CONTINUOUS DIGITAL

An agile alternative to projects for digital business

Allan Kelly Associates

Continuous Digital

An agile alternative to projects

Allan Kelly

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

This version was published on 2021-02-17

ISBN 978-0-9933250-8-3

Tweet This Book!

Please help Allan Kelly by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I'm reading Continuous Digital - the agile alternative to projects

The suggested hashtag for this book is #NoProjects.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#NoProjects

Contents

Free Book . . . . . **i**

What others are saying... . . . . . **ii**

Project Myopia . . . . . **iv**

Preface . . . . . **v**
Digital vi
Agile and continuous delivery vii
Products, not projects viii
Once upon a time... ix
And now for something completely different... ix

I An alternative **1**

1. Omnipresent software **2**

2. Software as an Asset (SaaA) **8**

3. Higher purpose **9**
Not money 10
Obliquity 11
Organizations and teams 12
Scale 13
Mutable 13
Individuals 14
Project completeness 15
Closure 15
Finally 16

4. Team-centric development 17
5. Work to be done 18
6. Value 19
7. Summarizing the alternative model 20

II Interlude 21

8. Diseconomies of scale 22
9. Diseconomies and risk 23
10. Living with diseconomies 24
11. Schedules 25
12. Time-value profiles and elastic deadlines 26
13. I need it yesterday! 27
14. Theory X, Theory Y and strategy 28
15. Planning 29
16. Piecemeal growth 30

III Teams 31

17. Devolved decision-making 32
18. Team strategy 33
19. Stable teams 34
20. Team lifecycle 35
21. Team lifecycle – another example 36

IV Money **37**

22. Real options and venture capital **38**

23. Continuous governance **39**

24. Budgeting **40**

25. Assets and accounting **41**

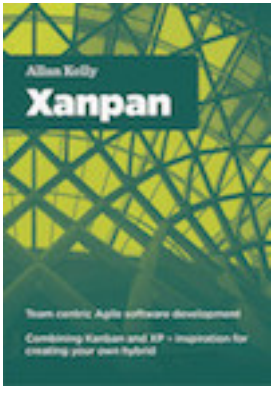
26. Money trouble **42**

Like to read more? **43**

About the author **44**

Also by Allan Kelly 44

Free Book



Xanpan is available for free

Xanpan: Team Centric Agile Software Development is available for free to all subscribers to Allan Kelly's newsletter[1]

[1] https://www.allankellyassociates.co.uk/xanpan_offer/

What others are saying...

'This book is terrific! It accurately diagnoses a problem with our model of software delivery and charts a new course. Well researched and well argued it shows the future.'

Kristian Kristensen, VP of Engineering, Ecommerce at The New York Times

'Over the past few years Allan has done a huge amount to bring #NoProjects into mainstream IT conversations. This book is a deep dive into #NoProjects, and I highly recommend it.'

Steve Smith

'The great thing about Continuous Digital is the plain sense that it all makes, so you realize this is exactly the way you want to work in the future. The bad thing about the book is that it doesn't include a Tardis to go back and start working in this way when you set out on your last 'project'. So bury that pain and just look forward, with this book to navigate by."

Steve Parks, CEO, Convivio

'What Allan has created here is nothing short of groundbreaking. This book is an insightful vision of the future of work that is emerging within organizations around the world. Be it #NoProjects or Continuous Digital, the movement towards funding and working along continuous value streams rather than finite projects is reaping benefits.

Beyond the vision, what Allan has written will provide you with a clear and practical grounding in how to get started across your organization. Whether you lead a team, a division or a company, you will learn something new about creating value.'

Evan Leybourn, author of *Directing the Agile Organization*

'Careful! Not only does this book offer you plenty of food for thought, it is practical enough to be your handbook for transforming a business into a digital one. Without resorting to boring management-speak, Allan explains how to take an organization to the next level by putting customers first, focussing on value and building effective feedback loops to emphasize learning.'

Sergey Timanin, Infrastructure Team Lead at Clarivate Analytics

'Those with an engineering background can sometimes feel it in the air when limits are reached too easily in a process. Then you start to get some intuitions, then you try to tackle

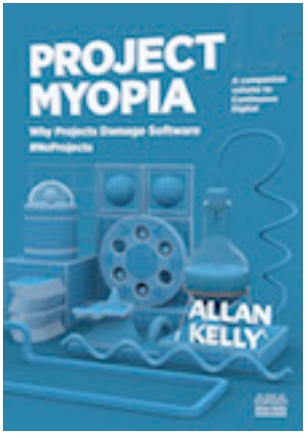
some points. But if you have the chance to read this book, you can have the great feeling to be in an accelerator.

Allan Kelly is not only able to identify the recurrent issue that other people don't want to see. He is also able to structure and use the exact word on concepts that will make you more confident in your daily management process.

In this cookbook, many solutions are proposed for your organization and it's up to you to do the right thing in your context. I warmly recommend you to read this book today as the golden age of pure project agility is about to end.'

Gwendal Tanguy, Director of Product Development, Swissquote Bank

Project Myopia



Project Myopia: A #NoProjects book

> *Project Myopia*: the belief that the project model is the only way of managing business change and development; not seeing digital development as a continuous commitment to growing the business; believing it will end and working for the end. #NoProjects

Project Myopia was the beginning of *Continuous Digital*. But Continuous Digital outgrew Project Myopia and the two seperated.

Preface

> *Practical men, who believe themselves to be quite exempt from any intellectual influence, are usually the slaves of some defunct economist.* John Maynard Keynes

Keynes was speaking long before Digital and long before there was a software industry, but his words still ring true. In the world of digital business too many people cling to the old model and assumptions of the pre-digital age, an age when *information technology* was simply a means of making a business more efficient.

As this still-young century continues, it has become clear that the business world is increasingly a digital world. The software companies of the last century are but the prototypes of the modern digital business.

Digitization forces companies to rethink the way they organize to do software development, technology operations and business alignment. Digitalization also means new management paradigms need to be embraced and some old ones let go.

