. And if every **process** meets the aim … then there will always be a process that is the 'weakest' link. And to keep your overall production '**high quality**,' that is the **process** we need to focus on.

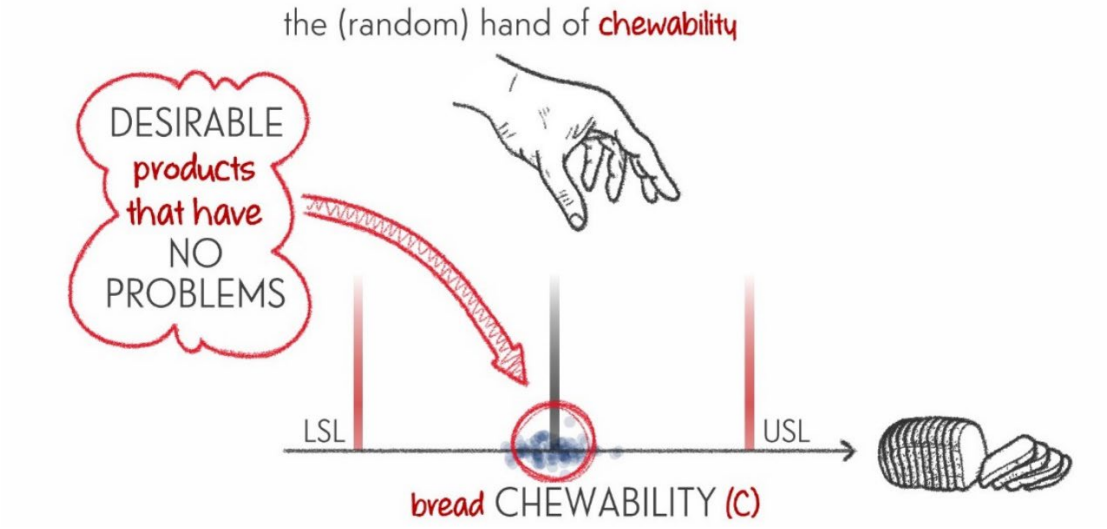
This is where we start to save time and money. We don't want to try and improve everything. If we know the 'weakest link,' then we only need to focus on that part of our production system to maintain **high quality**.

It is never a question of 'are we there yet?'

So we don't want our **process** to produce **products** like this …

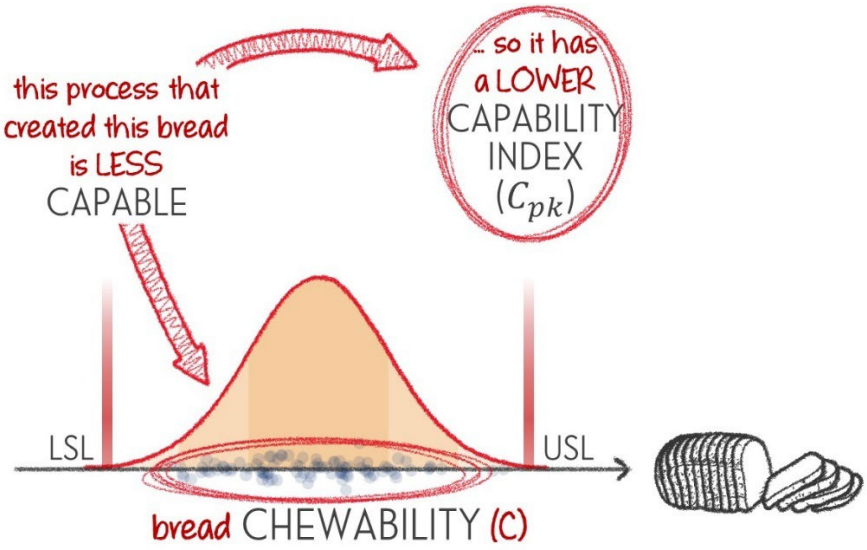


Instead, we want our **process** to look like this …

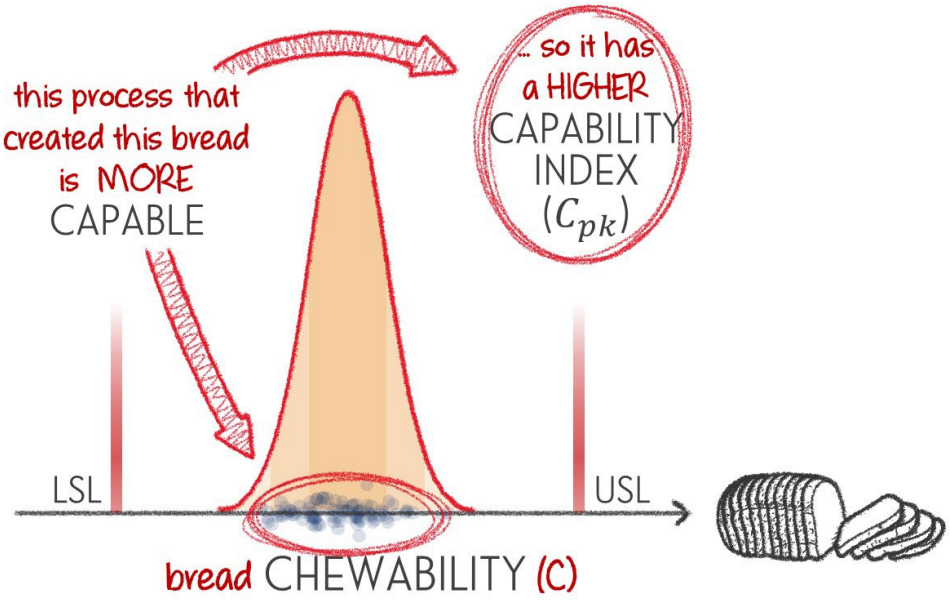


But perhaps the most 'valuable' trait of the **process** above is that any 'hint' of **out-of-control** behaviour is very obvious very early. As variation is minimized, changes in the underlying process are not hidden across a range of data points.

