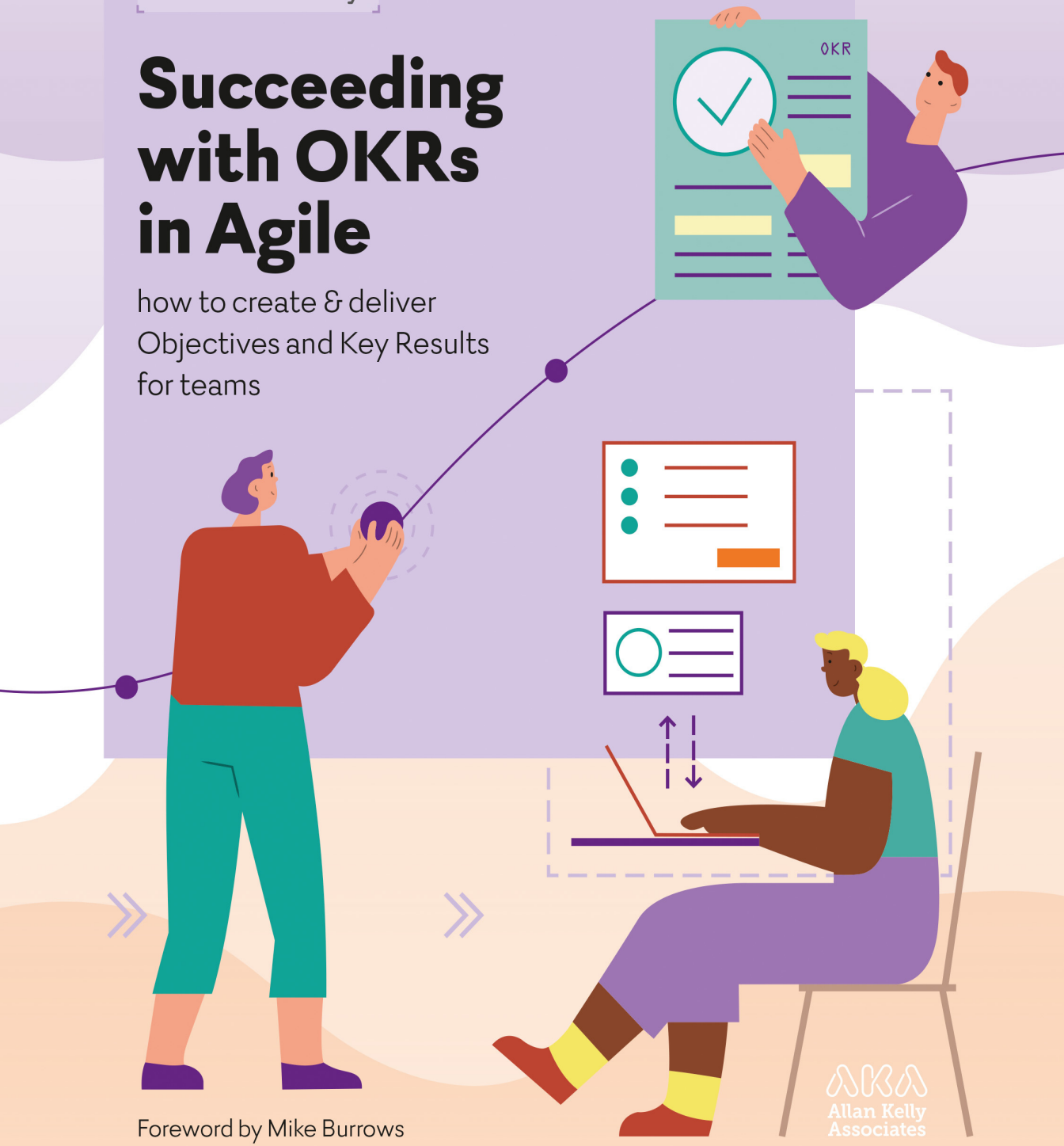
Allan Kelly

Succeeding with OKRs in Agile

how to create & deliver Objectives and Key Results for teams



Foreword by Mike Burrows

Allan Kelly Associates

Succeeding with OKRs in Agile

How to create & deliver Objectives Key Results for teams

Allan Kelly

This book is for sale at http://leanpub.com/agileokrs

This version was published on 2021-03-25

ISBN 978-1-912832-09-5

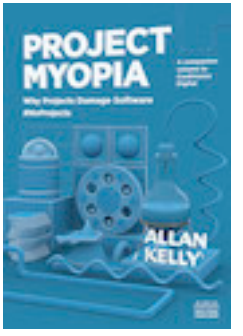
This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Free book when you subscribe

Be the first to hear of Allan Kelly's latest articles, books, events and insights, and receive discounts on workshops - subscribe now[1].

Plus receive a free ebook - currently *Project Myopia* - when subscribing[2].



Project Myopia

[1] http://allankelly.net/newsletter
[2] http://allankelly.net/newsletter

Foreword

What a timely book! It comes at a time of global pandemic, with severe consequences not only for health and prosperity, but also for how we relate to each other and how we work. To the agile context of this book, a truly celebration-worthy 20th anniversary comes amid some serious self-scrutiny.

Agile's issues aren't so deep-rooted that they aren't fixable, but they are serious enough to need to be confronted before the disillusionment turns into crisis. Summarizing, I identify three key problems.

Problem one: orientation. We're seeing a divergence in the agile community represented, not so much by framework choices, but by what might be called 'drive' or 'orientation'. It's between heavily backlog-driven styles on one hand and deliberately outcome-oriented approaches on the other. In the worst examples of the former, teams are ploughing through backlogs of requirements with minimal opportunity for feedback, the team's experience and the customer's eventual results being no better than mediocre. Hardly agile at all by any useful definition, but with enough of its trappings to cause lasting damage to agile's reputation.

Problem two: autonomy. This problem is partly a consequence of the first, and it's a growing tension between team autonomy – something fundamental to agile – and strategy. As I have written elsewhere, it's a funny kind of autonomy when strategy is something that happens to you, but when all of your work is represented by a backlog over which you have little control this is very much what it feels like. Similar feelings of disempowerment and disengagement are triggered when ways of working move out of the team's control, the oh-so-ironic result of agile being implemented through the traditional rollout project.

Problem three: organization. The temptation to scale up agile processes is ever-present, but to do so without paying careful attention to other crucial aspects of organization design is fraught with difficulty. One specific manifestation of this problem is strategy dressed up in agile terms but still formalized hierarchically as work breakdown structures. This practice not only amplifies problems one and two, it adds a third: the inability of teams, departments, business units and management each to express themselves in terms appropriate to their respective domains. However elegant and convenient these structures might appear on paper, in practice they turn out to be unwieldy and oppressive, massive impediments to the kind

of rapid feedback, learning and adaptation that businesses seek when they are encouraged down this path in the first place.

This book is unique in how directly it demonstrates how Objectives and Key Results (OKR) address these very real problems. Instead of teams ploughing through backlogs of requirements, they pursue meaningful objectives one key result at a time, an approach that is naturally outcome-oriented, iterative and adaptive. Instead of strategy being imposed on them from the outside, teams maintain in the form of OKRs their best understanding of how to meet the needs of their customers and other stakeholders. And instead of cascading hierarchies of objectives, mutual accountability and transparency act as enablers of self-organization and learning at every scale.

So it's all rosy then? Well, not so fast. OKR is not a million miles away from top-down Management by Objectives (MBO), a framework so plausible and yet so prone to devastating dysfunction that Peter Drucker, its highly respected creator, disowned it. It turns out that OKR is very much like agile; approach it from the wrong direction and some seriously bad things can happen.

Allan's treatment of OKR in this book isn't just enjoyable and practical (and I assure you that it is both), it stands for MBO's polar opposite. OKRs here aren't top-down, they are bottom-up, starting at team level and managed within an alignment process. They're expressed in each team's own words, the needs of others taken respectfully into account. Most dramatically, they change fundamentally how teams regard their backlogs. I shall leave it to Allan to explain that last one, but let me say now that as a long-time and vocal champion myself of outcome-orientation, I applaud his boldness.

There is more than one way to do agile, more than one way to do OKR, and multiple ways to combine them. Some of those combinations have enormous potential, and to anyone interested in exploring this exciting space I wholeheartedly recommend Allan's book. Well done my friend!

