HOW TO BRING CURIOSITY BACK TO BUSINESS

And how software can help

Karl Jeffery and Dimitris Lyras

CONTENTS

Published by Software for Domain Experts Ltd, 39-41 North Road, London, N7 9DP, UK. Www.softwarefordomainexperts.com. September 2018

Karl Jeffery is editor of Digital Energy Journal, a magazine about digital technology in upstream oil and gas, and managing director of Future Energy Publishing, with publications and events in upstream oil and gas, tanker shipping and carbon capture

Dimitris Lyras is director and founder of Ulysses Systems, Ulysses Learning, and a director of Lyras Shipping

WE CAN DO BETTER THAN JUST MAKE BUSINESS DECISIONS BASED ON THE NUMBERS

The idea of making business decisions based purely on numbers – such as past financial performance – is an approach which has largely worked well in the past.

But it is not the only way to do it.

There is so much knowledge that the numbers cannot convey. Numbers only describe certain elements of past performance, and so much in business which is not known until it happens. And numbers only provide an incomplete picture of past performance, a compilation of various pieces of status data, put together in a certain way.

The numbers are important – particularly the number about financial margins, because without a financial margin, a business cannot continue.

But numbers can never convey what people with the most detailed knowledge of the domain know, such as its markets and trends, and the business' capabilities.

So these people can be in a better position to make judgement than anyone who is only using the numbers as a basis for decision making.

We see the results in society everywhere of too much decisions by numbers rather than decisions by expertise or judgement. The same shops and restaurants on every high street (which got investment because of one shop which worked somewhere once). The high sugar levels in some food (because over the short term, high sugar food is easiest to sell).

The way all of our organisations, including non business ones such as schools and hospitals, turn into machines to try to reach targets, rather than machines which nurture excellent decision making and understanding of the domain (that's a fancy description for a good teacher or doctor, or a manager of these people).

Further, the amount of "decision making by numbers" in our society has got so high, with so many people doing it, it seems hard to imagine anyone could achieve competitive advantage from making better decisions-by-numbers than the next person. And meanwhile who knows how many good business decisions – or ideas – do not get implemented because there are no numbers to support them.

Banks, in particular, are big proponents of 'decision making by numbers' culture. That's partly because numbers are usually all banks have to make decisions on, and banks have a powerful role in our society which relies so much on debt.

But that does not mean everybody has to copy the way bankers do it. It does not mean investors need to close their minds to alternative decision making approaches.

And now we see the growth of algorithmic trading, or investment decisions made by a machine. The machine cannot see anything the people who understand the business can see. All it can see are past performance of stock price, or other past financial data. So it further embeds the idea that past performance is all that matters.

And we see a software industry designed to profit in this "decision making by numbers" investment culture. The software for our workplaces could be designed to support people's decision making and understanding of their domains. But it isn't. That's a waste of technology's amazing capabilities in 2018.

This book shows what might happen if we put things the other way around – develop software which is better able to nurture people's curiosity, support people's development of understanding, leading to improved decision making.

This would lead to the business operating better, being a more interesting and pleasant place to work, new business opportunities, and ultimately a more sustainable organisation with better performance over the longer term.

We don't have the numbers to prove it works.

BRINGING CURIOSITY BACK TO BUSINESS

Since the start of humanity, getting by in the world, and in business, depended on curiosity.

People became who they were and what they could do through listening, learning and understanding, and developing expertise. Their curiosity drove them on.

History shows what curious people did – to see if they could travel somewhere, create something, conquer something, survive in a new place, form a new relationship. Through developing understanding, we learned to work with what nature gave us. We learned how to organise and motivate others.

Your ancestors travelled on foot from East Africa, where humans began, to the Southern tip of South America, going via the land bridge from Russia to Alaska – all driven by curiosity and the need to survive, finding out new ways to live, staying somewhere for a while and then moving on.

We were evaluating options, understanding situations, working out the right pathway forward.