Our first process that has all those problems even though the observed data points are all within **specification limits** has a lower **capability index**.



Our second **process** which has few, if any issues has a higher **capability index**.

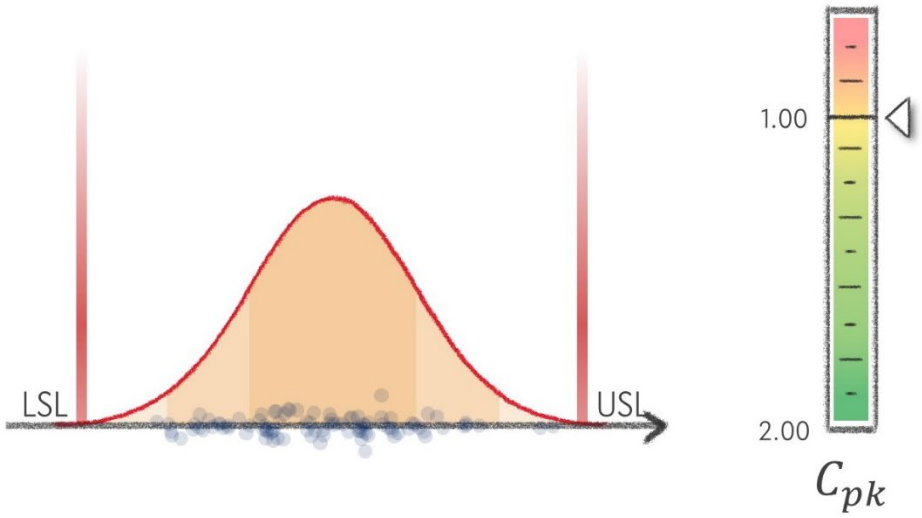


There are some VERY BASIC guidelines regarding **capability indices**.

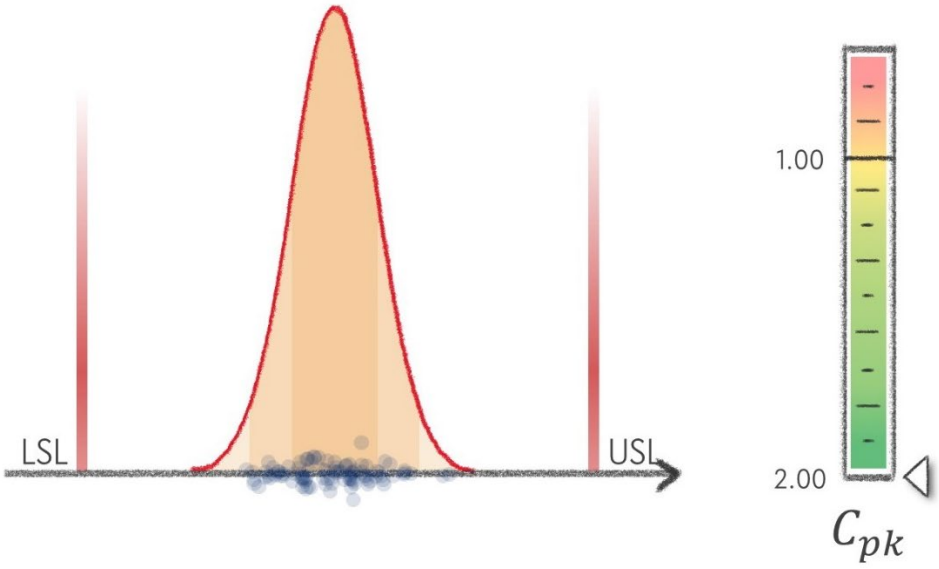
C_{pk}	two SPECIFICATION LIMITS	one SPECIFICATION LIMIT
not CAPABLE	< 1.00	< 1.00
barely CAPABLE	1.00	1.00
existing process	1.33	1.25
new process	1.50	1.45
safety-related existing process	1.50	1.45
safety-related new process	1.67	1.60
6σ quality process	2.00	2.00

These are arbitrary and should not be blindly applied to YOUR ORGANIZATION. They are useful for getting an idea of 'where you are at' in terms of maturity and the art of the possible. But you need to understand what goals apply to your product.

The '**less capable**' **process** illustrated below has a C_{pk} of 1.00. This represents the lowest level '**capability**' that industry accepts.