Mike Burrows January 2021, Chesterfield, Derbyshire UK

Founder, Agendashift agendashift.com[3]

Author, Agendashift: Outcome-oriented change and continuous transformation[4] (2nd edition March 2021)

Author, Right to Left: The digital leader's guide to Lean and Agile[5] (2019, audiobook 2020)

Author, Kanban from the Inside[6] (2014)

[3] https://www.agendashift.com/
[4] https://www.agendashift.com/books/agendashift-2nd-edition
[5] https://amzn.to/2NRZSoP
[6] https://amzn.to/3tqzsuy

Preface

This book is the product of Covid-19. What started life as some notes to myself on experiences using OKRs blossomed into a book during the first few weeks of England's first lockdown. Therefore, while one does not wish to dwell on Covid, it seems justifiable to write a few words about the lessons of Covid as they pertain to OKRs, and consider how OKRs can assist in this new world.

I am sure there were no risk logs in January 2020 that read 'World pandemic: international travel suspended; workers confined to home.' Few organizations were prepared for what happened in March, but few disruptions demonstrate the value of agility so clearly. Indeed, even those would not consider themselves agile had to reshape their working environment literally overnight.

Urgency begat purpose: 'stay alive, stay productive'. Purpose begat focus: everything nonessential was pushed to one side, while entire companies pivoted to home working. Success was measured not in money spent, not in the time it took, but one single outcome: survival – the ability to continue operations.

Purpose, focus, outcomes: those three nouns could themselves summarize OKRs. When setting OKRs teams need to draw on their purpose, their raison d'être – their reason for being.

Once set, OKRs should be the focus of all work: *don't get out of bed in the morning if it doesn't contribute towards the OKRs.* OK, that's a slight exaggeration.

Focus, however, is only a means to an end, an outcome. More specifically, an outcome that moves the whole team – indeed the whole organization – along the path to fulfilling its purpose. Thus, while that purpose is front of mind when setting OKRs, outcome is central when judging success or failure.

> *Did the outcomes advance our purpose during the period of the OKRs?*

Whether OKRs are met or not is almost secondary. OKRs are a hypothesis for the coming quarter, a best effort guess at what will advance purpose. At the end of the quarter, review and adapt – what could be more agile?

While the response to Covid-19 has demonstrated the key qualities OKRs espouse, that response itself is the result of the digitization of human life and world commerce. What if Covid-19 had struck 20 years ago? Back in 2000 the world could not have locked down and shifted to home working the way it did.

Cell phones, the internet, Amazon and UPS all existed then, but not at the same scale, omnipresence or power. Back in 2000 few people had broadband internet, and online shopping – from home at least – was slow. Except for a lucky few fast internet and video conferencing still only existed in the office. Had Covid struck in 2000 the economic damage and death toll would have been far higher.

And 20 years earlier still, 1980? Locking down for three months would have killed the economy. NTT launched the first cell phone in 1979, ARPANET became the internet in 1981, the same year IBM launched the PC, and TCP/IP was standardized the following year. A response to Covid-1980 would have looked a lot more like Spanish Flu 1918.

Digital made Covid-19 possible. Or rather, digital technology allowed the response that followed: the economy continued from home. The question nobody yet knows the answer to is: *to what degree will life return to pre-Covid ways?*

While each of us wants this awful illness gone and our old lives back, positive things have arisen from the crisis – this book for one. It is hard to see employers and employees reverting to office working without also supporting home working; some aspects of telemedicine will stay; Amazon and Netflix will retain new customers.

Depending on which commentators you listen to, Covid has accelerated the digital revolution by between five and ten years. It is clearer than ever that digital is here to stay. Both OKRs and agile have a role to play in this new world.

To start with, agile is the process change that accompanies digital. Agile is both the result of early digitalization – programmers received digital tools first and then invented agile – and agile is the way of working that accompanies digital tools. Nobody in their right mind tries to become a digital enterprise by writing a requirements document and building a Gantt chart. Digital demands agile.

Purpose, focus and outcomes become even more important when teams are distributed. Managers can no longer practice *management by walking around*, or even by watching staff. Managers must look at outcomes. Meanwhile, employees must redouble efforts to focus if working from a bedroom, especially with home-schooled children running wild! And everyone must share a purpose.

Digitalization means businesses are digital – or at least growth businesses are digitally driven – which means 'the business' is intimately coupled to the software. *The business is technology,*

and the technology is the business.

For a digital business to change and grow, so too must its software. The idea that a software project can ever be 'finished' is a fantasy. When the software stops changing, so too does the business.

Businesses run on software products, not projects. This means the management model must change. IT is no longer a cost centre *over there* that 'does projects' that are always delivered late. Digital is as fundamental as accounting or marketing, and digital work is continuous.

There are those who see OKRs as a reinvention of projects. They see them as a command-and-control tool used by managers, they seem them as a form of requirements document and they see the same goal-displacement failures.

I don't see them this way. OKRs fit well with the continuous agenda[7]. This book sets out that alternative vision.

OKRs provide a link to the 'bigger picture' – the purpose, the mission: they step up from sprint-sprint-sprint. Managers have influence in deciding what will be worked on, as they are entitled too: they are also stakeholders. Managers play a role in delivering OKRs, but managers are skilled in managing. None of that means managers can or should be using OKRs to boss a team about. Rather, OKRs are a mechanism for teams and managers to co-create shared goals that deliver beneficial outcomes.

During the 20 years since the agile manifesto many agile advocates, myself included, have at times felt as if agile has lost its way. There are places today that claim to work 'agile', but when old hands like me look at them they seem to be doing some watered-down version of agile that lacks any ambition to be better.

OKRs have the potential to reawaken the early ambition and drive inherent in agile. This time managers can join in too, not as obstacles to change, or change drivers, but as partners focused on the same outcomes for a greater purpose.

Allan Kelly, London, December 2020.

[7] Allan Kelly, *Continuous Digital*, 2018

Short quick lessons

Bottom up

Don’t impose OKRs from above. Don’t set OKRs top-down.

Do set OKRs bottom up. Allow each team to set their own OKRs to meet bigger goals.

Do let OKRs trickle up from the bottom.

Leaders should describe the ultimate goal, paint a picture and sketch out the future they want to make happen. Then let teams decide how they can make that future happen.

Every senior leader and layer of the organization has a duty to make those dependent on them successful.

Organization

Do make everything subservient to OKRs. Throw away the backlog.

Don’t attach names to specific objectives or key results: OKRs are a team sport, not a list of an individual’s tasks.

Don’t manage dependencies. Do eliminate dependencies.

Rather than create a complex OKR-setting process to manage interdependencies, seek to remove dependencies. Enhance independence even at the cost of redundancy and duplication, strip away insulation layers and help connect teams with customers. In other words, increase cohesion and reduce coupling.

True north

Do use OKRs to guide you and fight to stay on course.

Don’t change or abandon OKRs during a quarter without a fight.

But…

Don’t stick blindly to OKRs as the world around changes.

Do talk to the whole team and get agreement before going *off-piste*.

If you find that your goals regularly change over a quarter, try working in one-month cycles. If the rate of change is too much, and is valuable, then write OKRs that demonstrate the value of continually changing goals.

Remember that if you always prioritize firefighting over pursuing goals, you will only ever be a firefighter. Firefighting is a very respectable profession but not everyone is, or should be, a firefighter.

Leaders

Do build *psychological safety* and *make failure an option*: only when it is safe to try, fail and try again will people be ambitious.

Do make it completely clear what the organization's priorities are.

Do make yourself available to teams, to answer their questions and answer them quickly.

Remember: teams only have 12 weeks to deliver OKRs, so don't dally.

Do make resources available to the team or explain the constraints they must work within.

Do make clear the level of OKR achievement teams should be aiming for. If it is 70%, make that clear. If it is 80% or 60%, then make it even clearer.

Remember: OKRs belong to the team; you cannot tell them what to put in their OKRs.

Reviewing

Do practice *tough love* when reviewing OKRs.

Openly acknowledge you are doing so and recognize the need for psychological safety in the review process.

Ask questions such as:

- How does this create value?
- How will this be measured?
- Are these goals ambitious enough?
- Are the goals too ambitious?
- Are OKRs proving useful? Or are they getting in the way of real work?

Watch for signs that the team fear failure and lack psychological safety.

Team

Do make the team responsible for setting their own OKRs and delivering them.

Within the team Product Owners are first among equals when setting priorities: their work, skills and experience gives them insights into what customers want and value.

Teams which consistently achieve very high levels of OKR completion, or very low levels (say above 90% and below 50% respectively), deserve attention.

- Regularly hitting 90%+ of OKRs might lack ambition or, more likely, fear failure.
- Regularly missing OKRs by a wide margin might be a sign of over-ambition or failure to focus. More likely it is a sign of leadership failure or an organizational structure that does not adequately support the team.

Money

Do not link OKRs to bonuses and renumeration.

Just don't.