Continuous is an alternative management model for digital businesses – a management model, a new way for managers to think about technology. This book sets out that model.

Many practical men and women are today slaves to the project model of software development. The project model condemns many digital initiatives, programmers and companies to failure. Continuous offers an alternative.

The project model has always been a problematic fit for software development, but things have become more difficult in the last few years. The Continuous model supersedes the project model. Projects by definition end, but successful business doesn't end and successful software doesn't end. Agile and continuous delivery provide the tools for successful digital business.

The business is technology, and technology is the business.

When the business is based on technology, 'finishing' the technology destroys the businesses. The mindset encouraged by the project model – that software can be created as an independent activity and 'finished' – is anathema to digital businesses, where building technology is business building.

The rise of agile software development has made the problems with the project model more apparent. The arrival of Continuous Delivery[2] (CD) broke many of the principles on which the project model was based. Digital business demands a new way of thinking.

Digital

Few businesses have not felt the effects of increased digitization in the last decade. Modern business is digital. Fewer still are the number of high-growth businesses that are not dependent on software technology. If you want to be a high-growth business in the twenty-first century, you almost certainly need to build on top of a digital platform.

The digitization of business represents a force so powerful that few businesses are unaffected. Unleashing this force in your business creates massive opportunities: not unleashing the force creates massive threats. To some this will sound counterintuitive. Flexibility, according to this line of thinking, comes from being able to assemble a team when needed. Agile thinking postulates the opposite: flexibility comes by having a well-practiced team that is available when needed.

Information technology is no longer a side-show, something that happens off to the side of the real business. In the age of digital business, technology *is* the business, and the business is digital. Having a separate IT function is a luxury few growing companies can afford.

In the digital company technology development isn't something that happens elsewhere, bounded within a 'project' that will one day be 'finished'. The business has technology running all the way through it, like traditional English seaside rock. If this is not true in your company, then ask yourself and all those around you, if you foresee the business growing in the next decade?

Next ask yourself and your colleagues: "If a competitor wanted to disrupt our business, how would they do it?". The answer is – Digital.

It follows that not only are businesses digital, but business lines within businesses are digital. If a business line is to succeed and grow, the business line must be tied to an information technology capability, and that capability must be ongoing. Structuring technology change as a series of stand-alone 'projects' no longer makes sense. Projects end, but who wants their business line to end? Who wants their company to end?

[2] *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Humble and Farley, 2010

Agile and continuous delivery

Before agile many companies and teams considered a three-month release cycle as aggressive; an annual release was more common. Agile started to ask teams to make fortnightly releases, or at the very least be able to release every two weeks. Being *releasable* was more important than actual releasing, because staying at a releasable level forced teams to keep a high technical standard and thus gave options to business customers.

The best digital teams have now accelerated far beyond every two weeks. The Guardian newspaper in London commonly releases several thousand times a week and Amazon.com releases on average every 11 seconds.

A release every two weeks still looks like an impossible challenge to an average team – typically one working inside some large corporation. But for the best teams a two-week gap between releases looks like an eternity.

Regular releases well and truly break the link between project end date and software release. The parameters of date, scope and quality at the heart of the project model are managed completely differently.

Agile had already relaxed the idea of fixed – or at least predetermined – scope, such as requirements or work requests. In a continuous delivery environment teams build all sorts of feedback loops. When done well these feedback loops cause work to be added – that is, teams discover more valuable work – and other predetermined low-value work can be discarded.

Unlike some endeavors, the relationship between quality and time/cost is reversed in software development. Higher-quality (code) is delivered faster and at a lower cost than low-quality code that requires fixes and is hard to change.

Project managers are trained to see a tradeoff between higher quality and speed of delivery: *reducing quality reduces time*. But in software development the relationship is inverse – higher quality requires less time. Those schooled in the project model have been making things worse by advocating lower quality to reduce time.

The best agile teams were always multi-skilled and cross-functional. Rapid continuous delivery is simply not possible when organizational boundaries force teams to wait on others. As a consequence the idea of project teams working with specialist parts of the organization needs to change.

Such teams need to practice together, work together, adjust behavior and fine-tune their work. It no longer makes sense to create special, temporary teams for projects. Teams need far more continuity. Each piece of work is a rehearsal for the next piece of work.

To some this will sound counterintuitive. Flexibility comes not from being able to assemble a team when needed, but by having a well-practiced team available when needed.

The project model always had limitations when applied to software development. The project model set out an approach that did not accurately reflect what actually happened in software development. As a result traditional project managers spent much of their time patching the model to keep it relevant.

When one has an inaccurate model of the world it is hard to make good decisions and reliable predictions. Iterative agile presented an alternative model of the world within which it became easier to make reliable decisions.

Continuous Delivery makes the project model an even more inaccurate model of technology development. Once managers and teams have a mental model that correctly describes the work they do they can reason more effectively, become more effective and deliver greater value.

Products, not projects

One frequent riposte is the claim that there are two types of company: *project-oriented companies* and *product-oriented companies*. Project companies do something, finish it and move on, while product companies continue to build their products.

Imagine for a moment you are the Product Manager for Microsoft Office. One day you walk into Satya Nadella's office and say "Good news, project Microsoft Office is done, we can lay off the team and save some money." Think what that would mean.

Microsoft's second biggest cash-cow is dead.

Microsoft is a product company: Office, Windows, X-Box and much more. Unfortunately many product companies have succumbed to the project model: they do one project on their product, then another project on their product, then another, until the end of time. Product companies applying the project model delude themselves and make their work more complicated.

Software product companies can be considered the prototypes of modern digital businesses. Like modern digital businesses, their success depended on their ability to build software technology. But being Digital is much more than being a software business as of old. Being Digital isn't confined to *the product*: being Digital cuts across everything.

There is much digital companies can take from the product model and mindset, but they need to go further.

Once upon a time...