The '**more capable**' **process** illustrated below has a C_{pk} of 2.00. This is typical of a **high-quality process**.



let's talk about RELIABILITY ALLOCATION

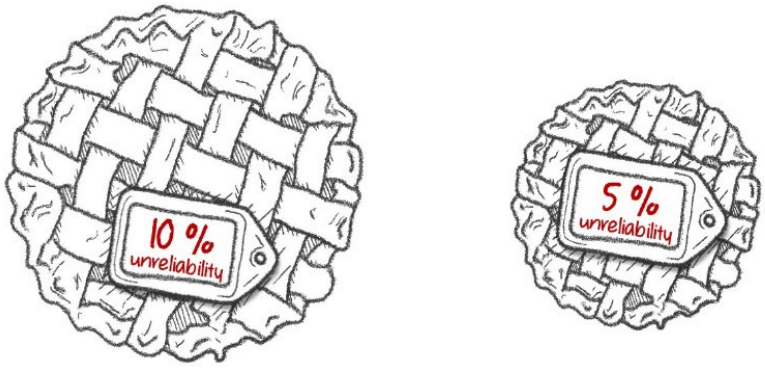
Why? Because this really helps us work out what MANUFACTURING QUALITY GOALS we need to come up with for each PART of our process. Or more correctly, each PRODUCT CHARACTERISTIC we have specifications for.

RELIABILITY ALLOCATION is the assigning of numerical reliability goals to subsystems and components quickly, cheaply and efficiently optimize system reliability performance.

There is a really simple underlying philosophy to **reliability allocation** that applies to **manufacturing quality goals**. We know that …

RELIABILITY is the (1) PROBABILITY that an item will (2) NOT FAIL for a (3) DESIGNATED PERIOD under (4) SPECIFIED CONDITIONS.

And now to **reliability allocation** – which can almost be looked at as how we divide a 'delicious pie' Or perhaps we should say the 'pie of Unreliability.' The pie represents how much '**failure probability**' we can tolerate for our system. For example, if we are aiming to produce a product, system or service with a **reliability** of 90 %, then our pie of UNreliability has a size of '10 %.' If we aim for a **reliability** of 95 %, then our pie of UNreliability now has a size of only '5 %.'

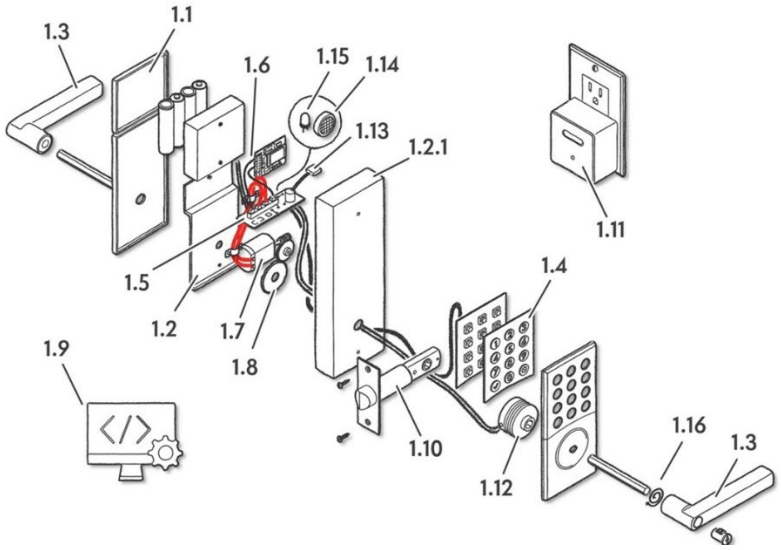


PIEs OF (UN)RELIABILITY?

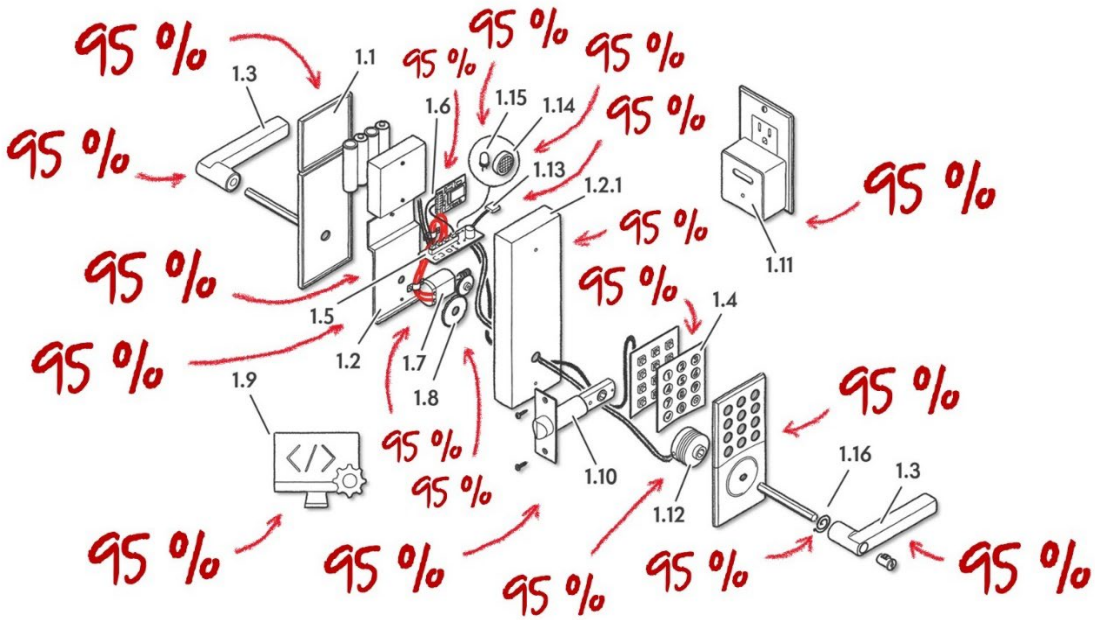
From a design perspective, we want a 'big' pie. This means we don't have to worry quite as much about making our system **reliable**. Which is inherently what most designers want to do. They are just like most humans who want to create and own something. And the smaller your pie of unreliability, the harder it is for your product, system or service to be '**reliable**.'