I Why OKRs

One's philosophy is not best expressed in words; it is expressed in the choices one makes. Eleanor Roosevelt, political figure, diplomat and activist, 1884–1962

1. Introducing OKR

> *Simple can be harder than complex: you have to work hard to get your thinking clean to make it simple. But it's worth it in the end because once you get there, you can move mountains.* Steve Jobs, 1955–2011, cofounder and CEO Apple Computer

OKRs = *Objectives* and *key results*: obvious, perhaps.

OKRs are about goals. Objectives are big goals; key results are smaller goals that build towards the objective. The question is, how ambitious do you want to be?

Make your goals big and ambitious and you might miss. Make your goals small and easily achievable and you will hit them, but will they be as satisfying? Satisfying to you? Satisfying to your organization? Its a risk-reward calculation: you decide.

Goals bring focus, and focus is powerful. But focus also means blinkered vision, which carries dangers – but if we aren't blinkered we may be overwhelmed. Software engineers might recognize this as abstraction.

> *The essential characteristics of an object... the process of focusing upon the essential characteristics of an object.* Grady Booch[1]

> *An abstraction that is appropriate for a given purpose is easier to study than the actual system because it omits details that are not relevant for that purpose.* Britton, Parker and Parnas[2]

Engineers may think of OKRs as an abstraction of the desired outcome to be delivered by the end of the quarter. That outcome is described in terms that speak to the customer and the benefit to be delivered. The engineering detail, the implementation detail, are hidden behind the abstract interface.

As with software design there are different ways of approaching the same thing: each has its own benefits and trade-offs. There may not be an obvious answer, but it is critical that everyone shares the same abstraction.

---

[1] Grady Booch, *Object-oriented analysis and design*, 1994

[2] Kathryn Heninger Britton, R Alan Parker, David L Parnas, *A procedure for designing abstract interfaces for device interface modules*, ICSE '81: Proceedings of the 5th international conference on Software engineering, March 1981

I sometimes think of OKRs as 'Test-Driven Management'. Decide what you want (the objective), next set a series of acceptance criteria: *key results*. Now get on and develop. Don't consider yourself done until you can pass the tests and meet the objectives.

When the acceptance tests are known, engineers have a wide degree of latitude in deciding how to meet their criteria. In doing so they will use their professional judgement and experience. They will also be constrained by the time and resources available: the existing products and technology will further bound a solution.

1.1 Dissecting OKRs

An objective is something you and your team wish to achieve. That objective is a goal to be achieved, something to aim for. It might be a mission in itself, or it might be part of a larger mission or some other 'higher purpose'.[3] The mission might be your product, a business initiative or some endeavor to help a client. Whatever it is, today's objective requires some significant work.

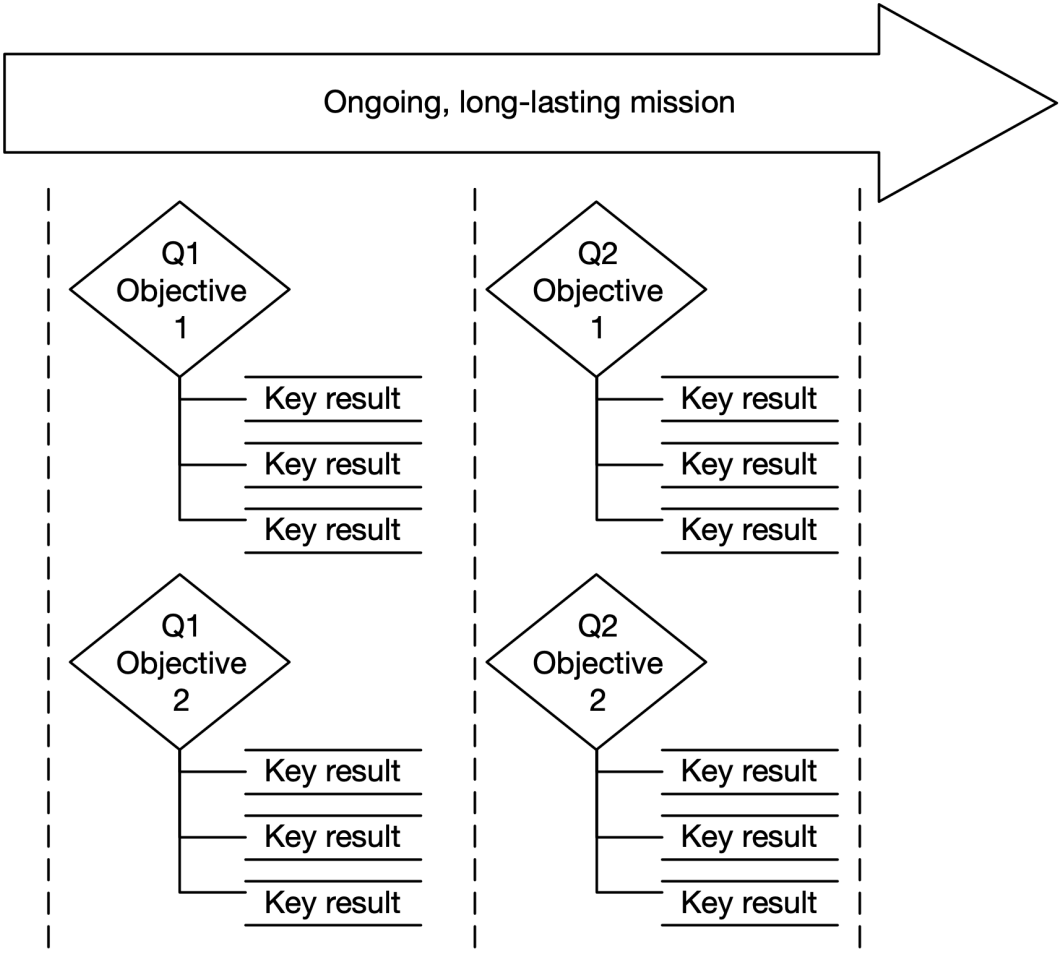
Key results are the important things that build towards that objective. Milestones, if you like, but I like to think they are more than just milestones. When I think of milestones I think of markers along a route. Such milestones don't have value in and of themselves; unless a milestone represents some tangible outcome, reaching it delivers little more than the right to say "We've reached the milestone".

Good OKRs are outcome-focused. OKRs are not about measuring progress towards a goal. Nor are they about ticking off work items on a manifest. OKRs are about delivering outcomes that add value. That's one reason why they are a good fit with agile.

Each objective will have several key results. Each result should be useful in and of itself. Achieving a key result should deliver some benefit – some value – to someone.

An objective should have its own *wholeness* that is more than the sum of its parts – the key results. Achieving the key results builds towards the objective, but the whole thing, the whole objective, should create more value than simply the value of the key results added together.

[3] See my earlier book *Continuous Digital* (2018) for a fuller discussion.



Missions are long-lasting, OKRs are reset every quarter

While the mission is largely immutable, OKRs get reviewed at the end of each quarter and new ones created for the next quarter. Some may need to flow over from one quarter to another, but each OKR-setting session should start with a blank sheet. If something is worth continuing into the next quarter it's because there is more value to be gained, not because something wasn't finished or sunk costs incurred.

Teams pursue several objectives over the course of a quarter, and each objective has several key results – hence objectives (plural) and key results (plural), although for a really focused team it could be one objective (singular) and key results (plural).

1.2 OKRs and agile

Those schooled in agile may say "Oh, an objective is an epic, and the key results are stories", but OKRs are more than that. OKRs are not a mini-backlog, rather they are a machine for generating stories. OKRs are more akin to sprint goals; indeed each key result may be a sprint goal itself.

So throw away your epics – they aren't a perfect fit for objectives and they don't add much anyway. The objective fills the same role as an epic: *the big thing to aim at.*

Then, rather than seeing key results as stories, see them as targets. Ask yourself: *what do we need to do to move towards that target?* And ask again, and again. Every time you need to decide the next thing to work on, go back to the OKRs and ask the question afresh.

I'd go as far as to throw away any backlog and drive all work from the OKR story machine.

While a quarterly cycle is standard, you might choose to work on some other cycle duration. It's up to you, but working in quarters has a good rhythm.

Some see the quarterly OKR review and setting as a giant sprint. While there are similarities in the routine, the logic is different. Sprint planning is very focused on immediate action and delivering in the next few days. OKR-setting should be more thoughtful and customer (demand) focused.

1.3 Think broadly, execute narrowly

OKRs need to be measurable. While this is good because it sharpens one's thinking and enforces honesty, it is also bad because hard numbers can get in the way of big thinking. Targets can blind one to unintended consequences and side effects incurred in meeting the target; the numbers used in targets have a nasty habit of changing in unpredictable ways.

