Allan Kelly

SECOND EDITION

Prefazione di: Mike Burrows

Come avere successo con gli OKR

una guida pratica per iniziare da subito

Traduzione a cura di:
F.Racanati, F.Fullone e D.Favre

Come avere successo con gli OKR (2nd edition, Italian)

Una guida pratica per iniziare da subito

Allan Kelly, Francesco Racanati, Dimitri Favre e Francesco Fullone

Questo libro è disponibile su https://leanpub.com/agileokrs2-it

Questa versione è stata pubblicata il 2025-04-07

Indice

Praise for the first edition

"Allan's writing is perfect for busy managers who need to set objectives and form initiatives that satisfy diverse stakeholders. Plus, it doesn't sugar-coat OKRs – they are part of a management system – not a medicine. Overall, a wonderfully concise and easy to read guide to using OKRs. Highly recommended." Russ Lewis, CxO coach

"Having read other books that argue strongly for OKRs as a panacea to achieve a high-performing organization, the perspectives here from Allan are balanced, informed by lived experience and provide patterns and anti-patterns to watch for. Insightful and powerful – thank you Allan for sharing such well-considered feedback." Richard James, Accenture

"This book is full of great insights and guidance around OKRs backed by multiple real examples that will be helpful in a number of domains… With OKRs increasing in popularity and numerous 'experts' emerging, save yourself the time and effort reading the wrong material and grab a copy of this book to learn more about the right ways to approach OKRs." Nicolas Brown

"This is absolutely brilliant book. If you really want to learn more regarding OKR in Agile and other important stuff related to it you must read this book. It's truly a master piece." N Mehta

"Super easy to read, with clear chapter summaries. Lots of solid content that is not hidden amongst fluff. Straight to the point and a must have for any agile practitioner curious about OKRs." Amazon buyer

"Allan has written plenty of good books, but this might be the best. It's certainly immediately useful to any team adopting OKRs. Whether you're just starting out or have been using OKRs for months, there's a wealth of hints, tips, heuristics and advice in this book." Seb Rose, Agile Coach

"This book, having established in its opening section the value of an OKR-driven approach, devotes the bulk of its focus to practical guidance which cuts through

the all too common confusion most of us encounter when actually writing and working with OKRs. The forewarnings section in particular provides refreshing advice for making OKRs work for you." Ewan Milne, Agile Coach

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 e-book – currently *Continuous Digital*[2] – when subscribing[3].

Continuous Digital

[1] http://allankelly.net/newsletter
[2] https://amzn.to/47HExTu
[3] 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. 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[4]

Author, Agendashift: Outcome-oriented change and continuous transformation[5] (2nd edition March 2021)

Author, Right to Left: The digital leader's guide to Lean and Agile[6] (2019, audiobook 2020)

Author, Kanban from the Inside[7] (2014)

[4] https://www.agendashift.com/
[5] https://www.agendashift.com/books/agendashift-2nd-edition
[6] https://amzn.to/2NRZSoP
[7] 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 that 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 see 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[8]. That book sets out an 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

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

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.

Preface to the second edition

Rereading the first edition I surprise myself. I realise many ideas I had for changes were there already, albeit sometimes in prototype. So, *why a second edition?*

Well, books can always be better, the writing clearer, the ideas crisper. But one has to draw the line somewhere and *ship it*.

Books can always be longer and cover more material. I originally wrote several more chapters for *Succeeding with OKRs in Agile* that never made it into the final version. Adding chapters does not necessarily make a book better. As I often say, "The longer a document is, the less likely it is to be read", and "The longer a document is, then – if it is read – the less it is remembered." *Less is more.*

So additions are deliberately limited. The curious may like Succeeding with OKRs in Agile Extra[9], which contains these chapters, additional essays and posts. One day *Extra* might be finished in its own right; until then the draft is available on LeanPub and in a state of flux.

So why a second edition – what is different?

After I finished the first edition I continued learning, talking about the book, talking to teams, observing and listening to others. Now I want to share that learning and understanding.

The majority of changes concern planning and key results. The discussion of both has lead to new chapters and some refactoring.

The key driver for changing both was seeing *key results* more clearly as *acceptance criteria*. Key results are – well – *key*: important, crucial, fundamental, and *results* – outcomes, consequences, a description of the world. While the first edition advocated this view with experience I wanted it to be clearer.

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

The more I worked with OKRs, the more I have heard from others about OKRs, the clearer it has been to me that *key results are the difficult bit*. People struggle with key results more than anything else.

To be fair, one might say that opinion is divided. Authors differ in their advice on OKRs. Even some of the most authoritative texts on OKRs give examples I would not recommend now. I am also sure that in the past I have written OKRs with key results that I would disapprove of today.

To square this circle I've outlined four different approaches to key results. I recommend that you avoid seeing key results as small pieces of the objective that are joined to make the complete thing. Rather key results are attributes of the target outcome that describe it – they describe the parameters the completed objective will meet. They might be constraints on the outcome ('Weighs less than 10kg'), sometimes they describe constraints on the input ('Costs less than $100 to manufacture'). Key results are, in agile terms, *acceptance criteria*. Which nicely furthers the idea that OKRs are *test-first management*.

Put it another way: objectives are not epics that are broken down into small pieces called 'key results'. Rather, objectives are akin to big stories – *business beneficial outcomes* – and key results are the acceptance criteria those stories should meet.

Reducing objectives and key results to an exercise in mereology means that key result writing becomes an exercise in work breakdown. That in turn means the OKR-setting becomes work planning. Consequently discussion of the production process displaces a focus on outcomes and customers.

Much else becomes easier when key results are interpreted as outcomes and acceptance criteria. OKRs cease to be glorified to-do lists and cascading OKRs become nonsensical. Key results as acceptance criteria might be harder to write, but doing so increases team freedom and options.

One example is in this book itself. Having a clearer perspective of key results allowed me to revisit the discussion of OKR-setting and planning. Under this view it becomes clear that OKR-setting is not work planning. Rather there is a two-step process: set OKRs, then plan delivery. Calling this out makes the whole setting/planning discussion clearer.

Revisiting these chapters also allows me to revisit the question of OKR cycle length. Notice I say 'OKR cycle', not 'quarter'. While there is the convention that OKRs operate on a quarterly cycle – three months, 13 weeks – this is not set in stone. I have come to see merit in both longer cycles, 16 weeks say, and shorter cycles, say ten weeks.

Today I see OKRs creating the opportunity for a new style of agile: *Objective Driven Agile*. Teams coordinate through interfaces called OKRs, backlogs only exist to meet an objective and are then discarded, planning is driven by those objectives and "How long will it take?" gives way to "How close can we get?".

But rather than lengthen this book further, *Objective Driven Agile* must wait for another day.

Allan Kelly, London, August 2023.

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

Objectives are outcomes you wish to bring about.

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

Don't change or abandon OKRs during a cycle 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 short 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 a few 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 that 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.

3 Questions

Whether you are about to begin your OKR journey or already in flight there are three questions, and answers, you should agree with you team and stakeholders. If you are working agile these questions are:

- **Which comes first: Backlog or OKRs?**

  Does the backlog drive the OKRs, or the OKRs drive the backlog?

- **Where does *business as usual* fit in?**

  When asked to do something important that does not directly relate to an OKR do you reply: "I'm sorry, OKRs take priority", or "OK, I'll do the OKR later" ? This is particularly important for teams covering *DevOps*.

- **Which is more important: predictability or ambition?**

  Do your stakeholders place higher value on predictability or on ambition, even if the team falls short?