We work out how we slice up our pie of UNreliability and spread it across all components and subsystems. If we are smart, we will give bigger slices of pie to those components and subsystems that face lots of **reliability challenges**. But how do we do this?

Well, before we talk about how to do **reliability allocation**, let's look at how NOT to do **reliability allocation**. So let's look at a relatively complex system like a smart lock, with all its different components.



Let's say that we have come up with a **5-year service life reliability goal of 95 %**. Let's now say that an inexperienced design team leader then also comes up with a **5-year reliability goal of 95 % for each component**.

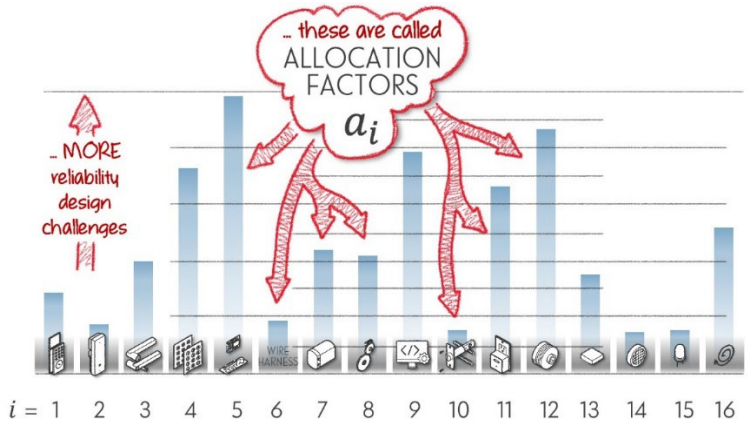


The problem with this is that for the smart lock to not fail, then ALL 17 components [1] need to not fail. So if all 17 components have a 5 % probability of failing in a 5 year interval, this means the overall **5-year system reliability is 42 %**. So we need to not only work out which components get bigger pieces of our pie of UNreliability, but how we slice each pie to make sure we realize our **system reliability performance requirements**.

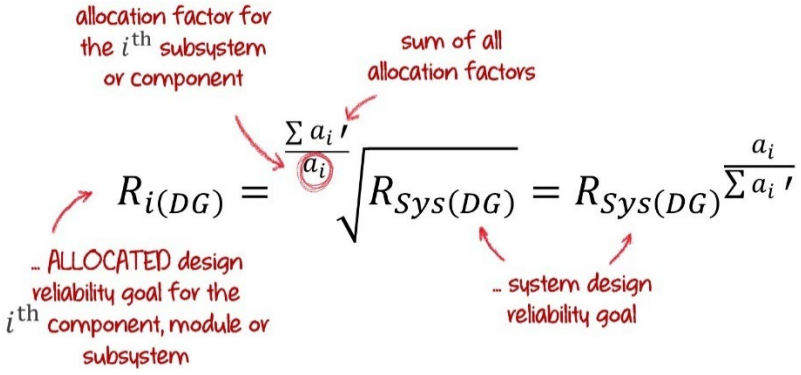
So how do we get this right? This is where our design team leader, chief reliability engineer, or whoever is responsible for the **reliability** becomes more important than ever.

In one scenario, a **reliability engineer** got a team of 4-8 cross-functional representatives into a room, and then facilitated a general discussion about 'reliability challenges.' After some time, she asked them to anonymously score each component on a scale of 1 to 10 ... with 1 representing a component that is UNLIKELY to have significant reliability challenges, and 10 representing a component that is REALLY LIKELY to have reliability challenges. These scores are gathered and averaged to create what we call **allocation factors**.

When this was completed for our smart lock, the **allocation factors** for each component turned out as follows:

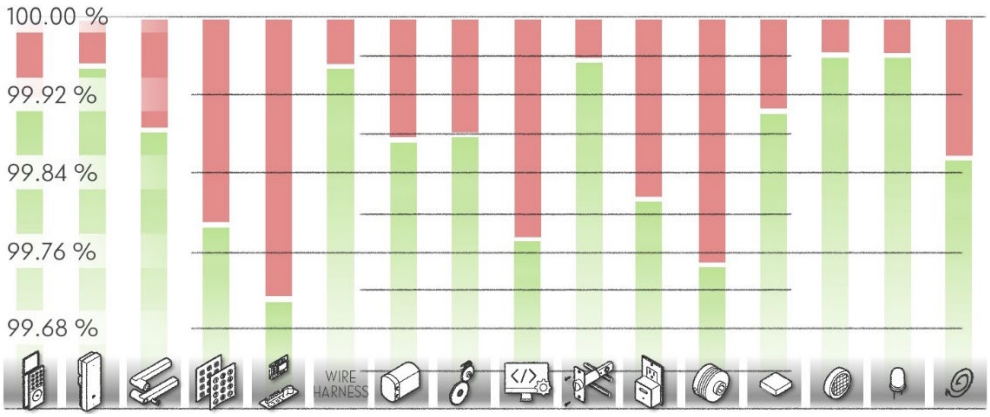


We then use these allocation factors to calculate subsystem and component **allocated reliability goals** with the following equation.