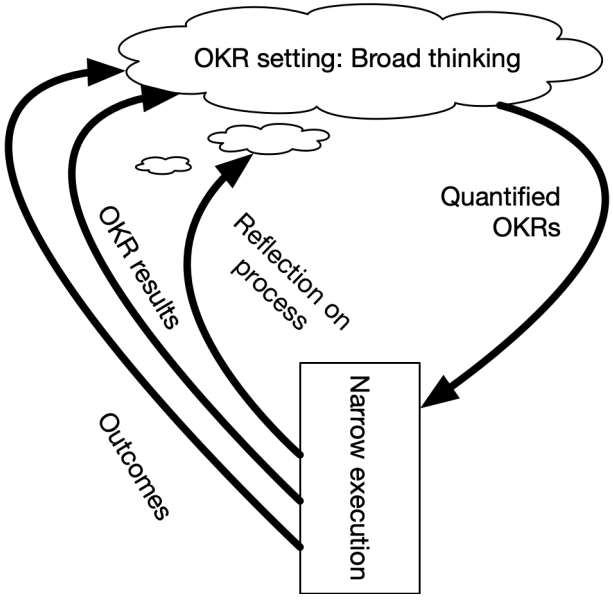
The laser-like focus of delivering OKRs needs moderating with expansive and considered thinking during OKR-setting. Teams should alternate between reflection and broad thinking during OKR review and setting and really focused actions during delivery. The former should last hours, the latter months.

When it comes time to set OKRs again the results of the previous OKRs will inform thinking, so too should thoughts on the OKR process. Were the OKRs too vague? Too strict? Too detailed? Did enough, or too many, conversations happen beforehand?

The most perfect execution in the world is nothing if it aims for the wrong target. Equally, the most perfectly defined target is worthless if setting it uses all the time and strips away risk and motivation.

Despite the hard thinking that goes into OKR-setting, the real success of OKRs is not whether any particular objective or key result has been achieved. OKRs are *transient objects* that serve to focus thinking and work. The real benefit is in the outcomes delivered.

The ultimate objective of any OKR is to produce an outcome that creates value and benefit to customers, users and other stakeholders.



Iterate between broad thinking and narrow execution

Iterate

Think broadly for a short window of time to set OKRs.

Work narrowly for a far longer period to deliver OKRs.

Default to staying focused on OKRs during delivery, but be prepared to fight fires if need be. There is no point in delivering against targets if the world has burned down. If all you ever do is fight fires, then you will only ever be a firefighter.

1.4 Ambition over estimation

Unlike burn-down charts, velocity and story points, OKRs are not for estimation or forecasting. I'd advise against estimating any objective or key result, but rather to challenge yourself. The aim of OKRs is not to do everything, rather the aim is to be ambitious, to be prepared to push further. For that reason, OKRs shouldn't be used to benchmark teams or individuals.

Teams are not normally expected to complete 100% of their OKRs – 70% is more common. Hitting 100% is easy if the team sets easy goals. With OKRs teams are encouraged to aim high – not impossibly high, but high enough to be challenged. If teams are meeting 100% then maybe they are not aiming high enough?

Each of us want to do well and achieve 100%, in many places anything less than 100% looks like failure. Therefore it is important that leaders at all levels provide an environment in which it is safe to fail – that is, provide *psychological safety*.

Benchmarking OKRs against other teams, attaching money to OKRs, attaching blame for missed OKRs, linking performance reviews or promotion to OKRs will all destroy that safety.

Psychological safety

Psychological safety is broadly defined as a climate in which people are comfortable expressing and being themselves. More specifically, when people have psychological safety at work, they feel comfortable sharing concerns and mistakes without fear of embarrassment or retribution. They are confident that they can speak up and won't be humiliated, ignored or blamed. They know they can ask questions when they are unsure about something. They tend to trust and respect their colleagues. When a work environment has reasonably high psychological safety, good things happen: mistakes are reported quickly so that prompt corrective action can be taken; seamless coordination across groups or departments is enabled, and potentially game-changing ideas for innovation are shared. In short, psychological safety is a crucial source of value creation in organizations operating in a complex, changing environment. Amy C Edmondson, *The Fearless Organization*, 2019

When using a burn-down chart the implicit goal is to reach zero. In the traditional project model the aim is to do everything asked for, even if that needs more time. With OKRs, in contrast, achieving 100% of the targets is failure: achieving every key result and every objective suggests the team was not ambitious enough.

The thinking behind OKRs is that a team that aims high and only achieves three-quarters of their target will still deliver more benefit than a team that aims comfortably low and achieves everything. Therefore it is wrong to judge teams and individuals on how many OKRs they achieve.

Instead, when it comes to assessing performance, look at the outcome, look at the value delivered, and how things are different compared to three months ago. Ask yourself: *How is the product, the company, the world, a better place for what the team has done?*

For that reason, OKRs are not going to sit comfortably with those who want certainty. Nor are OKRs going to sit well with those who want to tell others what to do. OKRs are set by the same people who are going to deliver them. OKRs are not about top-down control, they are about bottom-up engagement.

Those at the top can set the final destination, give some directions and paint a picture of the promised land, but it is those who are making the journey that get to decide on the means of transport and the route. OKRs are a permission giver, not a control rod.

Best within constraints

The aim is seldom to deliver the best-ever widget.

The aim is to deliver the best possible widget within the constraints.

Constraints come in many forms. Time, people and resources are the most common, and these can usually be summed up by money.

Time is always limited; the sprint and OKR quarter impose useful breakpoints. People are limited to what you have: you may get more in time, but right now you have what you have.

Solving problems within constraints is what engineers do.

2. Why use OKRs?

2.1 Mid-term planning

2.2 Test Driven OKRs

2.3 Communication

2.4 Warning

2.5 Summary

3. Focus

3.1 OKRs create focus

3.2 Summary

4. OKR History

5. Outcomes, value and benefits

5.1 Business benefit and value

5.2 Value

5.3 Pieconomics

5.4 Summary

II Writing OKRs

“Determine that the thing can and shall be done, and then we shall find the way.”

“A goal properly set is halfway reached.”

Abraham Lincoln, 1809–1865, President of the United States

6. Writing OKRs

"Less is more"

"God is in the details"

Ludwig Mies van der Rohe, architect, 1886–1969

Quarterly writing of new OKRs is primarily a strategy question: *what are the strategic priorities for the next quarter?* It requires broad thinking. What does the team aim to do? What targets will the team set for itself? More importantly: what will the team not do?

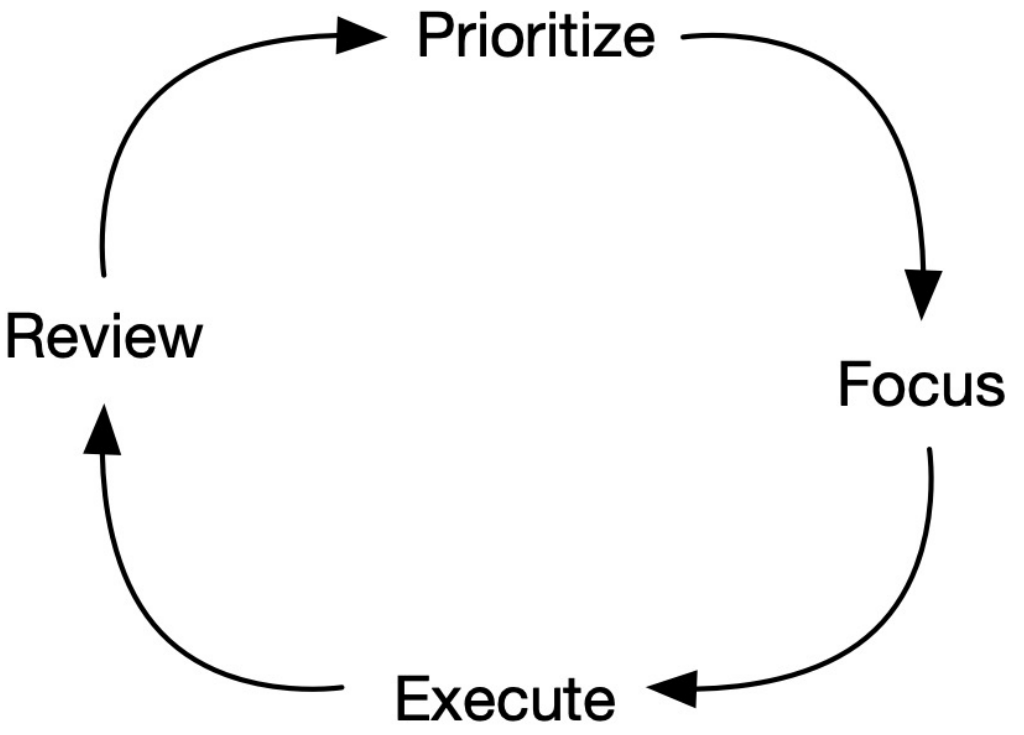
Delivering those goals an operational issue. It demands narrow, focused, action – prioritization.