Most organizations are structured along pre-digital lines – functional divisions and cross-cutting projects. Such thinking, including the project model itself, dates from the 1960s. Software people call this model 'waterfall' and date it to a 1970 paper[3]. Let's remember 1970 for a moment; I was still in nappies, but…

In 1970 the IBM 360 Model 195 was the cutting-edge mainframe of its time. This machine had 10 MIPS of CPU power, 4Mb of RAM and cost $250,000 a month to rent (about $1.75m in 2018 terms). Commercial programming was in Cobol against an IMS hierarchical database running on OS/360. It used green screen monitors and teleprinters for output.

By 2016 one could buy a Raspberry Pi computer for about $35. This has 4,744 MIPS, 1Gb of RAM (which is 1,024Mb, so 256 times more than the IBM), and runs an OpenSource Linux OS that can be programmed in languages such as Python, Scala, Ruby, C and even Cobol if you so wish. The results can be displayed graphically on your TV screen. Plus a Raspberry Pi can be connected to an Internet of several billions nodes. In 1970 there were less than a dozen Internet nodes.

When technology has changed so completely, is it not right to question the management practices and models that grew up at a time when CPU cycles were expensive?

In 1970 planning was cheap in relation to expensive CPU cycles. In 2018, with CPU cycles too cheap to measure, planning is extremely expensive.

And now for something completely different...

This book sets out to describe an alternative to the project model for developing software. Why do we need an alternative?

Because digital competition changes the way businesses need to operate.

Because the tools digital makes available allow different structures and new thinking.

Because the project model complicates reasoning about software development, and is holding back further improvements in software development in many organizations today.

Because digital companies need to rethink the traditional ways of managing and find new, effective, models to reason about their world. Management models developed in the 1950s and 1960s are not necessarily the best for leveraging modern technology.

[3] *Managing the Development of Large Software Systems: Concepts and Techniques*, Royce, 1970

Paradoxically, in the age of hyper-speed and rapid change the project model is too short-sighted. In the digital age acting fast, in ultra-short timescales, requires thinking and structuring for the longer term.

Adopting a Continuous model rather than a project model does not mean abandoning everything in the project model. As with so many models – agile in particular – it is necessary to look beyond the label and consider exactly what practices and processes the label refers to. Some of these practices and processes may be entirely appropriate in other models.

For example, some regard planning, objective-setting and governance as inherently part of the project model, but they are not. As this book will describe, planning, objectives and governance can be a meaningful part of the Continuous model.

Yet there is still much in the project model that needs rethinking for digital businesses to advance. Creating projects, assembling teams, waiting for the team to become productive and then dismantling the whole thing before starting again takes too long. Structures with longevity and devolved authority eliminate the start-up and shutdown phases that take so much time.

I will do my best in these pages to outline an alternative model, a model that I call *Continuous*. This model is a work in progress. There are examples of companies that work without projects, but that does not mean that your company can simply copy these as they stand.

The Continuous model is still evolving. It may never reach a point at which there is a fixed model – in fact I don't want to reach such a point. I want practitioners, managers and my readers to think!

I do not have a guaranteed risk-free solution. I have pieces of a solution that you need to examine for yourself and apply in your context. For this I recommend thinking, reflection and discussion.

Being an early adopter of this model may not be risk-free, but the benefits are substantial. Remember the old maxim: *profit is the return for risk*. Early adopters may risk more, but they will benefit more.

I An alternative

It's in the doing of the work that we discover the work that we must do. Doing exposes reality. Woody Zuill

1. Omnipresent software

> *Software is eating the world.* Marc Andreessen, 2011

Here at the beginning of the twenty-first century every company is a software company.

If that sounds a little overblown, let's agree: every *growth* company is a software company. Companies depend on software to deliver products and services that would not be possible with it. Companies that want unique products and services increasingly find that this means unique software. Anyone can have identical software – ctrl-C, ctrl-V – but identical software leads to identical offerings, identical products, identical services.

For decades high-growth companies have been predominantly technology-based, and increasingly all technology has become software-based; or perhaps, increasingly software has become an essential part of every new technology. Think of it as coupling your business to Moore's Law: microprocessor power doubles while price halves every 18 to 24 months.

> 'Firms that use more IT tend to have higher levels of productivity and faster productivity growth than their industry competitors.' Brynjolfsson & McAfee[1]

Last year – 2016 – I met a programmer in Sweden who works for a steel company. You don't get much more old-school than a steel manufacturer, and 30 years ago the only software such a company might use would be for the payroll. But today software is essential for scheduling production, routing molten steel around the works and improving performance.

Look at the way software has upended the world of media. TV programmes are broadcast digitally to powerful computers: today's TV sets. But then, who watches broadcast television? iPlayer, Netflix and Amazon mean we can watch what we want, when we want, thanks to software. Or consider newspapers: software running on tablets, phones and desktops has consigned profitable print-only newspaper companies to history.

Once a product becomes digital, everything changes. Ask the music industry. First expensive hardware allowed music to be digitized, then software reduced the price, then software created new ways of purchasing digital music. At first the industry resisted, but that didn't

[1] The Second Machine Age, Brynjolfsson & McAfee, 2014

stop the likes of Napster. It was only when Apple arrived with the iStore and iTunes that the industry started to see how it could survive.

As old industries such as newspaper publishing change and even die, the same technologies allow new ones to come into being. Look at print-on-demand services such as Lulu.com, which allow anyone to publish their own book without capital investment, or LeanPub.com, which is a totally electronic publisher.

Or take cars. A well-established industry is in the process of being disrupted before our very eyes. Who is making the most interesting cars today? Maybe it is Tesla, a company founded on software. A Tesla needs hardware, but could hardware substitute for software? It's not just the maps and the music: the drive train requires advanced software to optimize battery performance.

The engines of most modern cars require significant software to work. I read somewhere that 25% of the value of a new car is software. I can't find that quote any more, but I found sources saying that traditionally 10% of car's value was electronics but today it is nearer 50%. When I've mentioned these figures to people who know they say they are on the low side.