[1] Note that even though we have only labelled our components up to 1.16, there are two handles.

When we do this for each component of our smart lock, we get the following:



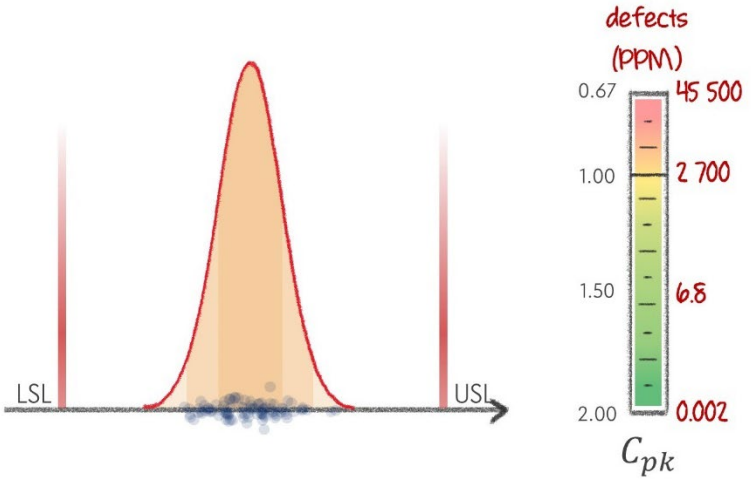
We can see that the 1.5 PCB had the highest **allocation factor** … meaning we expect it to have the most reliability design challenges. So … it gets the 'easiest' or 'lowest' **reliability goal**. Conversely, the 1.14 speaker and 1.15 LED have the lowest **allocation factors** … meaning we expect them to have the least reliability design challenges. And so they have the highest **reliability goals**.

… the most complex parts of our system to be the weakest parts of our system.

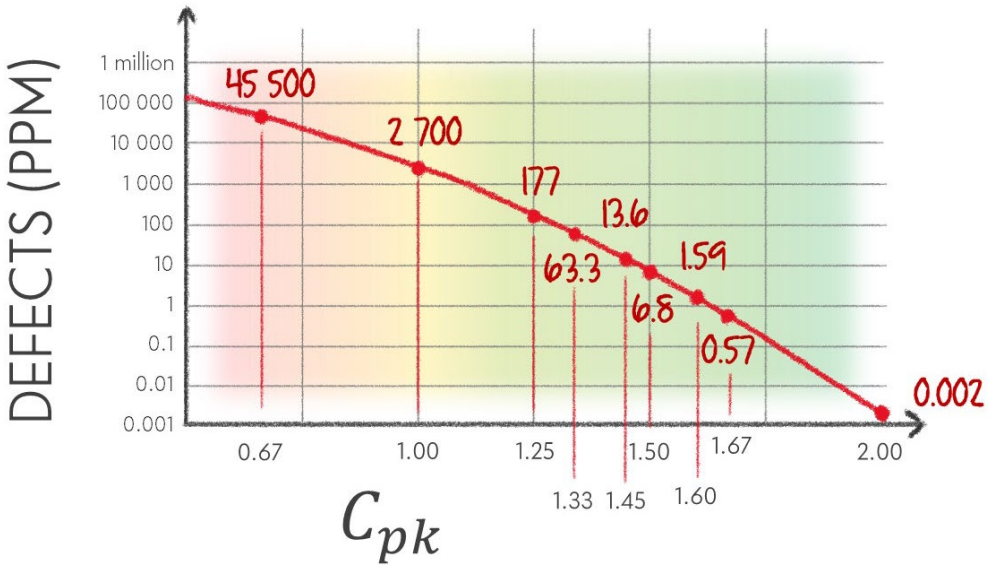
This might sound insane – but it is much harder and more expensive to make the most complex parts of our system to be the *most robust* parts of our system as well. It is much easier and cheaper for the most robust parts to be the simplest parts … or the parts we know how to make really **reliable**.

so how does this relate to MANUFACTURING QUALITY?

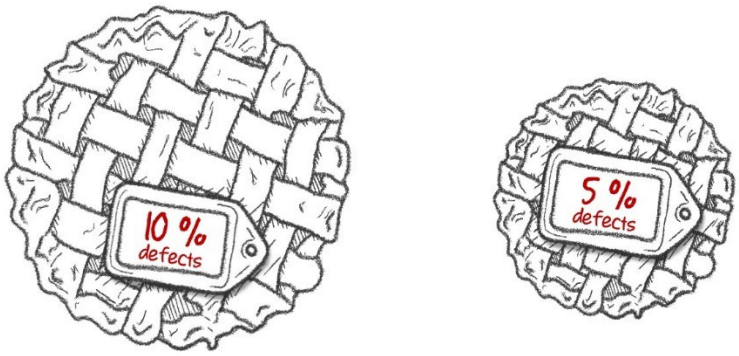
Our **capability index** is another way of representing the fraction of our products that are defective. For example, a **process** with a C_{pk} of 2.00 like the one below is expected to produce product with a **defect rate** of 0.002 PPM.



Defects (in PPM) for typical C_{pk} values are illustrated below. Note that the vertical axis (defects) is expressed along a logarithmic scale.



What this allows us to do is use **reliability allocation** concepts to identify **capability index goals** for key characteristics. So our pies become …



PIEs OF C_{pk}?