Prioritization is not just about deciding what to do, it is also about deciding what not to do. In writing OKRs for the coming quarter a team is consciously deciding what it will aim for. Perhaps more importantly, the team is also saying what it will not aim for. Everything that is not in the OKRs is, by definition, lower priority.

If one is totally honest, for many – if not most – teams, if something is not in the OKRs for a quarter is stands little if any chance of being done. Teams are expected to give OKRs the best shot possible. Work that is not in an OKR displaces work that is.

This can be a hard pill to swallow, but experience shows that it is immensely powerful. It is a lesson that is learned over and over again. Whether it be Peter Drucker, agile or OKRs, the message is the same: prioritize, focus, execute. This author relearns the same lesson several times a year.

Blinkered servitude to goals can be as damaging as the randomness of no goals and no shared aims. OKRs mitigate this danger by involving team members in setting goals and by reviewing and resetting goals on a regular basis, normally quarterly.



OKRs operate within a cycle: prioritize, focus, execute and review

The following chapters discuss the objective and key results part of OKRs individually. First, though, some ground rules are helpful.

6.1 Team setting

OKRs that are handed down from above for a team to deliver run against the self-organizing ethos of agile: one cannot impose ambition on team members. The OKR-setting process is an opportunity to enroll team members in the objectives.

In an agile environment team members, including you and me, expect to have a voice in setting team OKRs. Some team members, for example a product owner or team leader, may have a privileged position in setting OKRs, but they do not have a free hand.

Yet involving all team members as near-equals may create another problem, one that economists call *satisficing*. This occurs when people aim to 'play it safe' and avoid risk –

team members agree to set goals that they feel can be achieved comfortably.

Before engaging in OKR-setting the team should clarify where they stand on the aspiration spectrum: *are you setting utility OKRs or aspirational OKRs?*

Also be clear what the organization and related teams expect of you. While organizations may expect aspirational OKRs, teams may shy away from ambition and *play it safe*.

To complicate matters, other teams may not agree. While one team may embrace ambition and aim high (while knowing they may fall short), other teams may value predictability and set goals they are confident can be achieved without stretching. This causes tension, especially when such teams need to work together.

I remember being in one meeting where I was asked by another team what my team was planning for the next quarter. I shared the OKRs and was then asked "How many of them do you expect to achieve?". I said I didn't know for sure, but 70% would be reasonable. The other team reacted with horror. Their team valued predictability: 70% was 30% too short.

Mark aspirations

Some teams mark objectives or key results that they feel to be aspirational. For example, an asterisk is placed on those that they feel are more stretching.

When delivery of particular key results or even whole objectives are required by particular dates, teams could adopt a similar strategy. These could be marked with, say, a dollar symbol '$' to indicate value in predictable delivery. This could even be followed by a particularly important date, for example 'Android port working $July15'.

However, the more aspirational outcomes a team accepts, the less likely it is that any will be achieved. Similarly, the more defined deadline items there are, the more likely it becomes that aspirations will be missed.

The odd aspirational or fixed deadline item in a team's OKRs probably isn't an issue. Having a lot, though, is probably a sign that the team is losing its autonomy and external stakeholders are trying to control it.

6.2 Limited number

The aim of the OKRs is to create focus, so it is self-defeating to set too many OKRs. If you have 20 OKRs, which do you focus on? Of course if the OKRs are set at the team level, and

there are 20 people on the team, each could have their own OKR. But is that really focus? Is it even agile?

Teams exist to share work – multi-skilled and cross functional – towards a common purpose, especially in an agile environment. OKRs serve to provide that purpose, so it makes sense to have OKRs that allow the team to focus collectively.

So how many OKRs? That depends on how tightly you want to focus and how many things are being demanded of the team. OKRs serve to help say "No" – or at least "Not now."

Personally I've come to the conclusion that three is the maximum. While I'd like it to be fewer, two or even one, I more often find myself going in the other direction and accepting four. For the teams I've worked with four seems to work. Or rather, I try and hold the line at three, but accept four. Possibly if I tried to draw the line at four I'd end up having to accept five.

So if you want a hard answer to the question "How many OKRs should a team set?", the answer is 'Three… OK, maybe four.'

This rule of thumb serves both for the number of objectives and the number of key results for each objective, so a maximum of three objectives, or four if you must, each of which has a maximum of three (four if you really need them) key results. That is now nine (3x3) and 16 (4x4) results to aim for.

In my book 16 is a lot: even nine risks losing focus. A quarterly OKR cycle is 13 weeks, so this equates with slightly more than one week per key result to slightly less than one key result a week. Either way that is not a lot of time; those key results can't be too ambitious. If you want more ambition you probably need fewer key results on which to focus. But before you say "But doesn't it depend on team size?", let me say: no.

Certainly a larger team can do more stuff, but that dilutes focus. The aim of OKRs is to achieve collective focus and goals: the more things you try and focus on, the less sharp the focus will be. Too many results ends up looking like a shopping list of things to do.

Remember, in limiting the number you are not saying "All these other things are worthless and we won't do them." What you are saying is "All these things are worth doing, but if we try to do them all we don't get very far with any of them. So we will accept a few, we will do our damnedest to get them done, and then we will look again." In other words, right now, of all the things you could do, you need to just select a few to focus on.

6.3 Priority

So you set three (or four) OKRs.

That does not mean all OKRs are equal: some might be higher priority than others. Since when written down the OKRs will form a sequence – and may even be numbered – it is natural to see the one at the top as the highest priority, the one to do first.

In the spirit of 'do the simplest thing that could possibly work', it makes eminent sense to order the OKRs in priority order. The one at the top of the list is highest priority, and the one at the bottom lowest.

There are those who might insist that all OKRs are equal. This runs counter to the philosophy of 'prioritize, focus and execute'. Prioritization may be a hard decision, but it has a large payoff because it promotes focus, and focus promotes execution.

Before you accept that two OKRs are genuinely both priority one, ask "Is it better to achieve one completely and progress the other, or is it better to advance both but complete neither?". If the latter is true then maybe the OKRs – or at least the key results – need to be broken down a little.

While it complicates things, it is possible to prioritize key results independently of objectives by interleaving them. For example: key result #1 of objective #1, key result #1 of objective #2, result #2 and #3 of objective #1, remaining objective #2 key results… But I'd rather you didn't do it like this, simply because it complicates matters – although it is possible.

6.4 Effort

In an ideal world a team would have but one objective and could systematically work through key results one at a time. More often teams find that each objective could absorb all the time available, but the team needs to make progress against multiple objectives.

In such cases it makes sense to allocate effort against OKRs. For example, suppose you have a team of five people and you are going to be working on delivering three OKRs for the next 12 weeks (six sprints). You therefore have 60 days. You might want to allocate effort as follows:

- OKR 1: 10 days
- OKR 2: 40 days
- OKR 3: 10 days

Or if you are running two-week sprints:

- OKR 1: one sprint

- OKR 2: four sprints
- OKR 3: one sprint

Notice here that priority does not correspond to the capacity allocated. It is entirely possible to say "OKR 1 is our highest priority, we really need to make progress here, but it does not need to absorb lots of time." It is important to recognize that priority and capacity allocation are different things: just because something is important does not mean it should take up a lot of time.

This is a rudimentary form of *capacity planning*. There are three keys to making this work:

1. Teams have to stop at the end of the time: they cannot say "We used all ten days, but haven't finished and need more time." They need to have something deliverable at the end of the time-box.
2. The team will not produce the perfect solution, or even a complete solution: the team will aim to improve the current position – that is, the outcome will be better than the status quo.
3. The team has the authority (and skills) to decide what to do: it is given an objective and trusted to move toward the objective.

For example, suppose you've been asked to tender for some work for a new client. It might be really important to spend some time writing the proposal, but that does not mean it should absorb lots of time.

Importantly: capacity allocations are not estimates.

To produce estimates requires some pre-work. At the very least it requires someone to sit down and think about the work and think of some number, an 'estimate'. That in turn means that someone, the same person or someone else, needs to specify what the work is – what used to be called 'requirements'. That may lead to a discussion of 'designing' the thing. Suddenly there is a lot of pre-work to do and a lot of assumptions are being made.

While well-intentioned, pre-work creates problems.

Working backwards

If you follow my advice you will have three or four objectives each with three or four key results. That is between nine and 16 key results in total – and 16 sounds too many – so say about 12 on average.