The fact that Apple and Google are developing cars demonstrates that something is changing. Being able to build a physical car is no longer enough – in fact you can outsource that bit to Magna. Ford, Fiat, Toyota and other car manufacturers have been turning out very similar cars – similar to their previous cars, and even similar to each others' – for years. After the early 1980s 'quality revolution' little changed in the world of cars: successive Golfs, Camrys and Mondeos were hard to tell apart.

Then software arrived… Cars now have maps and navigation systems. Cars can park themselves, and soon they will drive themselves. For a car to be worth buying it must contain software technology: without it a car is a utility item that cannot demand a high price.

Now that the cars of tomorrow will compete on software capabilities it is logical for companies that have an advantage in the creation of software – Apple, Google and others – to enter the market.

Software is indeed eating the world.

Note the consequence of business dependence on software:

> Software *is* the business

When delivering the fundamental product or service of the company becomes highly dependent on software, the software is no longer optional: software can no longer be done by 'IT' in a back room – the business and the software technology are one.

Increasingly this gets called *Digital*.

Digital includes not just online sales and electronic delivery. Digital business includes 'big data' and analytics, the Cloud, phone apps – indeed cellphones and smartphones altogether, GPS positioning[2], artificial intelligence and the things that they make possible: drones, the gig economy and more.

Digital hides the underlying truth that is software. All these technologies are only possible with software. Digital business is dependent on software, which means that the business is dependent on those who create software.

Digital implies software development, but while 'software development' is a bit of a mouthful, 'digital' is short and sweet. This is not to say that the digital business is only a software development business; there is a lot more to digital as well, but the need to have access to, or to be able to create, software is fundamental to the business.

Software technology is now as important as marketing or accounting.

Unfortunately in 2016 it is still acceptable for even senior executives to say "I don't really understand software" and "Programmers are expensive geeks who I don't understand." Could you imagine these same executives saying:

> "I don't really understand accounting – whether to expense or capitalize…"
>
> "Why do those accounting nerds demand double-entry? Surely if we just did it once we could save money?"
>
> "I don't really understand marketing – I only know that half the money is wasted."
>
> "I just want marketing that works."

It is not so much that every company is a software development business, as that every company is software-dependent business. The more a business becomes digital, the more it takes on the characteristics, attributes and business models of what we used to call 'software houses'.

So let's talk about projects…

[2]In *Pinpoint*, 2016, Greg Milner suggests that the significance of global positioning systems has been overshadowed by the rise of the parallel Internet, and that the impact of positioning systems is potentially bigger than the Internet.

The project model of working dominates software development. (Let's leave the different definitions of 'project' until later.)

The project model is inherently short-sighted. Call it *Project Myopia*: projects end.

Do you want your business to end?

> "When will this business be done?"

> "When will your company finish?"

Questions like these are only asked about businesses that are in trouble. We don't want businesses to end. We want businesses to continue: to continue employing people, to continue supplying their products to grateful customers, to continue paying dividends to shareholders.

If the company *is* software and we want the company to continue, why do we think the software will ever be 'done'? Again, imagine the head of Microsoft Office development meeting with the Microsoft CEO, Satya Nadella, and saying "Great news, we finally finished Office after 26 years. We can lay off the team and get started on the next thing." Microsoft's profits would collapse.

Software businesses such as Microsoft are the prototype of future businesses. Software businesses have been exposed to the threats and opportunities created by software for longer. Those who work in these companies have long had access to early versions of the tools that are now becoming commonplace. This doesn't mean that these companies have always got it right – only that others can learn from such businesses.

The idea that software is never 'done' may terrify some readers – how does one know how much to spend? Some will rail against this idea – after all, we can control 'scope', we can push back on requests, and we don't need to upgrade with every new technology or OS change, and we…

Personally I don't think scope can be controlled. However, if you have an example where it can, then OK, I concede. But…

Changes are opportunities. Rather than viewing requests to change software as a bad thing that should be resisted in the name of 'done-ness' and cost control, we should see change as a positive that provides our businesses with growth opportunities.

How many times has Microsoft sold you Word?

How many times has Oracle sold your company their eponymous database?

Software companies have long succeeded in using the cycle of software upgrades to extract money from customers. It's true that the business models of software companies are changing too. In a world in which all companies start to look like software businesses, it becomes harder and harder to find any which are purely software businesses.

It is not just in extracting more payment from customers that software can deliver value to business. Take my Swedish friend: continuing to improve their software allows the steel mill to continually extract more cost saving from their plant.

In many companies – perhaps most – there is no single long-lasting 'optimal' point of production, or optimal processes. Many businesses never approach the optimal point because they don't know what it is. More likely they operate at some local optimum, which may be fine for a while, but when faced with a new competitor, a new technology or even just a change of personnel they are forced to find a new and hopefully better local optimum.

When a business is based on software, moving to a new optimum requires changing the software – or rather, changing the software allows a move to a new optimum.

Since it is not only a single business that is software-based, but all the other businesses with which it works, there are continuing opportunities to optimize one business by responding to or even initiating changes in others.

Take parcel delivery, for example. This was once a physical business: collect parcels, move them, deliver them. Then FedEx, UPS and others arrived, they allowed you to track a parcel online. But technology goes deeper: such companies could not run their businesses the way they do *without* software. This change has sparked changes in other businesses, not just competitors such as Royal Mail or US Mail, but in businesses that can now send parcels reliably across the globe in hours.

Is UPS' parcel software ever done? For such companies software is not a project to be completed. Software is their lifeblood. Finish the software and you finish the business.

I can already hear readers who work for delivery companies shouting "But we have software projects! And they work!"

I'm not disputing that many companies organize their software development using the project model, although I admit that I don't know the internals of FexEx or UPS. My point is that running such a model hinders such businesses, and that breaking free of the project model offers opportunities for companies to improve. The project model complicates the management of software development and induces myopia.

The problem is that the project model is ill-suited to software development. When software was a sideshow – an optional extra – companies could get away with patching a broken

model. Now that software *is* the business, using an ill-fitting model adds complexity, hinders improvement, and in some cases may well break the company.