Even if you are working in a less agile, more traditional style these questions are still relevant. The first question generalizes to "Which comes first: the work manifest/requirements or OKRs?"

Ultimately all three questions pertain to the priorities applied to your workflow. This book aims to help you answer these questions, and suggests my own answers.

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, while key results are attributes of that goal. Sometimes they might be interim goals on the path to big goals, but more often they are descriptive of the objective. The question is, *how ambitious do you want to be?*

Make your objectives too big and ambitious and you might miss. Make your objectives 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]

[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

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 tradeoffs. There may not be an obvious answer, but it is critical that everyone shares the same abstraction.

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.

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 endeavour to help a client. Whatever it is, today's objective requires some significant work.

Key results are the important things that make that objective meaningful. Some like to see them as milestones, but I like to think they are more descriptive of the whole objective. Rather than describe some discrete part of the objective, I have come to see key result an attribute of the whole.

For example, an objective might be to build an electric car. 'Car can travel over 400 miles on single charge', or 'Car can be fully charged in two hours', are key results that relate the whole. It might be easier to write key results

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

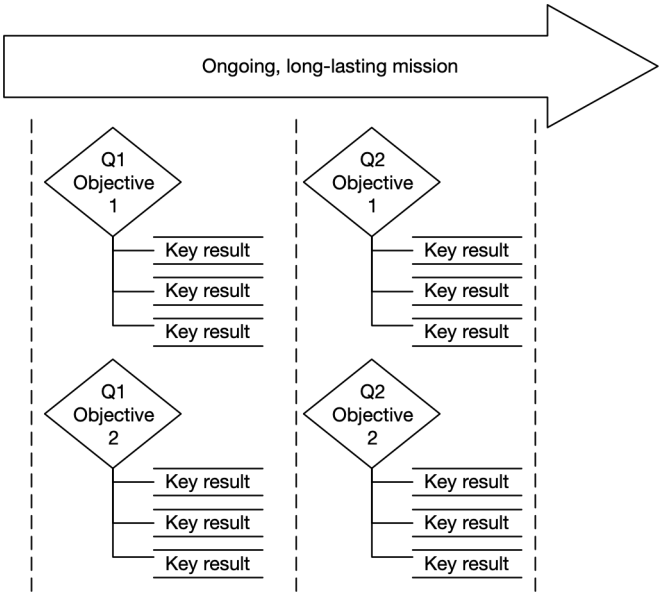
for components – 'Car has an electric engine' and 'Battery capacity exceeds 70kWh' – and the OKR is easier to achieve because those components might be individually achievable – but the ultimate objective is the whole.

Good OKRs are outcome-focused. I like to say "An objective is an outcome you wish to bring about".

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. Ideally achieving a key result should represent benefit (value). An electric car may still be useful if needs charging every 300 miles, but achieving 400 miles is worth more.

An objective should have its own *wholeness* that is more than the sum of its parts. 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.



Missions are long-lasting, OKRs are reset every quarter

While missions typically last a long time, 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 may pursue several objectives over the course of a quarter, and each objective has several key results – hence objectives (plural) and key results (plural). But remember, the more objectives the team pursues, the less it can focus. For a really focused team it could be one objective (singular), albeit with several key results (plural).

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 those targets?* Then 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.

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 cycle 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.

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.

As a rule of thumb objectives are inspiring and subjective. Key results are objective and quantified with numbers.

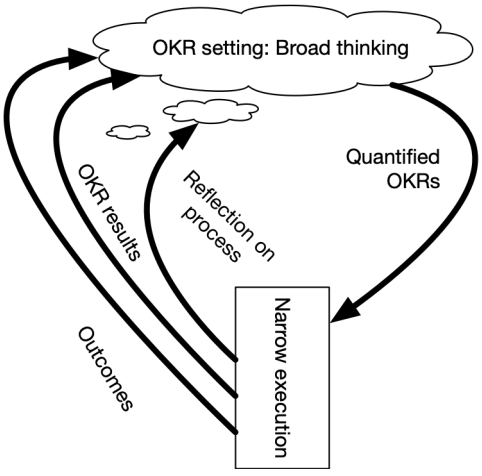
The laser-like focus of delivering OKRs needs moderating with expansive and considered thinking during OKR-setting. Teams should alternate between reflection and broad thinking during OKR review and setting and really focused actions during delivery. The former should last hours, the latter months.

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.

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. In agile environments OKRs are likely to be set by the same people who are going to deliver them. OKRs are not about top-down control, they are more 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.

2. Why use OKRs?

> *Previously, this realization would have resulted in replanning to move out the target schedule, perhaps repeatedly. Instead, given the group's commitment to the larger result, we found a much more aggressive behaviour. For example, the OpenVMS AXP group publicly committed to their target schedule and stated, "We don't know how to achieve this, but we commit to finding a way." The next day they went to a project management consultant for training on how to build an aggressive, attainable schedule.*
>
> Peter F Conklin, director of Alpha AXP Systems Development[1]

Objectives and key results – OKRs from here on – did not start life as part of the agile toolkit. Indeed they predate 'agile' by 20 or 30 years. Yet in recent years they have received more and more attention in agile circles. More companies and teams are experimenting with them, and it's no longer uncommon to find them used by agile teams. They may not quite be a standard item in the agile toolkit, but I wouldn't be surprised if they become one.

Why? Or perhaps, what makes OKRs a good fit with agile working?

To my mind there are three reasons why OKRs work well with an agile approach: they fill a need at the mid-term planning level, OKRs are essentially a test-first approach, and OKRs enhance communication.

If OKRs sound too good to be true – and reading some authors, they *can* sound like that – then rest assured: OKRs can go wrong in many ways. The laser-like focus OKRs create needs to be tempered with countermeasures.

[1] *Enrollment Management, Managing the Alpha AXP Program, *Digital Technical Journal* Vol. 4 No. 4 Special Issue 1992

Mid-term planning

OKRs fill a hole in the agile planning processes that many teams struggle with. By planning, I mean planning in all its forms – software, UX and test design, coordination and scheduling.

Agile, and in particular Scrum, has plenty of short-term planning tools: the morning stand-up is a form of daily planning, Scrum and XP teams will have regular sprint/iteration planning meetings, and retrospectives are also a form of planning.

Teams usually have some long-term plan too – although these are usually independent of the chosen agile framework.

A team may have a long-term plan or a roadmap[2], a mission or vision statement, a business plan or statement of the market opportunity, maybe a *job to be done*, or a customer problem the team is addressing. Such plans usually start beyond the current quarter and may look years into the future.

However, in the middle ground lies a problem. There is no standard way of thinking about the mid-term, the period beyond the sprint but a little longer than the quarter.

Once upon a time teams created 'release plans' that showed what they planned to 'release' (or rather, build and release) during the next few weeks. For some teams that might mean sketching what would be in each of the next six release over the coming quarter (that is, three months, 13 weeks). Or it might simply be a list of what was to be included in a release that only happened once a quarter.

However with widespread adoption of continuous delivery, release plans make less sense. What is the point of a 13-week release plan when a team releases many times a day?

Other authors have suggested other solutions. I myself have advocated 'quarter plans' – that is, the plan for the coming quarter of the year[3][4]. I and others have tried to sketch what a process would look like around that, but since there has

[2]Roadmaps suffer from several problems themselves. All too often they are little more than a list of features with speculative dates against them.

[3]*The Art of Agile Product Ownership*, Allan Kelly, Apress, 2019

[4]*An Agile Reader*, Allan Kelly, LeanPub, 2017

not been consensus on 'what is the right thing to do', few of those solutions have become mainstream.