Assume OKRs are set quarterly (that is, 13 weeks) – less one week for reviewing and setting

> OKRs. So each key result has one week of total team time. That's not a lot.
>
> While you probably won't execute each key result in sequence, this simple calculation gives you some idea of how much work should be involved in a single key result and how fast you should be ticking them off as the weeks go by.

6.5 Avoid planning by OKR

Teams can be tempted to use OKRs to explain their plan of action. To use an example taken from Itamar Gilad[1]:

> 'Objective: become a leader in the enterprise
>
> Key result: launch v2.2 of the mobile app
>
> Key result: integrate with sales force
>
> Key result: switch to new onboarding flow
>
> Key result: run ten paid campaigns'

As Gilad says:

> 'Here's why this is wrong. Objectives and key results are designed to convey goals — what we're trying to achieve, by when and how we'll measure success. Building, launching and promoting features and products are not the goals. The goals are the benefits we expect to gain from these actions.'

This demonstrates another problem with plans as OKRs: the dependency problem. If key result 1 is missed then the following key results will also be missed. While it is sometimes impossible to avoid one key result depending on an earlier one, it is obviously better if they are independent. Such dependencies create fragility and hinder agility. (Later chapters return to this problem.)

[1] *5 Ways Your Company May Be Misusing OKRs*, Itamar Gilad, https://app.getpocket.com/read/2841886122, access July 2020

6.6 The trouble with pre-work

Plans codified as OKRs suggests that some pre-work has been done to create a plan to start with. While pre-work itself is not inherently bad, it does imply that someone is spending time undertaking the pre-work. Such pre-work is by definition not part of the current quarter's OKRs: focus, time and effort are being diverted from the current OKRs. Thus pre-work makes hitting the current objectives harder.

Every day an architect spends thinking about work that they expect to happen next quarter is a day not spent doing the work of this quarter. Of course pre-work may be wasted if the objective changes or gets pushed back.

However there is a more insidious problem here. While performing pre-work like this may appear rational and conscientious, it can be self-limiting. Looking at the work in advance may discourage people from taking on ambitious work or deliberately reducing the goal.

Then there is the problem with effort estimates, which are notorious for being wrong. What if your analysis and estimates indicate that the work will take more than the quarter? Should you reduce the target? To do so would be to reduce your ambition.

What if your analysis suggests that the OKR will fit into this quarter, but only just. Does that make it too risky to take on? Should you allow contingency? What if the contingency takes the OKR beyond the quarter? Does that mean that another objective is squeezed out of this quarter?

As well as reducing your capacity in this quarter, pre-work may lead you to be less aspirational.

6.7 When to set OKRs

OKRs should be set collectively by the team, in a timely manner and with thought. Setting OKRs too early is problematic, because things change and the OKR-setting process is a distraction from current work. Setting OKRs too late is also problematic, because they don't get the consideration they deserve.

There is no specific time to set OKRs. When they are set will depend on the corporate calendar (when do quarters start and end?), whether anyone is doing longer-term planning (see later chapters) and simply: when the team has time.

However OKRs should not be set weeks and weeks in advance of the quarter for which they will apply. Nor should they be set at the last minute: teams need time to discuss the

objectives.

A few weeks in advance, say two or three, should be fine. Let them marinate for a few days and then review them. I've come to believe the last week of the quarter should be used to close current OKRs and set new ones. That might make for a packed week, but that is what I will aim for next time.

6.8 Not money

> *Visionary companies pursue a cluster of objectives, of which making money is only one – and not necessarily the primary one. Yes, they seek profits, but they're equally guided by a core ideology – core values and sense of purpose beyond just making money. Yet paradoxically, the visionary companies make more money than the purely profit-driven companies.* Jim Collins and Jerry Porras, *Built to Last*, 1994

For a commercial enterprise all goals might ultimately be reduced to 'Make more money'. Don't do this.

Those who read business books such as *Built to Last* will notice a common theme: *the power of purpose*. Management guru after management guru advocate for businesses to have a purpose above and beyond making money. Earning money, making profits, is simply a side effect of fulfilling a company's purpose.

Conversely, few business books or management gurus actually argue for the pursuit of revenue and profit as an end in itself. There is a reason for this – money doesn't motivate people. Few people, and even fewer software engineers, find making money motivating. Some are, certainly – some people want to make lots of money, and that's fine. But few people get out of bed in the morning thinking "Yippee, today I'm going to make more money for the company."

OKRs need to represent value, and value might mean money, but there needs to be more meaning to the objective than simply bringing in more money. Profit is a side effect of delivering on that purpose and creating value.

> *The profit seeking paradox… the most profitable companies are not the most profit-oriented.* John Kay, *Obliquity*, 2011

7. Objectives

7.1 Background analysis

7.2 Value

7.3 Obvious value

7.4 Wide objectives

7.5 Feature factories

7.6 One for the team

7.7 Testing trouble

8. Key results

8.1 Example

8.2 Test driven

8.3 Binary or analogue?

8.4 Summary

9. Measuring

9.1 Quantify

9.2 Measuring the impossible

9.3 Removing the subjectivity

9.4 Unintended consequences

9.5 Don’t boil it down

9.6 Summary

10. Key result tricks

10.1 Experiments

10.2 Hypothesis driven development

10.3 Time-boxed

10.4 Survey

10.5 Knowing when to stop

10.6 Summary

11. Planning cycle

11.1 Gather the team

11.2 When to set

11.3 Start late

11.4 During the quarter

11.5 End of quarter review

11.6 Mid-quarter review

11.7 Product Owner

11.8 Summary

III Working with OKRs

It's important not to overstate the benefits of ideas. Quite frankly, I know it's kind of a romantic notion that you're just going to have this brilliant idea and then everything is going to be great. But the fact is that coming up with an idea is the least important part of creating something great. It has to be the right idea and have good taste, but the execution and delivery are what's key. Sergey Brin, co-founder of Google

12. Organizing to deliver OKRs

12.1 OKRs everywhere

12.2 Sprint planning with OKRs

12.3 Traffic lights and status

12.4 Summary

13. OKRs and The Backlog

13.1 OKRs not backlogs

13.2 Backlog First

13.3 OKRs First

13.4 Return of the Sprint Goal

13.5 Summary

14. BAU “Keeping the lights on”

14.1 Option 1: Suppress BAU

14.2 Option 2: Reduce/remove BAU

14.3 Option 3: Make BAU better

14.4 Option 4: Objective Zero, add BAU

14.5 Downside

14.6 Summary

15. Executing

15.1 Keeping focus

15.2 Prioritise

15.3 Visual display

15.4 Revisit often: Sprint planning

15.5 Time slice

15.6 Summary

16. Going off-piste

16.1 Unplanned but valuable

16.2 Prepare for the unexpected

16.3 Track distractions

16.4 Summary

17. Beyond the quarter

17.1 Three horizons

17.2 From roadmap to OKRs

17.3 Feedback

17.4 Summary

18. Integrated planning

18.1 OKR roadmap

18.2 The PO & planning

18.3 Summary

IV Leadership

It should be noted in conclusion that management has a much greater impact on both companies and projects than almost any other measured phenomenon. Capers Jones, *Applied Software Measurement*, 2008

The quality of the people on a project, and their organization and management, are much more important factors in the success than are the tools they use or the technical approaches they take. Frederick P Brooks, *The Mythical Man Month*, Anniversary Edition, 1995

19. Strategy

> *"Would you tell me, please, which way I ought to go from here?".*
>
> *"That depends a good deal on where you want to get to", said the Cat.*
>
> *"I don't much care where…", said Alice.*
>
> *"Then it doesn't matter which way you go", said the Cat.*
>
> *"…so long as I get SOMEWHERE", Alice added in explanation.*
>
> *"Oh, you're sure to do that".*
>
> Lewis Carroll, *Alice's Adventures in Wonderland*

When it comes to agile, many are like Alice. Agile is about the here and now. It is about being fast. It is about being responsive. As long as we are fast enough, as long as we listen to what our customers are asking for now and then deliver it, and as long as we keep the system running and fix any defect the moment we see it, then we'll definitely get somewhere.

Seeing agile as a fast and responsive system is a valid point of view. For some teams and companies it is exactly the right approach, but not always. Sometimes a different approach is better. Business people have a word for this: *strategy*.

But hang on, doesn't the agile manifesto say 'Responding to change over following a plan' – and isn't strategy a plan? Surely an agile team should always be like Alice and shun plans?

Not quite.

Responsiveness – especially when rapid and driven by customers – can be a very effective strategy, but it can also be a sign of cluelessness. Running around rapidly in circles might sometimes be the right thing to do, but it doesn't always signify progress.