This means that we can come up with **capability index goals** for key **characteristics** based on how hard it is for them to be within **specification limits**. Those **characteristics** that we find 'easy' to be within **specification limits** are **allocated** the highest or 'hardest' **capability index** goals. Those **characteristics** that we expect to have the most challenges are allocated the lowest or 'easiest' capability index goals. This makes sure we prioritize effort on the production elements that matter the most.

So how do we do this? Using the following seven steps …

1. gather INFORMATION and prepare
2. establish PRODUCT QUALITY GOALS
3. determine QUALITY CHARACTERISTICS
4. create ALLOCATION FACTORS
5. ALLOCATE QUALITY GOALS
6. add MARGIN
7. LEAD and do something

STEP #1 … gather INFORMATION and PREPARE

Ideally, you and your team have already come up **with quality specifications** that are based on (amongst other things) the **reliability requirements** and other needs of your customer or user. If this hasn't been done … then you need to do it now!

If this has been done, and you have **quality or manufacturing specifications** for your product system or service, identify if this includes any **margin**. This is important for subsequent steps.

But our search for information doesn't stop here. We need to look for existing test data, field data, vendor performance sheets and anything else that can help us work out which components and subsystems will have the most trouble being **reliable and high quality**.

And perhaps the most important information source is the **system, design and process FMEAs**. This will provide us with all sorts of useful information that is already sorted and organized for the next steps in **manufacturing quality goal allocation**.

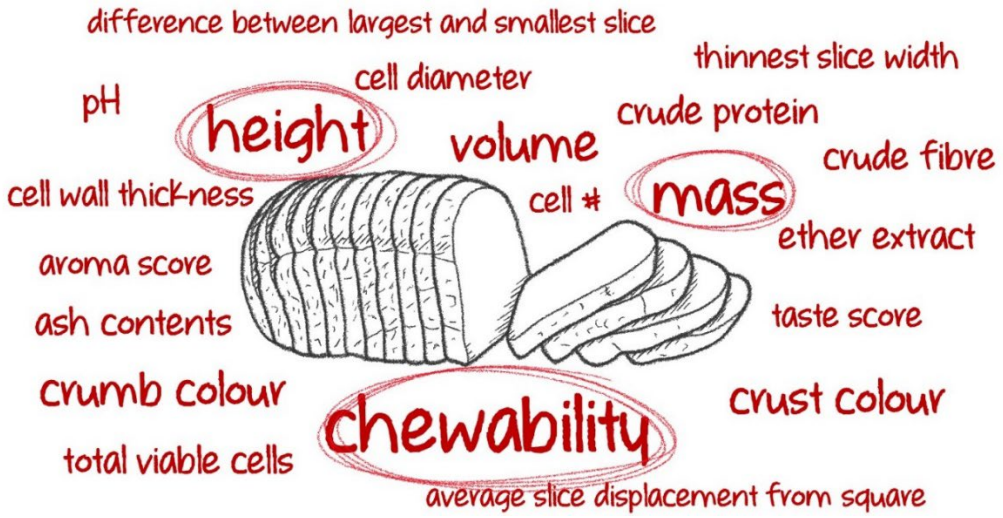
STEP #2 … establish PRODUCT QUALITY GOALS

Why do we (SOMETIMES) need to have separate SYSTEM QUALITY GOALS that can EXCEED SYSTEM QUALITY SPECIFICATIONS?

Many reasons. Perhaps you need to conduct demonstration testing which means your product, system or service needs to have a higher reliability or quality to have a reasonable chance of passing said test. Perhaps there are uncertainties in the use case, or there is a chance that the customer will use your thing in an unanticipated application, or you are unsure about what the service life might ultimately be. For any or all these reasons, you may need to have a **quality goal** that your product, system or service MUST achieve in order to meet or **demonstrate specifications**.

STEP #3 … determine QUALITY CHARACTERISTICs

Let's go back to our loaf of bread.



There are clearly lots of **characteristics** for our loaf of bread. Perhaps you need to consider all of them. Perhaps only a few are important. For the purpose of this discussion, let's assume that the three **characteristics** that 'matter' (or 'matter the most') are **height**, **mass** and **chewability**.

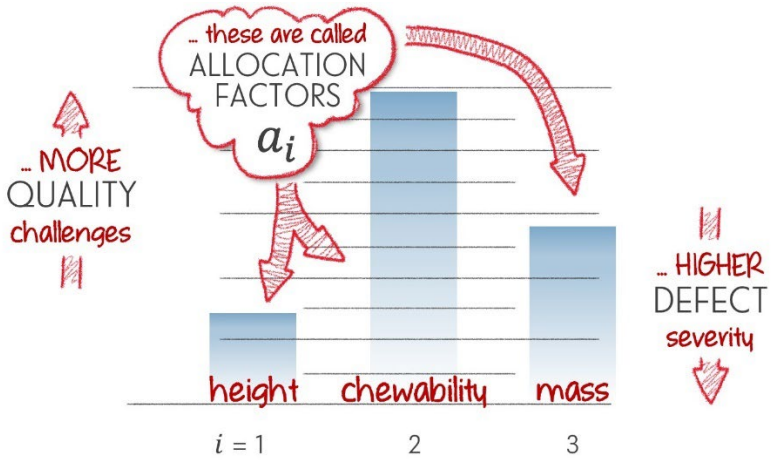
Remember that if everything is important, then nothing is important. Be judicious about which **characteristics** matter the most. In some cases, perhaps hundreds of **characteristics** are important. But if you can simplify, do so.

