Let's imagine that the job of a harvester is to use an axe to harvest trees, and the axe will deteriorate over time. Assuming that the following's the expected performance of the axe:

Fully sharp axe(extremely excellent effectiveness and efficiency; ideal defect rates) -

1. 1 tree cut / hour
2. 1 / 20 chance for the tree being cut to be defective(with 0 extra decent tree to be cut for compensation as compensating trees due to negligible damages caused by defects)
3. Expected number of normal trees / tree cut = (20 - 1 = 19) / 20
4. Becomes a somehow sharp axe after 20 trees cut(a fully sharp axe will become a somehow sharp axe rather quickly)

Somehow sharp axe(reasonably high effectiveness and efficiency; acceptable defect rates) -

1. 1 tree cut / 2 hours
2. 1 / 15 chance for the tree being cut to be defective(with 1 extra decent tree to be cut for compensation as compensating trees due to nontrivial but small damages caused by defects)
3. Expected number of normal trees / tree cut = (15 - 1 - 1 = 13) / 15
4. Becomes a somehow dull axe after 80 trees cut(a somehow sharp axe will usually be much more resistant on having its sharpness reduced per tree cut than that of a fully sharp axe)
5. Needs 36 hours of sharpening to become a fully sharp axe(no trees cut during the atomic process)

Somehow dull axe(barely tolerable effectiveness and efficiency; alarming defect rates) -

1. 1 tree cut / 4 hours
2. 1 / 10 chance for the tree being cut to be defective(with 2 extra decent trees to be cut for compensation as compensating trees due to moderate but manageable damages caused by defects)
3. Expected number of normal trees / tree cut = (10 - 1 - 2 = 7) / 10
4. Becomes a fully dull axe after 40 trees cut(a somehow dull axe is just ineffective and inefficient but a fully dull axe is significantly dangerous to use when cutting trees)
5. Needs 12 hours of sharpening to become a somehow sharp axe(no trees cut during the atomic process)

Fully dull axe(ridiculously poor effectiveness and efficiency; obscene defect rates) -

1. 1 tree cut / 8 hours
2. 1 / 5 chance for the tree being cut to be defective(with 3 extra decent trees to be cut for compensation as compensating trees due to severe but partially recoverable damages caused by defects)
3. Expected number of normal trees / tree cut = (5 - 1 - 3 = 1) / 5
4. Becomes an irreversibly broken axe(way beyond repair) after 160 trees cut
5. The harvester will resign if the axe keep being fully dull for 320 hours(no one will be willing to work that dangerously forever)
6. Needs 24 hours of sharpening to become a somehow dull axe(no trees cut during the atomic process)

Now, let's try to come up with some possible work schedules:

Sharpens the axe to be fully sharp as soon as it becomes somehow sharp -

1. Expected to have 19 normal trees and 1 defective tree cut after 1 \* (19 + 1) = 20 hours(simplifying "1 / 20 chance for the tree being cut to be defective" to be "1 defective tree / 20 trees cut")
2. Expected the axe to become somehow sharp now, and become fully sharp again after 36 hours
3. Expected long term throughput to be 19 normal trees / (20 + 36 = 56) hours(around 33.9%)

Sharpens the axe to be somehow sharp as soon as it becomes somehow dull -

1. The initial phase of having the axe being fully sharp's skipped as it won't be repeated
2. Expected to have 68 normal trees, 6 defective trees, and 6 compensating trees cut after 2 \* (68 + 6 + 6) = 160 hours(simplifying "1 / 15 chance for the tree being cut to be defective" to be "1 defective tree / 15 trees cut" and using the worst case)
3. Expected the axe to become somehow dull now, and become somehow sharp again after 12 hours
4. Expected long term throughput to be 68 normal trees / (160 + 12 = 172) hours(around 39.5%)

Sharpens the axe to be somehow dull as soon as it becomes fully dull -

1. The initial phase of having the axe being fully or somehow sharp's skipped as it won't be repeated
2. Expected to have 28 normal trees, 4 defective trees, and 8 compensating trees cut after 4 \* (28 + 4 + 8) = 160 hours(simplifying "1 / 10 chance for the tree being cut to be defective" to be "1 defective tree / 10 trees cut")
3. Expected the axe to become fully dull now, and become somehow dull again after 24 hours
4. Expected long term throughput to be 28 normal trees / (160 + 24 = 184) hours(around 15.2%)