OKRS offer the opportunity to fill that gap and provide the glue between the short-term daily and sprint planning and long-term plans.

OKRs potentially provide a way of balancing the demands of here and now with the need to steer some kind of course. By focusing on the quarter OKRs can provide mid-term goals, consistent enough over several sprints to achieve meaningful results, but flexible enough not to mislead the team.

Quarterly, three months, 12 weeks

In just about every case I have heard of, OKRs are reviewed and updated quarterly – four times a year. I can imaging using them on a shorter cycle – every six weeks maybe – or on a longer cycle, perhaps annual – but the consensus seems to be quarterly.

Quarterly seems about right; three months is a good balance between thinking more strategically and getting on and doing it. It implies sticking with something for long enough to see whether it works, but not so long that one is flogging a dead horse long after it has stopped breathing.

However, companies already make heavy use of quarterly cycles for things like budgets, sales targets and performance reviews. There is a good case for not adding another process on the same cadence. A ten-week or four month cadence might work better for OKR cycles. This would also avoid coupling OKRs to things like performance reviews.

Test-driven OKRs

Finally, one reason why OKRs work well with agile is that OKRs are a high-level implementation of test-driven development. I sometimes think of them as 'test-driven management'. Consequently they sit well with the agile mindset.

Each objective (the 'O') has a set of key results: the 'KRs'. Each of these key results is a test: has the result been achieved?

Each key result should be measurable. One should be able to look at the key result at the end of the period and say: did we achieve it? Or better still: how much did we achieve? How close did we get?

In other words: right at the start of the period, when setting the KRs, people are thinking "How will we know that this is done" and "How will we test that this has been achieved?"

Anyone who has practiced test-driven development at the unit test level, written acceptance criteria for a user story, or sketched a BDD-style scenario before any code is written will recognize this approach. This is what agile calls *test first.*

Test-first works well for at least two reasons. First, a test-first approach creates focus – yes, focus again! By knowing the tests that the work must pass to be successful, one is able to discount some work and measure progress towards the desired result. Anyone who has written automated test cases that show as green and red result bars will know the motivational power of such a feedback loop.

> Green means success, and you want more success!
>
> Red means something failed and it's damned annoying – you want to make it work!

Either way you get a dopamine hit and are motivated to carry on – fix the red thing or add more and get more green.

The second reason test-first works is because it tells you when to stop: *stop when the tests pass.*

Given any piece of work there is always a temptation to keep doing more and more – particularly when there is a positive feedback loop in place. Yet doing more and more means you will do more than is required – the famous 'gold plating' that managers believe software engineers regularly engage in.

My friends Jon Jagger and Kevlin Henney like to ask: "Why do cars have brakes?" Most people answer the question "So you can stop!" or "So you can drive safely." Jon and Kevlin like to say: "So you can drive fast."

I vividly remember the day the brakes failed on my first car. Or rather I remember driving my car to get the broken brakes fixed. The car – the engine – worked, but I had to navigate traffic lights and a hill between my apartment and the garage. As the car had manual gears (that's a stick shift for American readers), I could slow down by changing down (although I pretty much stayed in first the whole way) and use the handbrake to stop. Needless to say, I drove very, very slowly.

When you work test-first you don't stop until the tests pass[5].

That is true at the unit test level, the story level, and – with OKRs – the business objective level.

By the way, if you want a third reason why test-first works, let me add: it forces you to think about what you want in advance. While that is true, I think the first two reasons are far more powerful.

Communication

By summarizing the work a team has done and will do into a standardized format, OKRs make it easier to communicate what a team is doing. I like to think of OKRs as creating an interface, or API, for the team. That might be for communication within the team, with other teams, or to managers higher in the hierarchy.

Similarly, the standardized format simplifies status reporting. Today, everyone knows that 'percentage of requirements done' is a meaningless metric: teams need a way of showing what they have achieved in the current period, what they are doing now and what they (hope) to work on next. OKRs offer a way of summarizing this information.

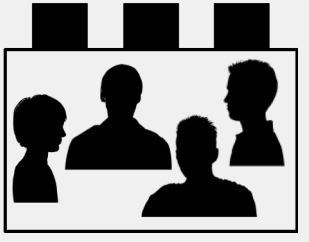
Because OKRs offer a means of communicating status and progress, they also offer a mechanism for judging success and failure. Success motivates continuation – "More of the same please!" – while failure motivates change: "Don't let that happen again!"

[5]Although when coding after the tests are passed, you probably engage in a little refactoring. Refactoring is essential, but knowing when to stop refactoring can be hard, simply because there is no test to tell you when you are done.

A team API

The team, especially the product analysts, listen to what the senior leaders say about purpose, strategy and company objectives. They also listen to what customers and the market want. And they listen to the needs of peer teams, the product itself and many other sources of work requests.

Each cycle the team responds with OKRs which form an API - *application programming interface*. This API tells others the outcomes to expect from the team in the coming cycle. The API sets out what the team will aim to achieve, what they will forego and the priorities. There may be some negotiation during the setting process but once the OKRs are set the team is trusted to do their best.



OKRs create an API for the team

The team

In addition to communicating outside a team, OKRs enhance communication inside it. OKRs build a shared understanding and goals, and thus create a vocabulary for discussing work.

This first happens in OKR-setting, where team members have a voice about what is being proposed and how the goals are formulated. In creating focus and common goals OKRs bind the team together – more about focus in the next chapter.

Of course, such benefits depend on teams playing a role in OKR-setting. Unfortunately, where companies impose OKRs from outside the team, such benefits are lost. (Chapter 25 has more on cascading OKRs.)

Warning

Few things in life come with no downside, and OKRs carry certain risks. I'll dig into some of these risks later, but right now I want to make it clear that OKRs are not risk-free. Like any powerful tool, OKRs entail certain risks: the more you understand the risks, the better you will be at avoiding them.

Sometimes the right thing to do is throw the OKR away and work on what is in front of you. If your live server goes down and customers are without services there is no point in saying "Sorry guv'nor, I'm working on my OKR." OKRs create focus, but they shouldn't create blindness.

That philosophy extends to the results too. I'm going to argue that OKRs should be measurable, but not everything that counts can be counted. Not everything that is important can be mapped out in advance.

Consider your life partner, or, if you are single, think of your parents. When choosing to spend the rest of our lives with one person, who draws up a requirements list?

Actually, Charles Darwin did. Reportedly Darwin drew up a list of pros and cons for marriage. Cons included:

- 'Being forced to visit relatives, and to bend in every trifle'
- 'Loss of freedom to go where one liked, the conversation of clever men at clubs'
- 'Terrible loss of time'

Still Darwin concluded:

> *It is intolerable to think of spending one's whole life, like a neuter bee, working, working – only picture to yourself a nice soft wife on a sofa.*

He ends his notes 'marry – marry – marry Q.E.D.' [6]

> *Darwin married Emma Wedgwood in January 1839, a little over two years after completing his journey on HMS Beagle.*

Even though I, and possibly you too, like to think of ourselves as rational people when it comes to big life decisions, rational tools are often abandoned. Life partners, whether to have children or not (and how many!), divorce (Heaven forbid) and even buying a house are more likely to come down to emotion rather than rationality.

As humans we sometimes do things not because they are rational, but because we want to. Call it intuition or motivation. If we only did things that we could justify rationally (before the event) life would be boring and in time machines could probably replace us. Sometimes the important thing is *what do we want to do?*

Summary

- OKRs create focus.
- Set and reviewed on a quarterly basis, OKRs fill a gap in agile between sprints and roadmaps.
- Being test-first in nature, OKRs fit well with the agile mindset.
- Some things are more important than OKRs, and sometimes those things can't be measured.