OKRs highlight this question. A team could set OKRs every quarter to 'react to customer requests'. Setting such an objective would be a conscious act, and sharing such a goal with stakeholders would validate this decision. It is also possible that a team might find that, while stakeholders want a reactive team, it has other priorities.

19.1 Big goals

The best agile teams certainly are reactive and should be, but that doesn't negate the advantages of having an overarching strategy. While I say 'strategy', you might substitute 'goal', 'mission', 'vision', 'BHAG' (big hairy audacious goal), 'MTP' (massively transformative purpose) or *strategic intent*. In other words, some *big goal* the team and perhaps the whole organization is aiming for.

However you formulate it, the important thing is to have an overarching idea. The idea might be a target, a goal to aim for, or you may have one or more principles that guide you in your work – a *true north*.

Strategy may be a place you aim to reach, or a way you intend to be. Feel free to choose. The important thing to realize is that a strategy is not a plan. Or rather, a strategy need not be a plan.

There are certainly plenty of companies for whom a strategy is a plan. For them strategic planning follows strategy formulation. *Strategy as a plan* is certainly one view: another sees strategy as a pattern of consistent behavior over time[1].

This pattern may be a conscious decision: 'We will seek out large corporate customers who will pay top dollar for our product'. Or it might be emergent: one day you notice that the majority of your profit is coming from a few big customers who are paying top dollar.

Consequently strategy can be forward-looking or backward-looking. Strategy can help explain what has happened in the past. That might not sound immediately useful, but it is: recognizing (and naming) past behavior allows one to either promote it in future or deliberately deviate from it.

Once you start to think of strategy like this, it becomes clear that a team that lives in reactive mode – "We don't need no friggin' strategy – we listen to customers and do what they want" – is in fact pursuing a strategy: a strategy of prioritizing customer feedback and responding rapidly.

This is a perfectly legitimate strategy, and may be the right one for many teams. But that does not mean that it is the only strategy, or that it is the right one for your team right now. By all means pursue a reactive strategy, but please make sure that following it is a conscious decision and not one you wander into.

[1] *The Rise and Fall of Strategic Planning* (1994) by Professor Henry Mintzberg is a tour de force that demolishes the idea that strategy can be determined in advance and then executed through strategic planning. The history of strategy and strategic planning has direct parallels with the agile-versus-waterfall debate.

Strategic intent

However you define strategy it has to start with a goal – even if that goal is simply to live in the moment and not make bets on long-term goals. Such a goal is free of 'how', it is not a plan, it may even lack a 'why' or a 'when'. The goal is the thing to aim for, sometimes called *strategic intent.*

Imagine you want to get fitter. That is your strategic intent – it is a goal. You might then devise a plan (for example, 'Join a gym'), set a deadline or allocate a budget. Or it might just seed a thinking process that informs future actions: eat more healthily, drive less, walk more.

Whenever you need to make a journey, the best option might be to drive. It is more convenient than getting a bus, faster than walking or cycling and cheaper than a taxi. The benefits of walking take time to show and only materialize when repeated regularly.

On a visit to Starbucks a muffin can look very attractive. The value of one muffin is always more than the value of not eating one. No one muffin will add noticeable weight, neither will one stop you from losing weight and becoming fitter, but cumulatively it's a different story.

Knowing the goal, the *strategic intent*, informs such decisions even without a plan.

19.2 Agile makes strategy more important

Having a strategy is actually more important for an agile team than it was in pre-agile working. Unfortunately, the ethos of agile too often means that teams pursue a reactive strategy by default. Teams, and in particular Product Owners, either don't consider strategy, or simply think 'agile' means doing what customers ask for as soon as possible.

OKRs form a link between the big and possibly nebulous strategy and the specific code-face work of agile teams. OKRs derive from strategy goals and feed into sprints. Think of them as a decomposition step if you like.

Agile gives teams the tools to be very reactive, but that very capability means that teams need to decide consciously how to use it. Being reactive is satisfying: acting on any given request makes you feel good. It provides feedback: you solved a problem and added value. But that doesn't mean it's always the right thing to do.

Think of it like a knife. When your knife is blunt, the effort required to cut something means you need to choose very deliberately which cuts to make. Since it takes time to make the cut, you have a little extra time to change your mind before the damage is irreparable.

Agile gives you a very sharp knife: you can cut anything with it, but that doesn't mean you should cut everything. You now need to think more carefully about what are the right cuts to make. The danger is that one gets carried away with making cuts and receiving positive feedback without realizing that the same time, energy and tools can generate even more benefit.

Strategy elements

It is common to talk about strategy as some god-like entity, all encompassing and indivisible. Perhaps the greatest strategies are like this, each individual piece forming a vital cog so that the whole is greater than sum of its parts.

That is not always the case. Often strategy is divisible and contains different elements – *strategy elements*. Some pieces are closer to the core than others, some can change without affecting the whole.

Surrounding the core strategy are multiple elements which, while they contribute to the whole, may vary. Things like financing models, the degree of outsourcing and technology platforms might be absolutely core to the strategy, or may be variable elements. Sometimes it is only in retrospect that one might see which was truly strategic and which was merely tactical.

19.3 Opportunity cost

Suppose you spend a day satisfying the requests of a single customer. You might feel great and you might deliver a lot of business value. But how else might that day have been spent?

Quite possibly doing something else with the time would result in even more business value. In doing X you do not do Y: the lost benefit of doing something else is what economists call *opportunity cost*.

Now this could – and in the past often has – become its own time sink. Fear of not doing the most valuable things can be debilitating. Faced with a dozen possible things to do, one spends time anguishing over which one is the most valuable to do now.

Unfortunately the clock is ticking: more time spent deciding on the best thing to do means less time to actually do anything. In the extreme more time gets spent analyzing and agonizing about the best thing to do than is spent actually doing the thing[2].

Again this is where strategy comes in. Rather than turning each and every 'what to do' decision into a long-drawn-out analysis, a strategy provides a filter. Instead of agonizing about 12 things, the list of things to do drops to four or five.

19.4 What not to do

A strategy doesn't just tell you what to do – more importantly, a strategy tells you what *not* to do.

The reactive strategy tells you not to make big plans, not to promise features to customers, not to make changes that impede future options and never to say 'No' to a customer request.

Conversely, a strategy that commits the team to targeting a few high-paying customers implies not responding to every customer request, not developing parts of the system used by low-end customers, limiting support and accepting that mass-market customers might criticize the product in public forums.

Suppose your strategy is to pursue growth in the US market. You may have many customers in Europe who all deserve to be listened to and helped. A reactive strategy would treat all customers similarly. But when pursuing a US-market strategy, European customers drain your resources. It may make sense to help those customer transition away from your products.

A stated strategy provides a guide for decisions and cohesion for the team and product.

19.5 The backlog

Backlogs everywhere are full of more work than a team can do this millennium. Teams almost never burn down the backlog. Strategy allows a Product Owner to choose what to do or not to do.

What gets delivered is the result of a thousand small decisions. When those decisions lack cohesion the final product – and code base – lacks cohesion. 'Biggest bang for the buck' works a few times but is short-sighted. Applied repeatedly, you may end up with a 'Homer car' – a bunch of cool features that collectively satisfy nobody but Homer Simpson.

[2] I keep a dice by my desk; when I spot myself falling into this trap I number the options and roll.

Nor is it just backlog management that can benefit from a strategy. The team itself can benefit. The strategy is a form of goal – or maybe a goal is a form of strategy: you decide. Having a shared goal means that individuals can align their thoughts and efforts, increase their focus and share work more easily – and can celebrate together!

The software benefits too: design and refactoring decisions are informed about the aims of the product.

19.6 Don't forget the technology

Specifically, don't forget technical liabilities – what most people call *technical debt*. Having a strategy in place allows team members to reason about the technology solution and how its weaknesses will affect work. Having a strategy gives context to a conversation about how liabilities will be addressed, as well as the results if they are not addressed.

Technical excellence and consistently high quality are key strategy elements for agile teams. While some might see this as an excessively principled position, keeping technical quality high and minimizing liabilities, is actually a pragmatic position that will lead to more efficient working and improved return on investment.

However, not everyone agrees that 'quality is free'. Each team must therefore navigate this issue for itself. Having an explicit shared strategy for the team, product and business must be a precursor for such navigation. Without a strategy to reference there is no common position to start from.

Technical liabilities

'Technical debt' is a misunderstood and overused metaphor. Debt has a good side: mortgages allow families to buy houses, credit cards make Christmas affordable, debt allows businesses to grow and governments to respond to pandemics.