Because the project model promises 'an end' – and because people like closure – executives and managers suffer from myopia. They fail to comprehend the implications of the fight against a never-ending process, and so pass up opportunities for growth and market leadership.

Software technology has allowed businesses – whether they be delivery companies, steel mills, banks, airlines or whatever – to grow ('scale' in the jargon) beyond the wildest dreams of company managers 100 years ago. But if they are to continue to exist, let alone grow, over the next 100 years, then they need to think differently.

One of the ways they need to think differently is to accept software development – 'digital' – into the core of the business as an ongoing activity, rather than something that happens over to the side and that will be done 'one day'.

2. Software as an Asset (SaaA)

> *Total factor productivity growth increased more between the 1990s and 2000s in IT-using industries, while it fell slightly in those sectors of the economy that did not use IT extensively.* Brynjolfsson & McAfee[1]

[1] The Second Machine Age, Brynjolfsson & McAfee, 2014

3. Higher purpose

> *I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the earth.* President John F Kennedy
>
> *…a computer on every desk, running Microsoft software.* Bill Gates
>
> *Beat Xerox.* Canon Group
>
> *…to organize the world's information and make it universally accessible and useful.* Google

Goal, mission, vision, objective, BHAG (*Big Hairy Audacious Goal*)[1], MTP (*Massively Transformational Product*)[2] – call it what you will, but the underlying idea is the same: *a higher purpose.*

Organizations and people are motivated and inspired when they have a higher purpose. Organizations and people work better together when they share a common higher purpose.

In the Continuous Digital model each team is a miniature business in its own right, so each team needs a sense of higher purpose. In addition to the overarching organizational purpose, there needs to be a team-level shared higher purpose.

Survival is not enough: organizations are akin to organic life – they want to survive; they want to live; they want to grow. But survival alone, while necessary, is not enough. Organizations and the people in them need a higher purpose to motivate and focus on.

Over the years various authors have suggested different names and forms for this higher purpose; some are given above. While the details differ, the essential element is the same: something bigger than the individual, something that gives meaning, something to aim for, something against which to measure progress.

[1] *Built to Last: Successful Habits of Visionary Companies*, Collins and Porras, 1994
[2] *Exponential Organizations*, Ismail, 2014

Not money

> *Human beings don't only want comfort, safety, short working hours, hygiene, birth control and, in general, common sense; they also, at least intermittently, want struggle and self-sacrifice, not to mention drums, flags and loyalty parades.*
> George Orwell

Few people are motivated by money in anything other than the short term. Making money – creating revenue, making profits, enhancing return on investment – are objectives best pursued indirectly. Experience has shown that when organizations only focus on making money, things go wrong: think of Enron, Lehman Brothers and countless less famous examples.

The economist John Kay goes further in suggesting that directly targeting profit impedes the propensity to do good and generate profit. Kay cites numerous examples, such as the British chemical company ICI, which successfully generated profits for decades while it aimed at:

> '…serving customers internationally through the innovative and responsible application of chemistry and related science.'

In 1997 ICI restated its higher purpose as:

> '…to be the industry leader in creating value for customers and shareholders through market leadership, technological edge and a world competitive cost base.'

Ten years later ICI was gone[3].

Even the one-time champion of 'shareholder value', Jack Welch, has described single-minded pursuit of profit as "The dumbest idea in the world."

Making money is necessary for businesses and individuals, but more importantly, money is information. Healthy cash-flow shows a business that it is doing well, and failure to make enough revenue to cover costs tells a business that it needs to change or pack up.

Modern commerce has developed a mass of 'financial engineering' techniques that allow money to be extracted from an organization. Unfortunately these techniques dilute the informational value of money for the underlying business. That a company can pay large dividends while the core business rots demonstrates the prowess of financial advisors and obscures the underlying issues.

[3] *Obliquity: Why our goals are best achieved indirectly.* John Kay, 2017

Obliquity

> *Obliquity describes the process of achieving complex objectives indirectly... oblique approaches recognize that complex objectives tend to be imprecisely defined and contain many elements that are not necessary or obviously compatible with each other, and that* **we learn about the nature of the objectives and the means of achieving them during a process of experiment and discovery**[4].
> John Kay, 2017

Kay's concept of oblique goals is not only attractive for technologists, but describes in one word the whole *raison d'être* for software development. Developers start with a necessarily fuzzy goal because the goal exists in a fuzzy world, but our machines do not tolerate fuzziness; they know only the finite states of one and zero. If the problem is already sufficiently precisely defined, then the computer can already solve the problem.

> 'The computer is very good at solving the problem we have specified and asked it to solve, but less useful when we are not quite sure what the problem is.' Obliquity, John Kay, 2017

Almost by definition the activity of instructing the computer to solve a problem is oblique, fuzzy, ill-defined and requires one to learn about the solution and the problem.

Software development is inherently the work of turning oblique and fuzzy objectives into specific systems. In doing so engineers deploy an armory of tools that seek to clarify the problem in hand and focus attention and effort, while engineers offer up potential solutions that may do more to help understand the problem than to provide a solution.