[6] Darwin C, *The Autobiography of Charles Darwin*, ed. N Barlow, London, Collins, 1958 and quoted by John Kay, *Obliquity*, 2011

3. Focus

OKRs create focus

Summary

4. OKR history

5. Outcomes, value and benefits

Business benefit and value

Value

Pieconomics

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 is an operational issue. It demands narrow, focused, action – prioritization. Think strategically when setting, think narrowly when delivering.

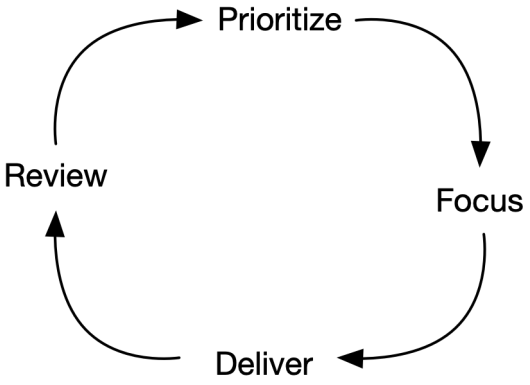
Prioritization is not just about deciding what to do, it is also about deciding what not to do. In writing OKRs for the coming quarter a team is consciously deciding what it will aim for. Perhaps more importantly, the team is also saying what it will not aim for. Everything that is not in the OKRs is, 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 it stands little if any chance of being done. Teams are expected to give OKRs the best shot possible. Doing work that does not support an OKR displaces work that does.

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, deliver and review

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

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 enrol 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?*

'Utility OKRs' serve to prioritize, organize and communicate work. They do not embody aspirations – teams may deliberately seek to stay in their comfort zone. Utility OKRs go against a lot of OKR literature that emphasizes aspirations. But organizations that value predictability or lack psychological safety such OKRs can still be useful, and may be a stepping stone towards more aspiration.

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 in which my team presented their OKRs for the quarter. An internal customer asked "How many of these OKR 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 goals are required by specific 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 $July23'.

However, the greater the aspirational outcomes a team aims for, the less likely it is that they 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, could be a sign that the team is losing its autonomy and external stakeholders are trying to control it.

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 aim for a maximum of three objectives, or four if you must, each of which has a maximum of three (or 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.

Paradoxically, bigger teams should aim for fewer OKRs. When teams are bigger it is harder to achieve focus. Bigger teams do not deliver more OKRs, bigger teams demand more focus. Rather than load a big team with six OKRs, I would prefer to split the team in two and ask each to pursue a few OKRs.

Remember, in limiting the number you are not claiming that all the 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 won't get very far with any of them. So we will accept a few, and 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.

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.

Effort

In an ideal world a team would have but one objective and could systematically work through key results. 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.

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.'

[1] *5 Ways Your Company May Be Misusing OKRs*, Itamar Gilad, https://itamargilad.com/5-ways-your-company-may-be-misusing-okrs/, access July 2020

This demonstrates another problem with OKRs as plans: the dependency problem. If key result 1 is missed, then the following key results will also be missed – a domino effect. 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.)

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

However there is a more insidious problem here. While performing pre-work 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.

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, your chosen OKR cadence, how many other teams and stakeholders you need to consult with, whether anyone is doing longer-term planning (see later chapters) and 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 busy week, but that is what I will aim for next time.

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.

As Alex Edmunds, who I discussed in Chapter 5, puts it in *Grow the Pie*: "The pie-growing mentality… aspires to grow the pie – to create value for society – because doing so benefits both investors and stakeholders alike. Profits, then, are no longer the end goal, but instead arise as a byproduct of creating value".

Conversely, few business books or management gurus actually argue for the pursuit of revenue and profit as an end in themselves. 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

Summary

- Think strategically when setting OKRs, prioritize and don't try to do everything.
- agile ethos OKRs demand that the teams who will deliver them set them.
- OKRs should be aspirational, although aspirations may need to be tempered for the environment.
- Creating focus with OKRs demands that the number of OKRs for each cycle is limited to no more than most people can count on the fingers of one hand.
- The length and events of the OKR cycle, the amount of pre-planning and estimation, differ between teams.
- OKR outcomes should deliver value, but that does not mean targeting money directly.

7. Objectives

Background analysis

Objective value

Obvious value

Wide objectives

Feature factories

One for the team

Testing trouble

Take time but not too much time

Summary

8. Key results

Test first

Testable key results

Binary or analog?

Summary - the end

9. Four types of key results

Type 1: Acceptance criteria

Type 2: Plan

Type 3: Lego bricks

Type 4: Vertical slices

Contrast

Implications for cascading

Domino effect

Summary

10. Objective worked example

The date

Minimal?

Context and constraints

Pharmacy

MVP

Full size

What is key?

Summary

11. Measuring

Quantify

Measuring the impossible

Removing the subjectivity

Unintended consequences

Don’t boil it down

Summary

12. Key result tricks

Experiments

Hypothesis-driven development

Time-boxed

Survey

Knowing when to stop

Summary

13. OKR cycle

OKR cycle

Cycle length

OKR setting is not work planning

What about work planning?

Summary

14. Planning players

Product Owner

Stakeholders

Managers are stakeholders too

Summary

15. Planning to plan

Schedule the events

When to set

Start late

During the cycle

End-of-cycle review

Mid-cycle review

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

16. Organizing to deliver OKRs

The absence of alternatives clears the mind marvellously.

Henry Kissinger

So you now have your OKRs sorted out and day one of the quarter has arrived – what do you do? How do you ensure that you give the OKRs your best shot?

Broadly speaking, there are two schools of thought on how to organize for OKRs.

The first school sees OKRs as additive: there is all this stuff that needs to be done, a backlog, business-as-usual, meetings and other daily shit. Work can come from many places: support desks and specific sales and personal objectives are common sources. OKRs are added as another factor in this mix without displacing any others.

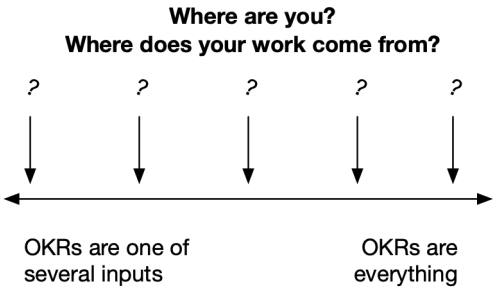
Maybe OKRs are added as an extra list of things you should be doing, or maybe they are an attempt to bring order to the other stuff. Either way, backlogs still need to be burnt down, support calls answered and so on.

In this mode OKRs add more work to what is probably an already full system. Some sort of magical thinking makes people believe that OKRs are the missing ingredient that will bring harmony to everything else.

The second school of thought says that *OKRs are everything*: every decision and action is subservient to the OKRs. Don't even get out of bed in the morning if it doesn't contribute towards an OKR. Every story, every process, every decision flows from the OKRs.

As is often the case, these two descriptions represent opposite ends of a spectrum. The first school sees OKRs as a new way of injecting work, while the second see OKRs as the origin of all work and all organization. You and your organization are free to choose where on this spectrum you position yourself, but please recognize the spectrum and make a conscious decision about where you want

to be on it. Set out clearly how OKR-driven work is prioritized against backlogs, BAU, support desk, sales and everything else.



How are OKRs prioritized against other work?

Having worked with OKRs I am in the second school of thought. To my mind, all decisions flow from the OKRs and build towards achieving their goals. Throw away your backlog, incorporate BAU into OKRs and don't do anything that isn't set out in the OKRs.