Debt is often the preferred form of business financing, so to a business person technical debt may well sound like a good thing. Engineers rarely perceive this interpretation, however.

Liabilities is a less ambiguous metaphor for everyone.

19.7 Shared mental model

Used like this, strategy becomes a heuristic that accelerates decision-making, particularly when agreeing OKRs. Stating such a heuristic allows sharing. No longer is the logic of decision-making locked inside one person's head: the whole team can share the same thinking.

While it is big decisions – strategy, OKRs, architecture, planning – that get the attention, every team member is constantly making many tiny decisions. What to call a function? When to make a function private or public? Is this a bug or a feature? Strategy and OKRs provide a guide when making decisions.

When people work together as a team decisions and actions need to be congruent. If the team is not making decisions that are in agreement, or members not acting in harmony with each other, performance suffers. At worst, in time the team may rip itself apart.

Strategic intent and strategy are foundations for high-performing teams. Teams that share an objective are more cohesive, work better together and have a better chance of achieving goals.

In an agile environment teams are more important than ever, so it is more important for them to share a common goal, common approach and common understanding. Quite possibly the benefit to the team from having a shared purpose is more significant than any of the other factors outlined here.

19.8 Summary

- The default agile strategy is for teams to be listening to customers and constantly reacting. This in itself is a strategy, but perhaps not the best one.
- Strategic intent, mission, vision, BHAG, MTP or just plain goal: it helps to have a common target.
- Strategy represents overarching principles and goals, makes a conscious decision as to how reactive to be, and acts as a filter to assist and accelerate decision-making.
- OKRs connect the overarching strategy with regular sprints. Rewriting them revisits the strategic intent and strategy.
- OKRs help to codify and communicate strategy and allow stakeholders to question that strategy.
- Strategy makes it easier to decide what not to do.

20. Leaders

20.1 Culture, goals and strategy elements

20.2 Day-to-day

20.3 Leaders and culture

20.4 Summary

21. Culture

21.1 Delivery culture

21.2 Customers

21.3 Openness and feedback

21.4 Psychological safety

21.5 Ambition

21.6 Summary

22. Leaders and planning

22.1 Broad-narrow

22.2 Forward planning

22.3 Cascade up, not down

22.4 Summary

V Forewarnings

It is impossible to live without failing at something, unless you live so cautiously that you might as well not have lived at all – in which case, you fail by default. J K Rowling, author

23. Aspirations

23.1 Utility mode

23.2 Creating aspirations?

23.3 Leaders and culture

23.4 An OKRs adoption route

23.5 Exercise: Where are you?

23.6 Summary

24. Every day pitfalls

24.1 OKR buffet

24.2 Late arriving OKRs

24.3 Adding to the story hierarchy

24.4 Counting problems

24.5 Respect for specialists

24.6 Respect for managers

24.7 Summary

25. Trouble with targets

25.1 Targeting the measurable

25.2 Questions measurement can’t answer

25.3 Goodhart’s Law

25.4 Goal displacement

25.5 Overcoming tunnel vision

25.6 Final warning: targets

25.7 Summary

26. Individuals and performance reviews

26.1 Integrating employee reviews with OKRs

26.2 OKRs for individuals

26.3 Summary

Closing words

> *What I'm proposing, to myself and other people, is what I often call the tourist attitude – that you act as though you've never been there before. So that you're not supposed to know anything about it. If you really get down to brass tacks, we have never been anywhere before.* John Cage, composer, 1912–1992

This book attempts to share what I learned during a year working with agile teams and OKRs. Maybe if I had waited until I had two years' experience I would have more and better advice to give, but I wanted to write it now while all these learnings are fresh in my head and before I lose that all-important *tourist mentality*. This is the book I wish I had had when I began the OKR journey.

If you had told me a few years ago that I would write a book about OKRs I would not have believed you. I was skeptical about OKRs; they sounded like a reinvention of MBOs – management by objective – with a similar set of associated problems plus a quantification fetish. Catch me in a pompous mood and I will readily claim credit for introducing the software industry to *Goodhart's Law*.

So when I learned the organization I was helping to become agile was also introducing OKRs I was armed with plenty of arguments – but I bit my tongue. Sometimes one has to pick one's battles – or at least pick the right time to fight.

After a little consideration I decided to see the introduction of OKRs not as a problem, not as something to fight, but as an opportunity. Working with OKRs could be a great experiment – do they work? Are my fears well-founded? If nothing else, it helps to *know thy enemy*.

Over the months of working with OKRs, helping two teams set and pursue them directly, plus being a member of a third team writing and pursuing OKRs, I had the opportunity to discuss OKRs with my fellow agile coaches, and my opinion changed.

As Nietzsche wrote, *'What does not kill me makes me stronger'*. After worked with OKRs intensively for a year I still have my doubts, but I can see how they work, and work well. Indeed, more than that, I see great promise in OKRs. I need to run some more experiments.

I wanted to capture this learning for myself, but, as every author knows, the person who learns most from a book is the person who writes the book. Writing a book forces one to distill

one's thinking and reconcile one's logic. Hopefully readers will learn too, but the process of capturing, structuring and communicating one's thoughts leads to more and deeper insights.

More than this, though, writing a book forces you to explore where your thinking goes. For example, in writing the strategy and planning chapters here, I had to do more than draw on direct experience: I had to extend my thinking to work out how the different moving parts of agile, OKRs, strategy and planning all interlocked.

So thank you, dear reader – thank you for helping me to learn. I hope I can help you too.

Would I recommend OKRs to a friend? Yes.

Would I introduce OKRs to a team and an organization? Yes.

Do I continue to harbor reservations? Yes again.

Like so many other tools, OKRs can be used for good or for bad. They can be used in better ways and worse ways. They are far from foolproof, but I believe they have a place.

Get out of jail free

> *The only thing you can do wrong in agile is doing things the same way as you did three months ago. Always be learning, always be experimenting and changing.*
> Allan Kelly

Some of the suggestions made in this book might not be acceptable to people in your organization. As with agile, you need to find your own way to OKRs. You can listen to sage advice, read esteemed books and copy best practice, but ultimately you have to find what works in your culture.

Be prepared to experiment. If it helps, consider every word in this book as a thesis to be tested through your own experimentation.

The truth is that while there may be hard and fast rules about OKRs at the likes of Google and Intel, most organizations are a long way from such rules. Anyone who claims to know about OKRs – including me – is retelling their experience gained in a particular context. Your context is almost certainly different.

When OKRs are combined with agile the people doing the work – like you – also have a say in how things work. Not only does agile push authority down to people, but agile allows – even mandates – that those doing the work have a voice in how the work is done. Agile allows itself to be modified. When OKRs are introduced to an agile environment one should expect their usage to change.

Therefore experiment with how you draft you OKRs, how you state measurements, how you document them, how you share and just about everything else. If people don't like it they will tell you and you won't do it again.

In the unlikely event that your company has chosen to make this book company lore, please use this section as your 'get out of jail free' card to break any rule.

Finally

This book is written, produced and published by myself, Allan Kelly through my company Software Strategy - also known as *Allan Kelly Associates*. That means I am responsible for all the "mistakes".

I like to think I'm good at expressing myself in my native language but frankly, spelling, punctuation, grammar and such is not my strong point. Despite a professional copy edit some mistakes will slip through.

If you have any comments or observations about this book, or have OKR stories to share please contact me, I am allan@allankelly.net.

ISBNs

Succeeding with Agile & OKRs, 2021

978-1-912832-06-4 Print version (Amazon)

978-1-912832-07-1 E-Book Mobi format

978-1-912832-08-8 E-Book ePub format

978-1-912832-09-5 E-Book PDF format

Further reading

Acknowledgements

Coming soon: OKRs extra

Less is more, so this book has tried to stay small.

But there is more to say about working with OKRs and agile. Some of that gets into thorny issues of managers, management, teams, value (just what is it?) and more. Those chapters exists, they're just not here.

To go deeper into OKRs, continue the journey with Succeeding with OKRs in Agile Extra[1] – coming soon on LeanPub. Register your interest today and be the first to know.

[1] https://leanpub.com/agileokrsextra

Also by Allan Kelly

The Art of Agile Product Ownership, Apress, 2019

Continuous Digital, Software Strategy/LeanPub, 2018

Project Myopia, Software Strategy/LeanPub, 2018

Little Book of Requirements and User Stories, Software Strategy/LeanPub, 2017

Xanpan: Team Centric Agile Software Development, Software Strategy/LeanPub, 2015

Business Patterns for Software Developers, Wiley, 2012

EuroPLoP 2009 Proceedings of 14th European Conference on Pattern Languages of Programming, Irese Germany, July 2009

Changing Software Development: Learning to be Agile, Wiley, 2008