STEP #4 … create ALLOCATION FACTORS

This is where our design team leader, chief quality engineer, or whoever is responsible for the **quality** of our new product or system works out the goals for each component or subsystem.

This is where our **quality engineer** can do what the reliability engineer did above to allocate reliability goals. She can get a team of 4-8 cross-functional representatives into a room, and then facilitate a general discussion about **quality challenges**. She can also ask team members to anonymously score each **characteristic** on a scale of 1 to 10 … with 1 representing a characteristic that is UNLIKELY to have significant quality challenges OR may have severe defect effects, and 10 representing a characteristic that is REALLY LIKELY to have reliability challenges OR may have minimal defect effects. These scores are gathered and averaged to create **allocation factors** that we will use for our **quality goals.**

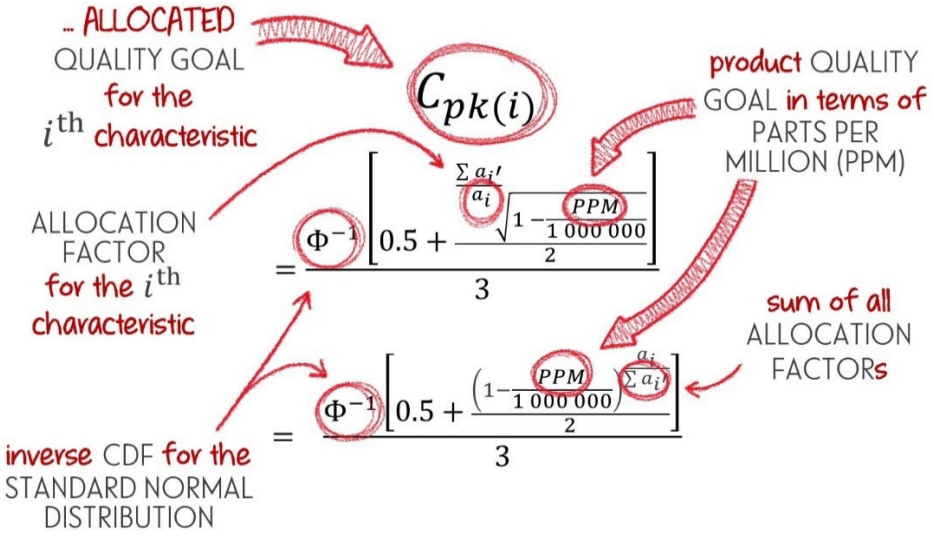
The relative scale for these **allocation factors** are illustrated below.



When it comes to **allocating reliability** or **quality goals**, the 'scale' of each factor is irrelevant. What is important is the relative scale of each factor to each other. These will then be used to create **manufacturing quality goals** for each **characteristic**.

STEP #5 ... ALLOCATE QUALITY GOALS

Once we have **allocation factors** for each **characteristic**, we use the following equation to come up with **capability index goals** for each **characteristic**.



The equation on the previous page is obviously quite complex! The Microsoft Excel formula for calculating these **characteristic quality goals** is …

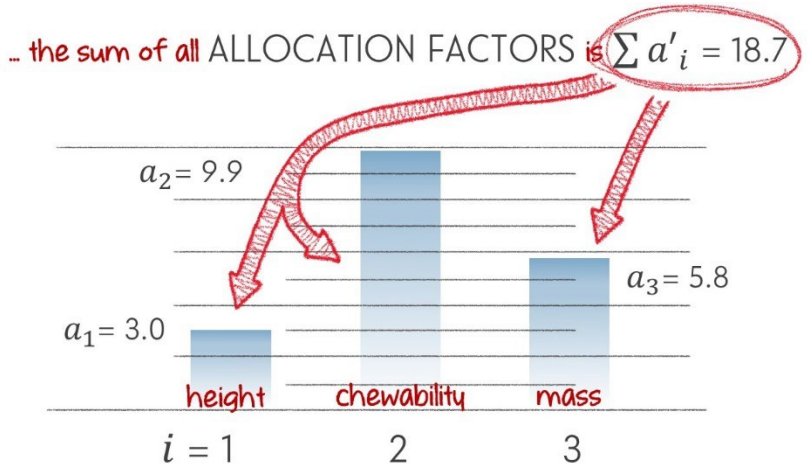
$$C_{pk(i)}$$

```
=NORM.S.INV(0.5+
(((1-([PPM]/1000000))^
([alpha_i]/[sum_of_alpha_i]))/2))/3
```

EXERCISE

Let's say that the QUALITY GOAL for our bread in terms of DEFECTS is 200 PPM. We have identified three key characteristics: (1) height, (2) chewability and (3) mass. Each characteristic has specification limits. So a loaf of bread is defective if its height OR chewability OR mass is outside of corresponding specification limits.

Let's look at the allocation factors that our lead quality engineer and her team came up with.