That might sound hard, but I see it as the power of OKRs. For three months they provide standing orders, a master strategy, agreement on priorities and a reference point for decisions.

One of the advantages of OKRs is that, because they are widely discussed, collectively agreed and shared, they serve to clarify priorities and connect strategy to execution. It is those priorities, the objectives and key results that drive all work: other work sources only dilute this power.

If you lean the other way, then hopefully much of this book is still useful. However you may want to skip some parts.

One word of caution: if your current methods of working are already difficult, if work is overflowing – too much is in progress – then adding OKRs to the mix is not going to help and may make things worse.

OKRs everywhere

Teams live or die by their ability to achieve OKRs. Team members shouldn't be scared of pulling discussion back to OKRs and asking "Will this contribute to

the achieving the OKR?".

If you find that the answer to this question is frequently a 'No', followed by a direct order to do the work anyway, then seek to cover these eventualities in OKRs. If you still find those in authority unwilling to stick to OKRs they have agreed, then you may have a bigger problem.

Once you have agreed your OKRs, make sure they are posted in a prominent place. Don't just list them on a Confluence or Sharepoint page that people have to deliberately seek out in order to read – make them obvious. Write them big. Post them on walls. Put copies on the team board, issue printed versions to every team member – do whatever it takes to make sure they are instantly available.

Bigger team, fewer OKRs

People are frequently surprised when they ask "How many OKRs should a team have?" and I reply "Three, if you twist my arm then four, but really no more than you can count on the fingers of one hand." The shock only increases when I add: "Ideally, there will only be one."

Their surprise is often matched by my own when they say "We have 18 OKRs this quarter."

Sometimes they follow up by saying "We have a big team so we can take on more OKRs." However this logic is flawed.

Remember that much of the power of OKRs stems from their ability to create focus. Obviously having more OKRs means less focus. So too does having more team members: the larger the team, the more views there are on what the team should be doing. It is more difficult for team leaders, including coaches and scrum masters, to focus the team. Having more team members dilutes the responsibility of each individual and increases the possibility of distraction.

When a team is larger, rather than tackle more OKRs per cycle, the team should have fewer OKRs in order to increase focus. While a team of five might be able to work on three (or possibly four) OKRs per cycle, a team of 15 should accept only one or two.

When faced with a larger team with a diverse range of objectives to address I split the team into several smaller sub-teams. A team of 15 might be reconstructed as three teams of five, each of which has its own set of three OKRs per cycle. As a result focus increases, individuals have greater responsibility and outcomes improve.

Sprint planning with OKRs

As with so much else, work begins with the sprint planning meeting – let's start with iterative processes like Scrum, XP and Xanpan. If you are running an iteration-less process such as Kanban then adapt the ideas here to your cadence. The more dynamic prioritization process in Kanban will benefit more from OKRs than Scrum.

OKRs need to be central to each sprint planning meeting. Planning meetings need to include a full review of the current OKRs. Go down the list of OKRs and tick those that are done. The Product Owner should then be able to direct the team to the highest priority OKR. It might make more sense to look at the objective as a whole, or to look at key results one-by-one; either way the Product Owner should give the team a clear statement of what is highest priority.

Avoid the temptation to 'do a little of everything'. Focus instead on one item and bring the collective brain power and energy of the team to bear on that one item. Focus exclusively on one item during the meeting before advancing to consider others – they are, after all, lower priorities. Involve everyone in the conversation.

If it becomes clear that the work to be done on an item during this sprint will not utilize the whole team, then advance to the next. During the meeting focus exclusively on this item until it becomes clear that work will not utilize the whole team. Don't waste time talking about OKRs that are not the focus of the coming sprint.

Aim to advance across a narrow front and achieve some key results before advancing to other objectives and key results.

Avoid pre-work if you can: a little bit of pre-work on objective #2 while focusing the sprint on objective #1 may look attractive, but time spent on #2 detracts from #1.

Agile is a team sport. As far as possible the whole team should be brought to bear on a very limited number of goals at one time – ideally just one.

Aim to work 'short-and-fat' rather than parallel 'long-and-thin' streams: that is – many team members working for a short period rather than having several work streams with one or two team members engaged per stream.

Avoid diluting focus by allowing experts to work on their chosen area. Peter might be the database expert and Rishi the Java expert, but starting two objectives in parallel because 'this one is for Peter and this is one for Rishi' dilutes focus and breaks the team ethos. Even if someone is less productive outside their specialist area, their work can still help deliver the whole objective sooner.

Traffic lights and status

It is useful to mark status against OKRs. This can speed up the review of OKRs, for example at the start of sprint planning, because the team does not need to consider items that are done and delivered.

Teams commonly attach traffic light status (red, green, amber) to objectives and individual results. Red signifies a problem, or that the objective/result will be missed. Green denotes that the team is on course and confident that the objective/result will be met. Amber is something in between: there is doubt or worry.

Personally I prefer a more fine-grained approach to status. The traffic light system does not clearly discriminate between what is green because it has been achieved and doesn't need any more consideration and what is green because it is on course and does require more conversation and work. At the very least a 'no colour' designation is needed to indicate work that has not started yet.

Although I have not had time to experiment here, I would suggest a designation like:

Colour	Status	Meaning
White (Clear)	Not started	
Yellow	Started, on course	Work in progress, confidence high
Green	Achieved	No more work needed
Red	Troubled	Work begun but problems encounter: time running out or technical issues?
Purple	Abandoned	Team has accepted the goal will not be achieved, no more work needed

Summary

- OKRs are it. Close your mind to everything else.
- The team exists to deliver OKRs: everything else is secondary.
- Find a simple mechanism to show the status of OKRs.
- Work short-and-fat through OKRs rather than salami-slicing them to work long-and-thin.

17. OKRs and the backlog

OKRs, not backlogs

Backlog first

OKRs first

Return of the sprint goal

Summary

18. BAU – keeping the lights on

Option 1: suppress BAU

Option 2: reduce or remove BAU

Option 3: make BAU better

Option 4: objective zero – add BAU

Downside

Summary

19. Executing

Keeping focus

Prioritize

Visual display

Revisit often: sprint planning

Time-slice

Summary

20. Going off-piste

> *"I wish it need not have happened in my time", said Frodo.*
>
> *"So do I", said Gandalf, "and so do all who live to see such times. But that is not for them to decide. All we have to decide is what to do with the time that is given us."*
>
> J R R Tolkien, *The Fellowship of the Ring*

If OKRs are to be effective it is necessary to measure all work against them: don't get out of bed in the morning if it won't move you closer to achievement.

OKRs can and should be a reason to say:

> "I'm sorry Dave, I can't do that: my objective is to investigate the anomaly and the first key result is to put the ship into orbit."

In other words: OKRs are a shield that can be used to deflect those who would distract and obstruct your progress.

However, OKRs should not be a reason to act immorally, unethically or negligently, or to ignore events, changes and crises around you. Sometimes the right thing to do is to say:

> "This is not in our OKRs but it needs attention. OKRs must take a back seat while we go off-piste. We will do it and work out how we pick up with the OKRs at the end."

Working only for OKRs is wrong, but so too is being too flexible and bending to every change and request. Teams need to find their own sweet spot somewhere between these extremes.

Writing this six months into the Covid-19 pandemic offers an obvious example. Teams that clung to OKRs without thought as workers were sent home, schools closed, internal travel ceased and countries locked down may well have been pursuing the wrong goals. Yet at the same time, some teams will have found their goals unchanged despite Covid once new working patterns emerged.