Cold lands were good because meat, once killed, would stay frozen for a long time. Hot lands were good for fruit and growing plants. Your ancestors would have had challenges working out where the food would come from after the animals in one part of the world had been killed, how to handle a warring tribe next door, keep children happy, or find a partner.

Today's business environment is very different. Funds are allocated by investors, and managers controlled by investors, based on what has worked in the past.

This is not a business environment designed to encourage curiosity, and which often represses it. It results in companies becoming more standardised, and less nurturing of curiosity in both our schools and organisations. It drives a race to develop the largest, most standardised companies.

So we end up with many businesses which look alike – airlines, hotels, telecom companies, supermarkets. They compete with each other on small details. Or build their competitive advantage solely on factors which were in place before the rush to satisfy algorithmic investors started – such as a long serving culture or brand, or assets not available to any other company.

Company managers obsess about keeping up with or 'benchmarking' their competitors on a range of different factors, rather than working out where they could be going, or making sure they are supporting their curious staff to experiment.

A successful company or organisation, of course, does require some degree of organisation and standardisation, running alongside the nurturing of curiosity. This book is not an argument of one over the other. It aims to show that we may be getting the balance wrong, and a pathway for changing the status quo.

Bear in mind that curiosity itself is a delicate creature. It cannot arise through purely linear thinking, because there is rarely a direct path from curiosity to business success. Curiosity needs its own intrinsic motivation. Like a teenager, it needs to be in an environment where it feels supported. None of this sounds appealing to investors.

The change we need may be for more people to understand the importance of curiosity to business, and what kind of environment – including software – can nurture it.

An environment which stimulates curiosity

Curiosity is not a "have it or you don't" kind of thing. Children probably have more curiosity than adults, and some adults have it more than others. But it can also be stimulated or killed by our environment.

Curiosity might be stimulated by a sense that there is somewhere we can get to, with a better understanding, or a problem to solve, and it is worth putting the effort in to figure out how to get there. Curiosity might be simulated by having the mental space and support to figure things out. Curiosity might be killed by a stronger force which occupies our mind space, such as 100 e-mails demanding a reply, or a screaming child.

Curiosity can be stimulated by others, when we talk about what we can see or what we have learned, or tell stories which help someone else understand their situation, or encourage each other.

Curiosity can be stimulated when we better understand where we are, and see gradual progress.

It helps when we understand and see the results of what we are learning and doing, or see some progress happening.

The raw material for expertise

Curiosity is the raw material which enables us to develop expertise – where we can do something useful in the real world. People use expertise to manage hospitals, produce oil, run trains, win football matches, keep good relationships with other counties, keep our supermarkets stacked at ow prices, keep people out of poverty, keep water supplied through summer months, minimise environmental damage and all of the other wonderful things people do, which don't get measured, but which keep our societies safe.

Someone with curiosity based expertise can add real value in business. They can spot problems emerging, see things which others don't, spot pathways forward, keep services being provided reliably, avoid the high costs of an unexpected problem.

We use the word "expertise" more often than "curiosity" to describe someone who knows how to do something. But this book focusses on curiosity because expertise can be something of a loaded word in today's society, with hints of elitism, access to advanced education, the idea of pretending you know what you are talking about.

Curiosity is available to everyone, not just those with access to formal education. And learning from education is very different to learning from your own curiosity.

Education will mainly teach you what other people have learned before you, rather than providing an environment for you to figure things out by yourself. It means you are not learning anything new. This book is about developing new knowledge, not understanding what other people can already understand.

An example of this is when we hear senior managers at companies say they consider a university degree just the raw material for someone gaining the ability to provide value to their firm.

Turning wasteland into a beautiful garden

Curiosity can take a small piece of wasteland and turn it into a beautiful garden. Curiosity can build a lively café in a place where there was previously just tables and chairs. Curiosity can turn a school into a place of enormous life and energy, where previously children just listened to a teacher.

Since humans existed, they have created beautiful worlds for themselves on land which initially appeared to offer nothing at all.