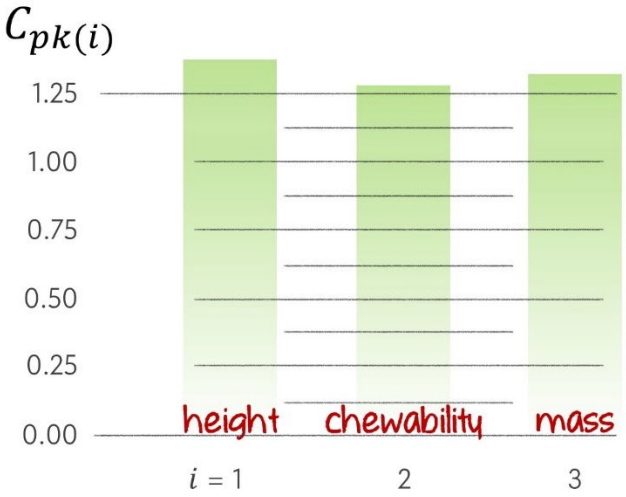


What is the allocated QUALITY GOAL for the CHEWABILITY in terms of C_{pk}?

```
=NORM.S.INV(0.5+
(((1-(      /1000000))^
(      /      ))/2))/3
```

… so therefore $C_{pk(2)} =$

So if we look at the **allocated manufacturing quality goals** for each **characteristic** ...



Chewability had the highest **allocation factor** and therefore has the smallest or 'easiest' **capability index goal**. **Height** had the lowest **allocation factor** and therefore has the highest or 'hardest' **capability index goal.**

STEP #6 ... add MARGIN

But there is a problem. Let's say our bread production system create loaves of breads with **height**, **chewability** and **mass** all described by the **capability index goals** above. This means that EXACTLY 200 loaves of bread out of every million are defective.

This is not good. Why? ... after all, we have met our **product quality goals**?

The reason this is not good (as it stands) is that there is no margin for error, degradation or problems.

As soon as our production system SLIGHTLY degrades or requires maintenance, we are now not meeting **requirements**. Our **process capabilities** need to combine in a way that exceeds our **quality goals** so that we have time to deal with the inevitable changes and challenges our production system will encounter.

A common margin that is introduce to our **capability indices** is 0.5. The 'six-sigma community' refer to this as a '1.5 sigma shift' as it represents an additional 1.5 **standard deviations** in the variation of our **characteristic** needs to include within **specification limits**. So our **characteristic capability index goals** now become this …

$$C_{pk(i)}$$

$$= 0.5 + \frac{\Phi^{-1}\left[0.5 + \frac{\sqrt{1 - \frac{PPM}{1\,000\,000}}^{\frac{\sum a_i'}{a_i}}}{2}\right]}{3}$$

$$= 0.5 + \frac{\Phi^{-1}\left[0.5 + \frac{\left(1 - \frac{PPM}{1\,000\,000}\right)^{\frac{a_i}{\sum a_i'}}}{2}\right]}{3}$$

The 0.5 can be seen as the analogous to the time we now give ourselves to address a problem that our **SPC** identifies before we no longer meet our **quality requirements.**

STEP #7 … LEAD and do something

This is the most important step. We can't just rely on our initial **allocated goals** to be 'perfect.' And because production challenges are (by definition) unanticipated … we will almost certainly have to adapt and accommodate. This is where that **quality margin** might just save the day.

But first, let's think about what would happen if we DIDN'T keep some **quality margin**. Everyone would get bigger slices of our pie of defects. And who doesn't like pie! Chances are they will eat every bit of that pie. In other words, **capability indices won't exceed allocated goals**.

But then there will be (at least) one characteristic which needs more pie. It came across one too many **production** challenges, and it is the only one that can't meet its allocated goal.

If we had kept some **quality margin**, chances are that some of our production team wouldn't have needed to eat all that pie. Keeping **quality margin** gives everyone **quality goals**. And this means people become aspirational … and we have some pie leftover to give to our struggling production engineer.

And (best of all) this costs us no time and no money. Just because we thought about QUALITY MARGIN from the start.

We have a list of six things we can do if and when one of our designers starts to struggle to meet its allocated goal.

1. (RE)ALLOCATE QUALITY from a characteristic that is 'DOING WELL'

… noting that this option is FREE and FAST – if we have QUALITY MARGIN

2. allocate quality from your QUALITY MARGIN

… noting that this option is ALSO FREE and FAST – if we have QUALITY MARGIN

3. consider OUTSOURCING

… which is something that many organizations like to avoid, but QUALITY ALLOCATION can help us realize if and when it makes financial sense to engage a third party who may be more proficient at manufacturing this component.

4. invest more TIME & MONEY into your struggling characteristc

… noting that QUALITY ALLOCATION will help you realize if you need to do this EARLIER, which of itself saves time and money

5 update the BUSINESS CASE warranty period, service life and so on

… noting that (again) QUALITY ALLOCATION will let you know if this needs to happen sooner rather than later.

It is way more important to LEAD AND DO SOMETHING than endlessly 'perfecting' allocated goals noting that we cannot anticipate every reliability design challenge.

… and there is more

This lesson is an introduction to **quality allocation** only. There are more courses you can take on **reliability allocation** that teach you how to create **allocation factors** on lots of different considerations including complexity, failure severity, the 'state of the art,' Technical Readiness Levels (TRL), weight, historical performance, or anything else that matters. There are also times when there are different levels of failure with different **reliability requirements** (critical versus non-critical failure reliability).

It is also possible to allocate the **Mean Time Between Failure (MTBF)**. We can also **allocate availability goals**, which takes into consideration systems that can be serviced and repaired to mitigate failure.

There are lots of other variations to **reliability allocation**. For example, what does the reliability goal need to be for a module that is added to an existing system with known reliability performance?