While many teams will have found themselves blown off course by the events of 2020, others will have found that having clear agreed goals provides stability in a turbulent environment.

Agonizing every day about whether the team is pursuing the right goals is a waste of time and energy. Equally, clinging to goals while fires burn and the world changes is wrong.

Sometimes a team needs to say:

> "Let's go off-piste, let's do what needs to be done. When things calm down we will regroup and assess where we are; which OKRs still make sense and what new priorities replace current goals."

It might be that when the crisis has abated the team can return its focus to OKRs, or at least a subset of them. Or perhaps the crisis has changed the world and OKRs need resetting. (Scrum aficionados may see a parallel here with the little-used *abnormal termination of sprint*.)

Unplanned but valuable

While OKRs can be a powerful means of keeping a team on track and reducing diversions and disturbances, some distractions are worth embracing, and not just because they are world-changing pandemics.

It should not be a question of 'planned work good, unplanned work bad', but 'what is valuable?'. If some unplanned work arises, the default position should be to refer to the OKRs and refuse it if it is not covered. But knee-jerk reactions need to be moderated: one should always listen to a request and consider whether it has value.

There is inevitably tension here, and both extremes are wrong: if OKRs are to mean anything, then they must mean that teams can turn down incoming work. But turning down all work because it does not build toward existing OKRs is equally wrong.

Ultimately the decision of whether to divert off-piste or to stay focused on OKRs will be a judgement call. There are no hard and fast rules on the right course of action, so all I can do is to make some suggestions:

- Is the thing being requested valuable in its own right? Is there reason to believe that it is more valuable than the current OKRs?
- Does the person making the request appreciate the consequences of doing this work and the potential knock-on effects?
- Is there time to consult other team members about the request? In particular, what does the Product Owner think?
- If you need to make a decision and there is no time or access to consult with other team members, ask yourself "What will others think?" and "If I do this, will I need to defend my decision, or will others agree?".

In the days when teams still used physical boards, I was known to write out the request on a card, take the card and requester to the team board and show them how their work request would impact others.

While any given request might be small, such as a bug fix a developer feels can be done in five minutes, one has to remember that even if it only takes five minutes, it represents a far bigger loss of time once mental context-switching time is added in. Then there is testing and release time to consider. Do not forget possible ripple effects and the risk of the fix going wrong.

Finally, remember that if one regularly agrees to 'small work', requesters will consider that to be the norm and will continue to ask for it. When given enough 'small work' that 'can just be squeezed in', then little of the strategic objectives will be achieved.

Prepare for the unexpected

One can never prepare completely for the unexpected, but teams can do some mental preparation, such as fostering a shared perspective on what should be accepted and what should be refused. While the Product Owner may be the one with ultimate authority, there are always circumstances in which individual team members will need to make a quick decision.

During OKR-setting teams might ask themselves to think of examples of work which, should it arise, will have priority over the OKRs. Similarly, the team could think of examples of work that would be pushed back and refused.

Such discussions can form part of team retrospectives. Teams might, in a non-threatening way, examine past decisions on unplanned work and consider whether the best decision was made. *Was the team right to refuse some requests? Were the accepted requests justified? What would have happened if a different decision had been taken?*

Track distractions

One way to approach the problem of unplanned but urgent work is to create a feedback loop, in true agile fashion. It might not save the day today, but it will help prepare for next time.

At the end of the quarter, when evaluating your OKRs and performance with stakeholders, ask them: would you rather the team had responded to fires and delivered fewer OKRs? Or should the team have focused even more and let others deal with fires?

In Xanpan[1] I describe how teams can track unplanned but urgent work in the same way that they visualize and track work planned in planning meetings. Basically they write out a new card – usually a yellow one – and put it on their board (physical or online) and treat it as any other piece of work.

If the work is really urgent it goes straight to 'work in progress', even if that displaces existing work. If it can wait a little while it goes into the 'to do' column

[1] https://amzn.to/2MhFBm3

until the next person becomes available. If it can wait a week or two it simply goes into the backlog for prioritization in the next planning meeting.

Importantly, by tracking the work teams can understand the nature of disruptions and quantify how much 'unplanned but urgent' work is asked of them – perhaps drawing a graph of requests per sprint. Over time the team can use this data to reason about the work: *Should they allow capacity in each sprint for unplanned work? Should they talk to specific people and ask them to submit requests before the start of the sprint?* Or maybe they need to remedy part of the system that generates unexpected work.

Teams using OKRs can do the same thing: count and track late-breaking requests that don't fit with the OKRs. This won't solve the problem immediately, but as the data grows the team will be able to reason about it and decide the best course of action.

For example, if the team regularly finds there are urgent BAU 'keeping the lights on' work request that fall outside the OKRs and cannot be ignored, then they may decide to adopt an objective zero to ensure such work is recognized.

Summary

- There are times when clinging to OKRs in the face of change is wrong. It is better to go off-piste. This is a judgement call.
- If you go off-piste, regroup later and assess the impact on the OKRs.
- Cultivate shared thinking on prioritization outside OKRs, think about what might happen and learn from what does happen for next time.
- Track work that doesn't relate to OKRs, understand where the work originates and how it effects the team. Then decide what to do.

21. Beyond the quarter

Three horizons

Rolling roadmap

OKR roadmaps

Feedback

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

22. 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 you are fast enough, as long as you listen to what customers are asking for now and then deliver it, and as long as you keep the system running and fix any defects the moment they are seen, then you'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, they also want other things, things that are even higher priorities.

OKRs are both the product of strategy, because strategy informs which OKRs are set, and the mechanism through which strategy is delivered. OKRs can play the role of 'strategy debugger', because they make strategy visible. When OKRs don't support strategy it is a sign that something is wrong, perhaps because strategy has not been communicated, or perhaps because it is absent.

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', 'purpose', '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 behaviour over time[1].

[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.

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 behaviour 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.

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*.

As originally described, the idea of strategic intent fits well with OKRs. Rather than starting with the here and now and extrapolating resources and capabilities, strategic intent focuses on the goal, the outcome, the desired future[a].

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.

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. But when there is *strategic intent* then the decision-making process is different.
>
> Knowing the goal, the *strategic intent*, informs such decisions even without a plan. The muffin is rejected.
>
> [a]*Strategic Intent*, Gary Hamel and C K Prahalad, Harvard Business Review, May–June 1989

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.

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.

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.

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.

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

What gets delivered is the result of a thousand small decisions. Without a strategy to guide the decision-making process, those decisions will lack cohesion; the final product – and code base – will lack 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.

Nor is it just backlog management that can benefit from a strategy. Teams themselves can benefit from having strategy and shared goals to focus thinking. Clear strategy and shared goals allows individuals to coordinate and align their efforts of these team members and share work more easily.

The software design benefits too. Strategy and goals inform design and refactoring decisions to create a more coherent design.

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 manageable, 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.

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.

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.

23. Leaders

Culture, goals and strategy elements

Day-to-day

Leaders and culture

Bottom-up more than top-down

Summary

24. Culture

Delivery culture

Customers

Openness and feedback

Psychological safety

Ambition

Summary

25. Leaders and planning

Broad–narrow

Forward planning

Cascade up, not down

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

26. Aspirations

> *Our problem is not that we aim too high and miss, but that we aim too low and hit.*
>
> Variously attributed to Artistotle, American motivational speaker Les Brown and others

Most of the accounts of OKRs emphasize their aspirational nature. While the aspirational attributes of OKRs are highly desirable, there is significant value in using OKRs even if your aspirations are a little more mundane.