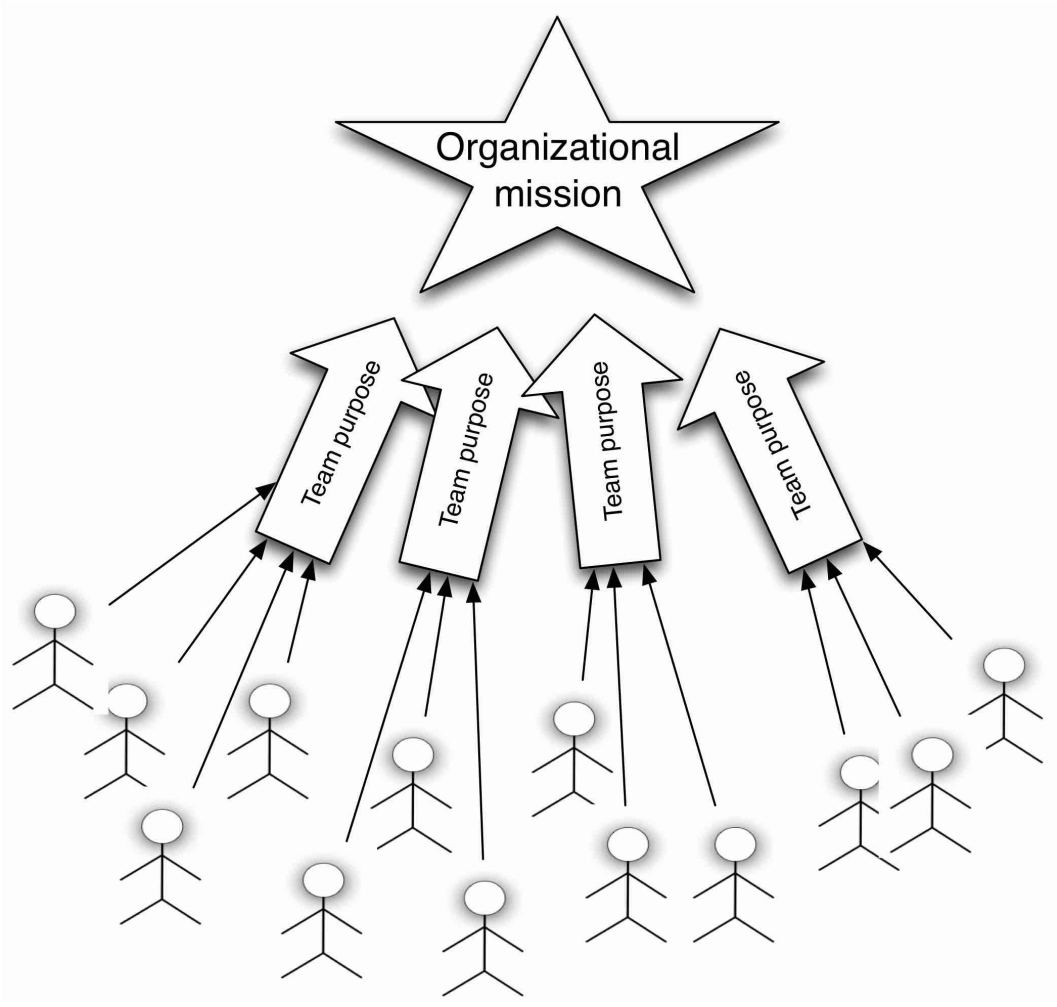
Embracing Kay's call for obliquity opens this author to the charge of hypocrisy, because I have written much on defining, specifying and quantifying work. But in truth software development always begins with oblique problems. Iterative software development is the act of repeatedly focusing on the problem and potential solutions in order to create better understanding. It is because problems software engineers address are inherently fuzzy and oblique that one must constantly attempt to clarify the problem and solution. It is through repeated and iterative attempts to define and specify the problem and solution that one advances towards both a solution and the problem.

When technologists are able to specify problems and goals in specific terms, then solutions can be crafted. Given a sufficiently precise problem statement, crafting a solution can be trivial. However, specifying the problem is far from trivial.

[4]Emphasis added.

Organizations and teams

That organizations need a higher purpose is well-established, but what is less well-established is that teams need their own higher purpose.



Individual aims build towards teams' higher purposes, which build towards the organization's higher purpose

While grand organizational – even national – goals may be highly motivating, they can also be very remote. While an oxygen tank designer deep in the bowels of Grumman Aerospace might understand how their work builds towards putting a man on the moon, does an

accountant deep in the bowels of Google understand how their analysis of expenses helps to organize the world's information? Teams need to have their own higher purpose, and they need to understand how that builds towards the ultimate higher pursue of the organization.

Given that authority is devolved to the team, and that the team is expected to both decide what to do and execute it, then it is not possible to govern a team on the basis of "Have they delivered X?", because the team decides what X is. Assessing effectiveness and progress is tricky.

Knowing both team and organizational higher purpose allows one to frame questions of effectiveness and progress in terms of the higher purpose, for example "Has this release contributed to the team's purpose? To the organization's purpose?"

Scale

Each team needs its own higher purpose.

This is especially true when a team needs to work with other teams in the organization. Since all teams ultimately share the organizational higher purpose, team-level aims should not be too dissimilar.

When an organization is small there may only be one team. The organization's higher purpose is the team's higher purpose. Life is easy. But with growth there will come a point at which teams need their individual sense of purpose. In the first instance it would seem rational to hand these teams a predetermined higher purpose. Indeed this might be necessary in order to establish a team in the first place.

Where a team is following the 'amoeba model' and splits into two teams, it is entirely possible that the two teams share a common purpose at least initially. Over time their higher purposes may diverge.

Mutable

Although a team may be given – or inherit – a higher purpose to start with, teams need ownership of that higher purpose. Therefore each team needs to have the authority to modify their higher purpose over time. As they are part of a larger organization, such changes need to be undertaken in harmony with the organization.

A team intending to modify its stated higher purpose would consult with others – its customers and other teams with which it works. The organization's governance board is

probably the ultimate arbiter of whether a proposed change of purpose can be made, although a team should probably avoid getting into a situation in which the board says "No".

If team members start to doubt their higher purpose and are unable to question and ultimately change their purpose, then they may be compelled to leave the team.

This might even be desirable – if one individual finds they are no longer motivated by the higher purpose, it may well be a sign that they should move. This is perfectly natural – individuals grow and change throughout their lives, and what they want and what motivates them changes over time.

However, if multiple team members find the higher purpose no longer motivates them or seems irrational, it may well be sign that the purpose needs revising. Again it is natural that over time teams' higher purposes will mutate and change – after all, the world around them is changing, technology is advancing, and sticking doggedly to a purpose that has become outdated is detrimental.

If team members are to truly believe in their higher purpose, they need to 'own' the purpose; if the team collectively owns the purpose, then it is allowed to change it. One of the privileges conferred by ownership is the right to change property. If 'owners' cannot change their purpose then they are not true owners.