Sharpens the axe to be somehow dull right before the harvester will resign -

1. The initial phase of having the axe being fully or somehow sharp's skipped as it won't be repeated
2. Expected to have 28 normal trees, 4 defective trees, and 8 compensating trees cut after 4 \* (28 + 4 + 8) = 160 hours(simplifying "1 / 10 chance for the tree being cut to be defective" to be "1 defective tree / 10 trees cut") when the axe's somehow dull
3. Expected the axe to become fully dull now, and expected to have 4 normal trees, 8 defective trees, and 24 compensating trees but after 8 \* (4 + 8 + 24) = 288 hours(simplifying "1 / 5 chance for the tree being cut to be defective" to be "1 defective tree / 5 trees cut" and using the worst case) when the axe's fully dull
4. Expected total number of normal trees to be 28 + 4 = 32
5. Expected the axe to become somehow dull again after 24 hours(so the axe remained fully dull for 288 + 24 = 312 hours, which is the maximum before the harvester will resign)
6. Expected long term throughput to be 32 normal trees / (160 + 312 = 472) hours(around 6.7%)

Sharpens the axe to be fully sharp as soon as it becomes somehow dull -

1. Expected total number of normal trees to be 19 + 68 = 87
2. Expected total number of hours to be 56 + 172 = 228 hours
3. Expected long term throughput to be 87 normal trees / 228 hours(around 38.2%)

Sharpens the axe to be fully sharp as soon as it becomes fully dull -

1. Expected total number of normal trees to be 19 + 68 + 28 = 115
2. Expected total number of hours to be 56 + 172 + 184 = 412 hours
3. Expected long term throughput to be 115 normal trees / 412 hours(around 27.9%)

Sharpens the axe to be fully sharp right before the harvester will resign -

1. Expected total number of normal trees to be 19 + 68 + 32 = 119
2. Expected total number of hours to be 56 + 172 + 472 = 700 hours
3. Expected long term throughput to be 119 normal trees / 700 hours(17%)

Sharpens the axe to be somehow sharp as soon as it becomes fully dull -

1. Expected total number of normal trees to be 68 + 28 = 96
2. Expected total number of hours to be 172 + 184 = 356 hours
3. Expected long term throughput to be 96 normal trees / 356 hours(around 26.9%)

Sharpens the axe to be somehow sharp right before the harvester will resign -

1. Expected total number of normal trees to be 68 + 32 = 100
2. Expected total number of hours to be 172 + 472 = 644 hours
3. Expected long term throughput to be 100 normal trees / 644 hours(around 15.5%)

**So, while these work schedules clearly show that sharpening the axe's important to maintain effective and efficient long term throughput, trying to keep it to be always fully sharp is certainly going overboard(33.9% throughput), when being somehow sharp is already enough(39.5% throughput).**

Then why some bosses don't let the harvester sharpen the axe even when it's somehow or even fully dull? **Because sometimes, a certain amount of normal trees have to be acquired within a set amount of time.**

Let's say that the axe has become from fully sharp to just somehow dull, so there should be 87 normal trees cut after 180 hours, netting the short term throughput of 48.3%.

But then some emergencies just come, and 3 extra normal trees need to be delivered within 16 hours no matter what, whereas compensating trees can be delivered later in the case of having defective trees.

In this case, there won't be enough time to sharpen the axe to be even just somehow sharp, because even in the best case, it'd cost 12 + 2 \* 3 = 18 hours.

On the other hand, even if there's 1 defective tree from using the somehow dull axe within that 16 hours, the harvester will still barely make it on time, because the chance of having 2 defective trees is (1 / 10) ^ 2 = 1 / 100, which is low enough to be neglected for now, and as compensatory trees can be delivered later even if there's 1 defective tree, the harvester will be able to deliver 3 normal trees.

**In reality,** [**crunch modes**](https://wiki.c2.com/?CrunchMode) **like this will happen occasionally, and most harvesters will likely understand that it's probably inevitable eventually, so as long as these crunch modes won't last for too long, it's still practical to work under such circumstances once in a while, because it's just being reasonably pragmatic.**

**However, in supposedly exceptional cases, the situation's so extreme that, when the harvester's about to sharpen the axe, the boss constantly requests that another tree must be acquired as soon as possible, causing the harvester to never have time to sharpen the axe for a long time, thus having to work more and more ineffectively and inefficiently in the long term.**

In the case of a somehow dull axe, 12 hours are needed to sharpen it to be somehow sharp, whereas another tree's expected to be acquired within 4 hours, because the chance of having a defective tree cut is 1 / 10, which can be considered small enough to take the risk, and the expected number of normal trees over all trees being cut is 7 of out 10 for a somehow dull axe, whereas 12 hours is enough to cut 3 trees by using such an axe, so at least 2 normal trees can be expected within this period.