Used in *aspirational mode* OKRs are 'moonshots'. OKRs motivate teams to achieve '10x' (that's *ten times* to most of us) performance. It could be 10x team effectiveness, 10x product impact or both. In aspirational mode OKRs aim higher than the team believes it can achieve. The organization accepts that OKRs will be missed, that teams will *fail*. In fact, teams are *expected* to fail their OKRs.

The underlying assumption is that a team aiming to achieve a tenfold improvement – say boosting website views from 1,000 a day to 10,000 a day – may miss its target. But in aiming ridiculously high, the team will outperform a more modest team that aims low and achieves its goal.

Failing to meet a tenfold improvement target – say achieving 5,000 views per day – will still be a greater improvement than a team that plays safe. A team that is playing safe may aim for a 10% improvement – say to raise views per day from 1,000 to 1,100 – and may well meet its goal.

The outcome-oriented nature of OKRs implicitly recognizes that labels like 'success' and 'failure' are less important than the result achieved. There is however a conflict hidden in this approach: a 'failure' can be a better result than a 'success'. This can create cognitive dissonance for both team members and those managing such teams.

An outcome can have both success ("We raised website views five-fold!") and failure ("We missed our OKR!"). As is often the case, the labels *success* and *failure* are applied after the event and depend on perspective. On the face of it OKRs are objective (you hit or miss) but subjectivity is never far away.

Utility mode

Alternatively the OKR mechanism can be used in *utility mode*. Even without 10x aspirations there are still benefits from having shared team OKRs. These have already been outlined, but are with repeating:

- Promoting an outcome orientation and business benefit.
- Prioritizing work to be done.
- Increasing focus by de-prioritizing potential work, thereby allowing focus on remaining work.
- Sharing goals across the team.
- Communicating team goals to the wider organization.
- Providing medium-term planning.
- Clarifying targets and objectives.
- Creating context for technical and business decisions.

Even without aspirations OKRs have plenty to offer. There is benefit in using OKRs even when used in a conservative utility mode. For those adopting OKRs for the first time, this approach sidesteps several potential pitfalls.

Predictability

Aiming high and accepting 'failure' sounds good and aspirational. But organizations value predictability; aiming high and accepting that you might miss doesn't sit well with those who want certainty. I remember one set of stakeholders who became agitated when told a team only expected to achieve 70% of planned OKRs.

Utility mode OKRs can help here too. Aiming high when stakeholders and the organization value predictability is probably not a good tactic. As a tool OKRs can still help and still deliver benefits. You might want to rate each OKR on a scale of 1 to 10, where 1 is unlikely and 10 very likely. Show your stakeholders, and if they are unhappy rework the OKRs so that they fit an acceptable risk profile.

To complicate matters, unfortunately organizations and even individuals are not always consistent. While one leader advocates aspirational OKRs, and even claims to accept failure, others might demand certainty. In such cases you might want to bring divergent thinkers together and outline the mismatch.

Creating aspirations?

It would be naive to claim that adding OKRs to any team will miraculously turn it into a high-performing aspirational one. Naturally I would love it if this were the case, but 'just add OKRs' is not as simple as 'just add water'.

OKRs are certainly one tool for nudging a team towards higher performance and greater aspiration, but they are not enough on their own. Promoting high performance and aspiration requires a supportive environment and culture – in other words, *psychological safety* (discussed earlier).

Many accounts of OKRs focus on the aspirational nature of OKRs in Google and Intel. As such these accounts say much about the culture and approach of highly successful companies. Entire books have been written about these companies and how they foster such a culture, so I will only point out a few elements:

- A 'safe to fail' environment.
- Motivated individuals.
- Opportunity and a resource-rich environment, with a willingness to let motivated individuals fail.
- Evaluation and reward systems that recognize failures as being equivalent to successes.

Even Google and Intel will fail on some of these points – and those who know these companies from the inside may see more inconsistencies. But even as they

fail on some points, they succeed on enough points to perpetuate a culture that values aspiration.

Once a company has such a culture the whole thing becomes self-perpetuating (indeed, all cultures tend to become self-perpetuating, for better or for worse). Individuals who value these attributes will want to work at such places, while those who don't share their values will go elsewhere. Peer pressure and individuals' desire to fit in will become self-reinforcing.

Leaders and culture

For companies that want to adopt aspects of aspirational culture – what might be called *Silicon Valley culture* – there are formidable obstacles. OKRs may well form part of that change, but they are not alone.

In particular, companies need to recognize that the people they employ are, almost by definition, different to people who work in companies with an aspirational culture. The existing company culture will have already filtered out some of these aspirational individuals.

Those who are employed have proved themselves compatible with the existing culture; that culture will have rewarded them for working within it and punished them for not doing so. Over time it will filter out people who are incompatible. Switching to an aspirational culture takes more than flipping the 'OKR light switch'.

Company leaders usually recognize this, but often fail to comprehend how their own actions are seen. A leader can stand on stage and tell their workforce passionately about the change they want to see, they can articulate OKRs in detail, and they can truly believe what they say. But workers have often seen this before: company change programmes come along regularly. Leaders frequently want to change something. Workers are almost programmed to be cynical.

As a result workers listen and watch. They look to see if the leader is *walking the walk* or just *talking the talk*. The cynical amongst them believe it is all 'management talk'. Any action the leader takes – any displeasure they voice, any incongruent set of actions, let alone anger or punishment meted out – will quickly be seized on as proof that they do not mean what they say.

An OKR adoption route

If you work in a high-performing aspirational company, then great. Adopt OKRs and everything will be even better.

For everyone else, let me make a suggestion.

Aspire to aspirational OKRs, but make the change in small steps.

A psychologically safe environment is critical for aspirational OKRs to work. So do a quick assessment, and if you are anything less than 100% sure, start the conversation about what is needed and ask for help. If your objective is success with OKRs, then the first key result is a boost in psychological safety.

Start by using OKRs in *utility mode*. Better still, perform the exercise I describe next; you might find you are not quite as 'utility mode' as you think. But wherever you start, just start.

Start setting OKRs on a regular basis. Work towards them and get better at setting and delivering against your OKRs. Inject safety into the system before teams are ready to use it, so that they can 'take up the slack' when they are ready. With time the team may be willing to take on more risk and be more aspirational.

While the team is perfecting its use of utility-mode OKRs, work on the rest of the company. My guess is that if you are adopting OKRs, you are not alone. Other teams may be adopting them, while more senior people in the company may want them to be adopted. Work with the grain and nudge these people in the right direction.

As you do so, the problems of using OKRs in an unfriendly environment will become clearer. Work on these issues. Use OKRs as a problem detector to find out what needs to be changed. Don't work alone, work with others who are adopting OKRs to nudge people and the organization in the right direction.

Work on yourself: indeed work to change yourself more than anyone else. Keep an eye on your language: 'success' and 'failure' are loaded terms. Make sure your actions fit with your talk: you are a leader too and you also need to 'walk the walk'.

Involve your personnel and human resources staff: ask them to observe your OKR-setting and ask for their thoughts. Show them the outcomes: working software. Keep talking to them.

Above all else you need to work out how OKRs relate to your performance appraisal programmes and salary reviews. The simplest advice is to keep OKRs separate from these annual checkpoints, but no organization seems to follow this advice.