Individuals

Each member of the organization should have a sense of mission. Inamori Kazuo[5]

Only when team members understand and share the team's purpose does it make sense to talk about individual goals or objectives. Indeed one might even ask: *if the team is the basic work unit, and if one is striving to create effective teams, then surely the team goals are individual goals?*

I'll leave that as an open question for discussion. However, if you decide that individual goals still have a role to play, it follows that individual objectives should not only be aligned with team and wider organizational goals, but should actually be *derived from* team and organizational goals.

[5] *Amoeba Management*: Kazuo Inamori, 1999

Project completeness

In the project model completion of the project is the goal. However, such an objective leads to goal displacement and damages profitability in a digital world. Still, individuals and organizations need meaningful and achievable goals.

Quantifying goals can be a great aid to determining whether they are met. However, sometimes it might be better to avoid quantification: set an oblique goal, make the goal something to strive for, and thus avoid attempts to 'game' the target. The problem with quantifiable goals is that while they appear objective, the goals can be met by oblique means, which undermine the goals' validity. This is highlighted by *Goodhart's Law*[6].

Closure

One of the advantages of the project model is that it allows individuals and organizations to achieve closure. Humans like closure: it provides a satisfying psychological safety. Something is done, complete, finished. That the continuous model deliberately avoids closure is itself something of a problem psychologically. However in the commercial world closure often creates problems.

> 'This shop is closed.'

Shops that close on Sundays can be an inconvenience – one cannot satisfy one's immediate needs.

> 'This shop has now closed.'

Shops that close and cease trading can be an even bigger disappointment.

Business closure is normally a bad sign – businesses want to live and continue!

Unfortunately organizations sometimes meet their goals, their BHAGS (big hairy audacious goals), their objectives, or complete their missions. Upon achieving their goals, both Canon and Microsoft faced existential questions and suffered commercial turbulence for several years. NASA too struggled in the years after Neil Armstrong became Kennedy's 'man on the Moon', and the final three Apollo missions were cancelled.

[6]https://en.wikipedia.org/wiki/Goodhart%27s_law

While one might sidestep such a problem by defining the higher purpose as something that is never complete, such an approach may be self-defeating if individuals feel that the aim is never achievable.

This is definitely one of those ‘nice to have’ problems. However it is one that should not be ignored indefinitely. The solution requires flexibility in the goal: once the goal is achieved, the organization needs to be capable of mutating it or adopting a new goal. The same should also be true of teams that achieve their higher purpose.

Finally

At an individual level a higher purpose helps to enroll and motivate team members. At an organizational level the teams’ higher purposes allow the organization to coordinate disparate teams and assess effectiveness.

The organization’s ultimate goals and objectives serve to coordinate teams and add to individuals’ own sense of purpose.

Oblique goals provide the team with latitude to innovate and allow problems to be reformulated as solutions emerge. Oblique goals better match the nature of technology work: creating the specific from the vague and interfacing the exact world of machines to the fuzzy world of people.

4. Team-centric development

Divide the organization as necessary into small units, and rebuild it as a unified body of small enterprises. Entrust the management of these units to amoeba leaders to cultivate personnel with managerial awareness. Kuzuno Inamori[1]

[1]Amoeba Management: The Dynamic Management System for Rapid Market Response. Kuzuno Inamori, 1999

5. Work to be done

Over and over, people try to design systems that make tomorrow's work easy. But when tomorrow comes it turns out they didn't quite understand tomorrow's work, and they actually made it harder. Ward Cunningham

6. Value

Price is what you pay. Value is what you get. Warren Buffet

Value is perceived benefit: that is, the benefit we think we can get from something. Tom Gilb, Competitive Engineering Concept 269[1]

[1] *Competitive Engineering*, Tomb Gilb, 2005

7. Summarizing the alternative model

Perhaps the most important complementary innovations are the business process changes and organizational co-inventions that new technologies make possible.
Brynjolfsson & McAfee[1]

[1]The Second Machine Age, Brynjolfsson & McAfee, 2014

II Interlude

> *Kurt Lewin argued that 'nothing is as practical as a good theory'. The obverse is also true: nothing is as dangerous as a bad theory.* Sumantra Ghoshal[2]

Digital business demands new mental models and new reflexes. Management models and reflexes that originated during the industrial era are poor guides in the digital age.

Less sooner is worth more than *more later*. Technology is not something that happens off to the side: it is an integral part of business. Ceasing technology enhancements is nonsensical; requests and opportunities to enhance technology need to be welcomed just as new customer opportunities are welcomed.

Simply exploiting economies of scale will no longer work. Much digital work exhibits diseconomies of scale.

Higher quality is cheaper and faster than low quality.

Before exploring the Continuous Digital model in more detail, some foundations need to laid.

[2] *Bad Management Theories are Destroying Good Management Practices*, Sumantra Ghoshal, Academy of Management Learning & Education, 2005, vol.4 no.1, 75–91

8. Diseconomies of scale

Whenever a theory appears to you as the only possible one, take this as a sign that you have neither understood the theory nor the problem which it was intended to solve. Karl Popper, Philosopher, 1902–1994

9. Diseconomies and risk

Only those who will risk going too far can possibly find out how far one can go.
T. S. Elliot, poet, 1888-1965

10. Living with diseconomies

It is difficult to get a man to understand something, when his salary depends upon his not understanding it! Upton Sinclair, American writer and politician, 1878-1968

11. Schedules

85% of companies do not quantify cost of delay. Reinertsen, 2009[1]

Six months' delay can be worth 33% of lifecycle profits. McKinsey/Reinertsen, *Electronic Business*, 1983

[1] *Principles of Product Development*, Reinertsen D., 2009

12. Time-value profiles and elastic deadlines

Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less. Marie Curie

13. I need it yesterday!

Yesterday all my troubles seemed so far away,

Now it looks as though they're here to stay.

Oh, I believe in yesterday.

Lennon and McCartney.