If this continues, eventually the axe will become fully dull, and 24 hours will be needed to sharpen it to be somehow dull, whereas another tree's expected to be acquired within 8 hours, because the chance of having a defective tree is 1 / 5, which can still be considered controllable to take the risk, especially with an optimistic estimation.

While the expected number of normal trees over all trees being cut is 1 of out 5 for a fully dull axe, whereas 24 hours is just enough to cut 3 trees by using such an axe, meaning that the harvester's not expected to make it normally, in practice, the boss will usually unknowingly apply [optimism bias](https://en.wikipedia.org/wiki/Optimism_bias)(at least until it no longer works) by thinking that there will be no defective trees when just another tree's to be cut, so the harvester will still be forced to continue cutting trees, despite the fact that the axe should be sharpened as soon as possible even when just considering the short term.

Also, if the boss can readily replace the current harvester with a new one immediately, the boss will rather let the current harvester resign than letting that harvester sharpening the axe to be at least somehow dull, because to the boss, it's always emergencies after emergencies, meaning that the short term's constantly so dire that there's just no room to even consider the long term at all.

But why such an undesirable situation will be reached? **Other than** [**extreme and rare misfortunes**](https://en.wikipedia.org/wiki/Black_swan_theory)**, it's usually due to overly optimistic work schedules not seriously taking the existence of defective and compensatory trees, and the importance of the sharpness of the axe and the need of sharpening the axe into the account, meaning that such unrealistic work schedules are essentially linear(e.g.: if one can cut 10 trees on day one, then he/she can cut 1000 trees on day 100), which is obviously simplistic to the extreme.**

Occasionally, it can also be because of the inherent risks of sharpening the axe - **Sometimes the axe won't be actually sharpened after spending 12, 24 or 36 hours, and while it's extraordinary, the axe might be actually even more dull than before, and most importantly, usually the boss can't directly judge the sharpness of the axe, meaning that it's generally hard for that boss to judge the** [**ROI**](https://en.wikipedia.org/wiki/Return_on_investment) **of sharpening the axe with various sharpness before sharpening, and it's only normal for the boss to distrust what can't be measured objectively by him/herself(on the other hand, normal, defective and compensatory trees are objectively measurable, so the boss will of course emphasize on these** [**KPIs**](https://en.wikipedia.org/wiki/Performance_indicator)**), especially for those having been opting for** [**linear thinking**](https://www.jodymichael.com/blog/linear-thinkers-overcoming-leadership-challenges/)**.**

Of course, the whole axe cutting tree model is highly simplified, at least because:

1. The axe sharpness deterioration isn't a step-wise function(an axe becomes from having a discrete level of sharpness to another such level after cutting a set number of trees), but rather a continuous one(gradual degrading over time) with some variations on the number of trees cut, meaning that when to sharpen the axe in the real world isn't as clear cut as that in the aforementioned model(usually it's when the harvester starts feeling the pain, ineffectiveness and inefficiency of using the axe due to unsatisfactory sharpness, and these feeling has last for a while)
2. Not all normal trees are equal, not all defective trees are equal, and not all compensatory trees are equal(these complications are intentionally simplified in this model because these complexities are hardly measurable)
3. The whole model doesn't take the morale of the harvester into account, except the obvious one that that harvester will resign for using a fully dull axe for too long(but the importance of sharpening the axe will only increase if morale has to be considered as well)
4. In some cases, even when the axe's not fully dull, it's already impossible to sharpen it to be fully or even just somehow sharp(and in really extreme cases, the whole axe can just suddenly break altogether for no apparent reason)

Nevertheless, this model should still serve its purpose of making this point across - **There's isn't always an universal answer to when to sharpen the axe to reach which level of sharpness, because these questions involve calculations of concrete details(including those critical parts that can't be quantified) on a case-by-case basis, but the point remains that the importance of sharpening the axe should never be underestimated.**

When it comes to professional software engineering:

1. **The normal trees are like needed features that work well enough**
2. **The defective trees are like nontrivial bugs that must be fixed as soon as possible(In general, the worse the code quality of the codebase is, the higher the chance to produce more bugs, produce bugs being more severe, and the more the time's needed to fix each bug with the same severity - More severe bugs generally cost more efforts to fix in the same codebase)**
3. **The compensatory trees are like extra outputs for fixing those bugs and repairing the damages caused by them**
4. **The axe is like the codebase that's supposed to deliver the needed features(actually, the axe can also be like those software engineers themselves, when the topic involved is software engineering team management rather than just refactoring)**
5. **Sharpening the axe is like refactoring(or in the case of the axe referring to software engineers, sharpening the axe can be like letting them to have some vacations to recover from** [**burnouts**](https://en.wikipedia.org/wiki/Occupational_burnout)**)**
6. A fully sharp axe is like a codebase suffering from the [gold plating](https://exceptionnotfound.net/gold-plating-the-daily-software-anti-pattern/) anti pattern on the code quality aspect([diminishing returns](https://en.wikipedia.org/wiki/Diminishing_returns) applies to code qualities as well), as if those professional software engineers can't even withstand a tiny amount of [technical debt](https://en.wikipedia.org/wiki/Technical_debt). On the good side, such an ideal codebase is the most unlikely to produce nontrivial bugs, and even when it does, they're most likely fixed with almost no extra efforts needed, because they're usually found way before going into production, and the test suite will point straight to their root causes.
7. A somehow sharp axe is like a codebase with more than satisfactory code qualities, but not to the point of investing too much on this regard(and the technical debt is still doing more good than harm due to its amount under moderation). Such a practically good codebase is still a bit unlikely to produce nontrivial bugs regularly, but it does have a small chance to let some of them leak into production, causing a mild amount of extra efforts to be needed to fix the bugs and repair the damages caused by them.
8. A somehow dull axe is like a codebase with undesirable code qualities, but it's still an indeed workable codebase(although it's still quite painful to work with) with a worrying yet payable amount of technical debt. Undesirable yet working codebases like this probably has a significant chance to produce nontrivial bugs frequently, and a significant chance for quite some of them to leak into production, causing a rather significant amount of extra efforts to be needed to fix the bugs and repair the damages caused by them.
9. A fully dull axe is like a unworkable codebase where it must be refactored as soon as possible, because even senior professional software engineers can easily create more severe bugs than needed features with such a codebase(actually they'll be more and more inclined to rewrite the codebase the longer it's not refactored), [causing their productivity to be even negative](https://www.mehdi-khalili.com/dealing-with-net-negative-producing-programmers) in the worst cases. An effectively broken codebase like this is guaranteed to has a huge chance to produce nontrivial bugs all the time, and nearly all of them will leak into production, causing an insane amount of extra efforts to be needed to fix the bugs and repair the damages caused by them(so the professionals will be always fixing bugs instead of delivering features), provided that these recovery moves can be successful at all.
10. A broken axe is like a codebase being totally [technical bankrupt](https://diginomica.com/youve-heard-about-technical-debt-technical-bankruptcy-business-killer), where the only way out is to completely rewrite the whole thing from scratch, because no one can fathom a thing in that codebase at that point, and sticking to such a codebase is undoubtedly a [sunk cost fallacy](https://en.wikipedia.org/wiki/Sunk_cost#Fallacy_effect).
11. While a codebase with overly ideal code qualities can deliver the needed features in the most effective and efficient ways possible as long as the codebase remains in this state, in practice the codebase will quickly degrade from such an ideal state to a more practical state where the code qualities are still high(on the other hand, going back to this state is very costly in general no matter how effective and efficient the refactoring is), because this state is essentially [mysophobia](https://en.wikipedia.org/wiki/Mysophobia) in terms of code qualities.
12. On the other hand, a codebase with reasonably high code qualities can be rather resistant from code quality deterioration(but far from 100% resistant of course), especially when the professional software engineers are disciplined, experienced and qualified, because degrading code qualities for such codebases are normally due to quick but dirty hacks, which shouldn't be frequently needed for senior professional software engineers.

**To summarize, a senior professional software engineer should strive to keep the codebase to have a reasonably high code quality, but not to the point of not even having good technical debts, and when the codebase has eventually degraded to have just barely tolerable code quality, it's time to refactor it to become having very satisfactory, but not overly ideal, code quality again, except in the case of occasional crunch modes, where even a disciplined, experienced and qualified expert will have to get the hands dirty once in a while on the still workable codebase but with temporarily unacceptable code quality, just that such crunch modes should be ended as soon as possible, which should be feasible with a well-established work schedule.**