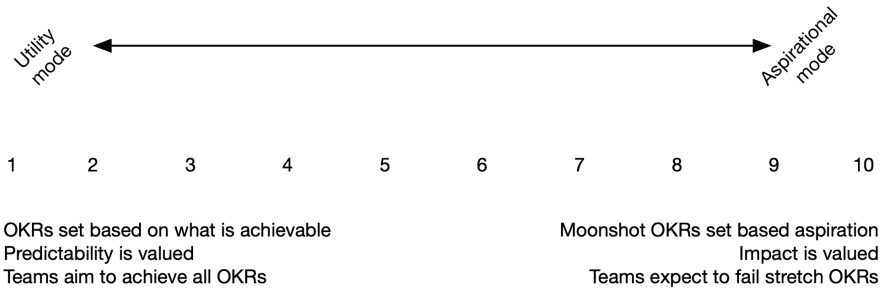
Progress to conversation with your superiors and the people to which the team answers. Some discussions are better held in the open with other team members, while others are better held in private with superiors.

This isn't a comprehensive list of suggestions, it isn't even a long list, but it is a starting point.

Exercise: where are you?

Imagine a line of numbers from one to ten. At one end 'one' is labelled *utility mode*. Here OKRs are used for team cohesion, shared understanding and medium-term planning. Achieving the objectives and key results is important.

The other end of the line, 'ten', is labelled *aspirational mode*. At this end teams are stretching themselves, aiming for 10x solutions and shooting for the moon. The team, those around them, and importantly management, recognize that OKRs may be hit or missed. The real evaluation is the outcome of the work of the team.



Utility mode is one end of the spectrum, aspirational mode is the other

All ten points on this spectrum are respectable places to be. Each position represents a legitimate way of working.

Ask yourself: *where would you put your team on this spectrum?*

The position you choose will naturally reflect your own ambitions and aspirations. It will also reflect the environment you work in: high-risk high-reward start-up or low-risk modest-reward legacy bank.

Knowing your appetite for risk will help you when setting OKRs.

More importantly, you will want to agree this position with your team. After all, the whole team is signing up to a set of OKRs, so it is important that the whole team understands the context. Don't impose your position on the team, so ask members where they think the team should aim.

Start by drawing the line on a board. Mark it 1 to 10 and talk about what each extreme implies. Then have every team member write down the number that reflects the position they think the team should occupy. Wait till everyone has written down their own private number, put the papers in a pile and shuffle them to keep answers anonymous. Then mark the numbers on the 1 to 10 line.

Discuss the results. Maybe the whole team agrees, which is great. If not, ask why some people might vote low and some high. Work through the reasoning and decide on a shared position.

Summary

- Advocates of OKRs usually emphasize their aspirational nature. While aspirational OKRs can be immensely powerful, they only work when the organization has a compatible culture and provides psychological safety.
- The other benefits of OKRs justify using them in utility mode: to create focus around shared priorities, promote medium-term planning, communicate direction and more. Such utility OKRs can have benefits even if it is only your team using them.
- OKRs can help where a company is attempting to transition to a more aspirational culture. However they are not enough on their own. More needs to change to make the organization and culture aspirational.

27. Everyday pitfalls

‘OKR buffet’

Late-arriving OKRs

Adding to the story hierarchy

Counting problems

Respect for specialists

Respect for managers

Summary

28. Trouble with targets

Targeting the measurable

Questions measurement can’t answer

Goodhart’s Law

Goal displacement

Overcoming tunnel vision

A final warning: targets

Summary

29. Individuals and performance reviews

Integrating employee reviews with OKRs

OKRs for individuals

Summary

Close

Please review

If you enjoyed *Succeeding with OKRs in Agile* please consider leaving a short online review[1] on your favourite site to help others.

The author will be very grateful!

[1] https://amzn.to/47GVDki

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

The first edition of this book attempted 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. The second edition refine those learnings and shares a few more.

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 working with OKRs intensively for a year I still had my doubts, but I could 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.*

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 your 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 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 (once 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, second edition, 2023

Print: 978-1-912832-25-5 Amazon

Print: 978-1-912832-30-9 second source

Electronic: 978-1-912832-26-2 ePub

Electronic: 978-1-912832-27-9 PDF

Audio: 978-1-912832-31-6

First edition, 2021: 978-1-912832-06-4 (print), 978-1-912832-08-8 (ePub)

Further reading

OKRs

Measure what Matters, John Doerr, 2017

5 Ways Your Company May Be Misusing OKRs, Itamar Gilad, https://itamargilad.com/5-ways-your-company-may-be-misusing-okrs/, accessed July 2020

Measuring

How to Measure Anything, Douglas W Hubbard, 2010, John Wiley and Sons

Competitive Engineering, Tom Gilb, 2005, Elsevier Butterworth–Heinemann

Problems with measurement and targets

Goodhart's Law, Charles Goodhart, https://en.wikipedia.org/wiki/Goodhart's_-law, accessed July 2020

Obliquity, John Kay, 2011

The Tyranny of Metrics, Jerry Z Muller, 2018

Agile and teams

Right to Left: The Digital Leader's Guide to Lean and Agile, Mike Burrows, 2019

Amoeba Management, Kazuo Inamori, 1999

Xanpan: Team Centric Agile Software Development, Allan Kelly, 2014

Continuous Digital, Allan Kelly, 2018

Management

The Fearless Organization: Creating Psychological Safety in the Workplace for Learning, Innovation, and Growth, Amy Edmondson, 2018

Simply Managing, Henry Mintzberg, 2013

The Rise and Fall of Strategic Planning, Henry Mintzberg, 1994

Hypothesis-Driven Development, Barry O'Reilly, https://barryoreilly.com/how-to-implement-hypothesis-driven-development/

Lean Startup, Eric Ries, 2011

OKRs extra - coming soon

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[2] – coming soon on LeanPub. Register your interest today and be the first to know.

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

Acknowledgements

First edition

Thanks to Inga Wassenhoven, Dez Conner, Bjorn De Wael, Ramage Marzden, Inge Gordon and Frederik Van Herterijck for letting me be part of their OKR journey. Double thanks to Inga and Dez for deep discussions on the nature of OKRs and hours spent drafting and redrafting potential OKRs.

Thanks to Mike Burrows for comments and suggestions on a very early draft of this book, and thanks to an almost anonymous reader, Sam, who sent me feedback via LeanPub.

Thanks too to everyone who bought the early versions of this book, for providing the monetary feedback that showed people would find it interesting and that I should keep writing.

Second edition

One can learn so much from questions. The questions I've had since the first edition have helped me improve this book.

So thank you to everyone who has asked a question: about this book directly or on social media, in OKR consulting or training sessions, after presentations (especially *Reawakening Agile with OKRs* and *Honey, I Shrunk the Backlog*.)

And thank you to everyone who has given me the opportunities to get those questions: by invited me to speak, advise, consult or just talk about OKRs.

Also by Allan Kelly

Books to be Written: a non-fiction author's how-to guide to writing, publishing and marketing[3], Software Strategy/LeanPub, 2023

The Art of Agile Product Ownership[4], Apress, 2019

Continuous Digital: an agile alternative to projects for digital business[5], Software Strategy/LeanPub, 2018

Project Myopia[6], Software Strategy/LeanPub, 2018

Little Book of Requirements and User Stories[7], Software Strategy/LeanPub, 2017

Xanpan: Team Centric Agile Software Development[8], Software Strategy/LeanPub, 2015

Business Patterns for Software Developers[9], 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[10], John Wiley and Sons, 2008

[3] https://amzn.to/3DwgCsK
[4] https://amzn.to/3spqRH4
[5] https://amzn.to/2CUbMbW
[6] https://amzn.to/2wZW9JM
[7] https://amzn.to/2P6VB1K
[8] https://amzn.to/3sj0b0c
[9] https://amzn.to/3mRwZXr
[10] https://amzn.to/3sfinI5