Without curiosity, this cannot happen. The starting materials are all you can see. A piece of wasteland is a piece of wasteland. Some junk data is junk data. A non performing child is a reject.

Rigidity kills curiosity

And the biggest killer of curiosity might be rigidity in your organisation, to be told there's a single way to do something, or that your role in the organisation is to follow the system, including its software.

Yes, it is necessary for companies to have certain rigid elements about how they do things, or require a basic knowledge of their employees. But that can just be the starting point for developing curiosity, not closing down curiosity altogether.

Curiosity can easily be killed – and society might be doing just that. A world without any curiosity is a sad place.

Curiosity and creativity

Sometimes organisations nurture what they believe to be creativity. They encourage people to develop new ideas, to think about things differently.

But there is an important differentiator here. Curiosity is about finding new pathways, not creating something. Curiosity is stimulated by a sense that there is something to go to. Creativity may mean this, but not necessarily.

Organisations might shudder at the idea of encouraging 'creativity' among staff, thinking it leads to people wanting to develop their own solutions to problems or go their own way. Creativity can push in an opposite direction to the organisation itself.

Or some organisations say they encourage 'creativity' but everyone understands it is creativity within a very narrow scope, such as pinning interesting pictures around your cubicle, or planning an unusual office day out. They are nurturing tiny levels of curiosity within a very rigid business.

Curiosity is different. It is a desire to see what might be possible for the organisation and our role in it – stimulated by a sense of progress. That is something a rigid organisation cannot do.

Linear thinking hijacks curiosity

Our minds are easily hijacked by a quest to reach linear destinations. Somehow we have evolved so that the linear thinking easily pushes out the curiosity based thinking from our minds.

Consider the most popular sports – running, boxing, football, tennis, cricket, even fishing. All involve a quest to a linear destination – hit the ball back, kick it up the field, get lots of runs, run faster, win the fight, catch the most fish.

There are non-linear elements of all these sports – but the core goal is pretty linear. Audiences readily understand the quest to a linear destination of the people involved, which is why these sports gather big audiences or big participants.

And our software – or the way we work with it – can seem designed to encourage linear thinking. Certainly social media, where it pushes us to learn new ways to get ‘likes’ or ‘friends’. And also e-mail, where each of us face every working day 100 messages designed to demand our attention. All of this brings out our linear thinking and kills curiosity.

When we use software which is badly built, or which is closely designed to how computers ‘think’, that also forces more linear thinking. Because a computer, of course, can only think linearly. There are people who can think abstractly and with curiosity when working with a linear computer – such as expert hackers and programmers – but these people can be pretty rare.

Computer games, and ‘gamified’ workplace applications, with scores in the top right of the screen, also push our brains into linear thinking.

Building software which encourages linear thinking is like making food loaded with sugar. People will be drawn to it, but it won’t help them in the longer term.

Software can be designed to simply be a tool to help us – like Notepad, or the basic functionality of Microsoft Office. No scores, alerts, or forcing us to think like a computer. But it could also be written specifically for the needs of our domain. We’ll discuss how this could be done later.

Some things curiosity likes

Curiosity likes to discover similarities and patterns, so it can see and better understand what is going on. It sees things happening more than once, and wants to understand why. For example, a doctor sees a patient with a similar symptoms to a patient from a few months ago, and remembers how the treatment turned out.

Curiosity likes a thriving social environment, where people are perhaps lightly challenging each other or helping each other develop their understanding.

Curiosity sometimes likes a personal teacher, someone who has a deeper understanding of a situation based on past experiences with it and can point things out. This does not mean curiosity wants to see things the way the teacher does.

Curiosity likes to be engaged with a problem. This engagement can be driven by different ways, such as a sense of progress, a sense of a threat if a problem is not solved, or just a lack of distractions.

Curiosity can like a goal, a sense that there is somewhere to get to. This can be external, such as a qualification or job target, or internal, such as a desire to develop a certain skill.

Curiosity loves story telling from a person, because stories can convey a richness of what is going on, better than it can gain from looking at things, or raw data.