14. Theory X, Theory Y and strategy

There is a great deal of talk about loyalty from the bottom to the top. Loyalty from the top down is even more necessary and much less prevalent. General George S. Patton

15. Planning

Failure to plan is planning to fail. Project manager maxim, source unknown

Our conclusion is that planning is a centralizing process, discouraging the very commitment it claims so earnestly to require. Mintzberg[1]

So the real purpose of effective planning is not to make plans but to change the microcosm, the mental model that these decision makers carry in their heads. de Geus[2]

---

[1] *The Rise and Fall of Strategic Planning*, Henry Mintzberg, 1994
[2] *Planning as Learning*, Arie de Geus, Harvard Business Review, 1988, 66, 70

16. Piecemeal growth

> *Gall's Law: A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system.* John Gall, 1975[1]

[1] *How Systems Really Work and How They Fail*, John Gall, 1975

III Teams

Everyone thinks of changing the world, but no one thinks of changing himself.
Leo Tolstoy

17. Devolved decision-making

Plans are only good intentions unless they immediately degenerate into hard work.
Peter Drucker

18. Team strategy

Never tell people how to do things. Tell them what to do, and they will surprise you with their ingenuity. General George S. Patton

19. Stable teams

You can't have great software without a great team, and most software teams behave like dysfunctional families. Jim McCarthy, *Dynamics of Software Development*

20. Team lifecycle

No organization which purposefully and systematically abandons the unproductive and obsolete ever wants for opportunities. Peter Drucker, *Age of Discontinuity*, 1992

By definition, innovative ideas look like bad ideas, and if they didn't, everyone would have them. Bill Gross, Investment Fund Manager

21. Team lifecycle – another example

We began with this idea of Bob's [Robert Rauschenberg] that you work with what's available, and that way the restrictions aren't limitations, they're just what you happen to be working with. Steve Paxton, dancer

IV Money

By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race. Alfred North Whitehead, Mathematician and philosopher

22. Real options and venture capital

If you want to have good ideas you must have many ideas. Most of them will be wrong, and what you have to learn is which ones to throw away. Linus Pauling

23. Continuous governance

All organizations need a discipline that makes them face up to reality. They need to recognize that the probability that any activity or program will fail is always greater than the probability that it will turn out successful, let alone that it will accomplish what it was designed to do. Peter Drucker, *Age of Discontinuity*, 1968

24. Budgeting

A target is what we want to happen. A forecast is what we think will happen. Never combine the two in one number, like a budget does. Bjarte Bogsnes[1]

Cost budgets tend to be spent, even when the initial budget assumptions change (which they almost always do). Managers do not behave like this to cheat; they do it because the system encourages them to do so. Managers see budgets as entitlements, as bags of money handed out at the beginning of the year. Nobody gets fired for spending their budget. Spending too much is, of course, bad, but spending too little is not good either. Bjarte Bogsnes[2]

[1] Bjarte Bogsnes, on Twitter@bbogsnes, Feb 2013
[2] *Implementing Beyond Budgeting*, Bjarte Bogsnes, 2009

25. Assets and accounting

Do not be alarmed by simplification, complexity is often a device for claiming sophistication, or for evading simple truths. J. K. Galbraith

Software is an asset to the business that owns it but it cannot be a static asset. This book

26. Money trouble

The twin policies of managing for profit and maximizing shareholder value, at the expense of all other goals, are vestigial management traditions. They no longer reflect the imperatives of the world we live in today. They are suboptimal, even destructive – not just to the rest of society, but to the companies which adopt them.
Arie de Geus[1]

[1] *The Living Company*, Arie de Geus, 1997

Like to read more?

I hope you enjoyed this sample. If you would like to read more…

Buy Continuous Digital today on Amazon[2]

Amazon logo

[2]https://amzn.to/2CUbMbW

About the author

Allan inspires digital teams to effectively deliver better products through Agile technologies. These approaches shorten lead times, improve predictability, increase value, improve quality and reduce risk. He believes that improving development requires broad view of interconnected activities. Most of his work is with innovative teams, smaller companies - including scale-ups; he specialises in product development and engineering. He uses a mix of experiential training and ongoing consulting. When he is not with clients he writes far too much.

He is the originator of Retrospective Dialogue Sheets[3], the author of several books including: "Xanpan - team centric Agile Software Development" and "Business Patterns for Software Developers", and a regular conference speaker.

Contact: allan@allankelly.net

Twitter: @allankellynet[4]

Web: http://www.allankelly.net/[5] and http://www.allankellyassociates.co.uk/[6]

Blog: https://www.allankellyassociates.co.uk/blog/[7]

Also by Allan Kelly

Project Myopia: Why projects damage software #NoProjects

Little Book of Requirements and User Stories

Available from your local Amazon[8]

[3] http://www.dialoguesheets.com/
[4] https://twitter.com/allankellynet
[5] http://www.allankelly.net/
[6] http://www.allankellyassociates.co.uk/
[7] https://www.allankellyassociates.co.uk/blog/
[8] https://www.amazon.com/Little-Book-about-Requirements-Stories-ebook/dp/B06XZZ6BQD

Xanpan: Team Centric Agile Software Development

Ebook: https://leanpub.com/xanpan[9]

Print on demand: Lulu.com[10]

And your local Amazon[11]

Business Patterns for Software Developers

Published by John Wiley & Sons

Available in all good bookshops and at Amazon[12]

Changing Software Development: Learning to Be Agile

Published by John Wiley & Sons

Available in all good bookshops and at Amazon[13]

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

[10] http://www.lulu.com/shop/allan-kelly/xanpan-team-centric-agile-software-development/paperback/product-22271338.html

[11] https://www.amazon.com/s/ref=nb_sb_noss?url=search-alias%3Daps&field-keywords=Xanpan

[12] https://www.amazon.com/Business-Patterns-Software-Developers-Allan-ebook/dp/B007U2ZT7K

[13] https://www.amazon.com/Changing-Software-Development-Learning-Become/dp/047051504X