HOW THE FINANCIAL MINDSET KILLS CURIOSITY

The financial mindset does not like curiosity much.

This is because the financial mindset is geared around results, which represent past performance. The curiosity mindset is geared around what might be possible.

To illustrate how fundamentally different these things are, consider that it may take 2 or 3 years for curiosity driven decision making to reflect in business results, and business results may reflect decisions which were made 2 or 3 years ago. So curiosity and financial thinking are looking at points in time 4 to 6 years apart.

And there are other types of thinking which the financial mindset has learned to like very much.

The financial mindset likes people who like status, because this status can be provided to people who achieved verifiable financial performance. A quest for status drives people to purchase expensive holidays and more expensive plane tickets, providing more money to the company for providing only a little more service.

The financial mindset likes brands, and people who care about brands, because brands turn something intangible and non ownable, such as the ability to make a handbag, into something a company can own.

The financial market likes sugar in food, because putting sugar in food is an easy way to make children desire it, ask their parents for it, and achieve sales. The more hidden the sugar the better, because it takes longer for people to figure out what is going on.

Curiosity hates all of these things. Curiosity sees a quest for status as a quest to shut down curiosity based thinking, a quest for guaranteed comforts and to live in a bubble, rather than to spend time with different people it might learn from. A status mindset can mean giving up bothering to do something interesting and chasing status instead.

Curiosity interprets "enthusiasm for brands" as a lack of desire to try out something new. Curiosity interprets a quest for sugar in food as a drive for quick satisfaction rather than the slower, harder kind, which only curiosity can bring.

There are a few financial people who encourage curiosity, because they are able to see beyond the short term numbers, and understand that curiosity is the only way the company can develop new products and approaches. Curiosity is needed for company staff to find paths over the inevitable bumps in the road.

Algorithmic and rule based investment

Financial decision making is increasingly made by rules, spotting patterns, and developing financial 'indicators'. Following rules and making calculations can be programmed into machines. For example, the higher your 'return on investment' the more investment you get.

To a degree, this has been a success. People have invested by developing and following their own 'rules of thumb' since investment began. Rules make it possible for a company to be more organised and structured in how it invests. Many investment decisions have been made at a level of disconnect from the underlying business for many years.

Computers can handle more data than people can and do it faster. There are patterns in share price movements which give early warning of a profit potential which an algorithm can exploit.

The drawback is that too much rule based investment, or 'investing in what worked before', means there is very little investment available for 'what might work in future'. A rule based investment company can only grow by acquiring start-ups which have proven a new way of doing it, and that's quite a restrictive way to grow, since start-ups can only be developed in certain ways.

And rule based investment will lead to companies being more standardised, and declining returns to investors – companies making 2 per cent margins.

Bringing judgement into investment

There are a few examples of where people are able to make business decisions based on judgement rather than just numbers. Top football managers. Steve Jobs in Apple. A venture capitalist who can spot good investments before they generate profits. People who are comfortable following the herd, as with Bitcoin and Tesla. People who are capable of creating and driving a herd of investors.

The energy and shipping industries have many professionals with deep understanding about their domains and how they work, so are able to allocate funds based on something which has not yet happened.

Perhaps the most important question is keeping the balance right – making sure we still have plenty of domain expert investors in the picture. And where we do have a domain expert allocating funds on behalf of an investor behind them, the investor behind them is encouraging the development of curiosity, not pushing the funds to be allocated purely on the basis of numbers.

Managing people by numbers

The problem of too much investment-by-numbers gets compounded when we bring the management of people into the issue.

Managers are making decisions about who gets put into which role and how they get rewarded.

Will they reward people who develop expertise through curiosity which can be of service to the organisation – or will they reward people who achieve performance based on what has worked in the past?

Is the manager themselves using curiosity to develop skills to understand which kind of people serve the organisation well, and how to keep them motivated and curious? Is the manager developing or demonstrating empathy levels to understand what staff really feel, and domain expertise to recognise whether the work is adding value to the organisation, even if it is not achieving obvious targets? Is the manager human-centric enough to be able to trust staff to get on with it?

Is the manager willing to invest in staff collaboration? Does it appear as allowing multiple salaried employees to waste time talking to one another, or does it appear as people increasing each other's motivation and understanding? Are we developing 'innovation clusters' or business environments where we try to build the best machine to develop something which worked well before?

Are staff being pushed too hard to achieve good scores in their "key performance indicators" or improve productivity, without an understanding that forcing increased focus or effort in one direction generally results in decreased focus somewhere else? Is this focus on a target driving linear thinking, when what is really required is much more abstracted or non-linear thinking?

Believing in a lack of empathy

Some managers go so far as to say that they ought to be able to think linearly about their work, it is their job to allocate investment based on past performance, and they are not interested in curiosity based thinking at all.

These people are perhaps the most dangerous of all.

This is the mindset which says, “if you don’t hit your numbers, you’re out”. Everybody understands the business needs to hit certain numbers to survive, and the staff have a role in this. But most sensible people see that the business probably does not want people to be solely motivated on the numbers, because these are the least curious employees you could ever have.

This is also the mindset of the heavy handed bureaucrat, who believes that the most important factor is that the people fit with the system the bureaucrat has designed, and makes the bureaucrats life easier – rather than the system itself actually delivering, such as making effective decisions about who should be allowed to stay in a country.

SOFTWARE CAN KILL CURIOSITY

There is a better way to kill curiosity than bad management – giving people bad software to use at work.

Software which demands or reduces work to a succession of form filling, approvals, button pressing and worst of all trying to constrain work to a place computers can understand.

Computers themselves are not curious, can only think linearly, and can only follow simple commands, although following them very quickly. If their linear thinking – or series of instructions – comes to an end, they stop running.

But this does not mean that working with software must be anything like this.

Software development should be focussed on elevating the way people work with the software, not reducing it.

Software can support curiosity when it is invisible, in the sense that the customer is thinking about what is happening in her domain, not the task of using the software.

Software can support curiosity when it is transparent, so you know what it is doing and why it is presenting what it is presenting.

Software supports curiosity when it is well modelled to your work, so it is accurately giving you what you actually need.

A person uses the software tools for their work in the same way that a gardener uses a range of tools for her work.

When the reverse happens, the software is complex, has a logic nobody understands, it means that if the customer masters how to use the software at all, the customer is still not learning anything about the domain.

Keeping technical people away from software

Developing software is a highly technical task, and so people with a technical mindset are good to have. But this is probably not the mindset you want dominating the software development.

Technical people are interested in technical things. So they may be more interested in their technical creation – the software – than whether it supports what people want to do with it. They may be more interested in looking for a technical solution to a problem than helping people find solutions to a problem.

And making software which supports what someone wants to do with it is really hard – much harder than just building software to do a task. It requires human skills to understand what people are thinking and needing while they work. This is something they may not even be able to tell you. It requires discipline to remove functionality which distracts from the main task.

Technical computing people may be focussed on the core computing concepts which have governed most of software for decades – calculations, storing and fetching data in databases, filtering data, building database schemas.

But the real world does not fit into databases. Yes, in the real world, we might filter, for example to work out which tasks in front of us are the most important, or which we absolutely must complete. But we are filtering a very different way to how a computer does it, taking into account many more variables, many of which a computer does not know.

The stereotypical geek could be defined as a person who does not wish to be a person. If that is true, this person should not be allowed anywhere near software design.

A focus on AI kills a focus on curiosity

Today's digital technology specialists are often big fans of artificial intelligence, its unknown capabilities, the science fiction writing about what may be possible, the 'singularity' where machines become more powerful than people.

If that is your mindset, why would you care about making software which supports a person's curiosity? So AI people, although highly curious themselves, may have a role to play in software's inability to support the curiosity of its customers.

Software people and understanding the real world

Making software for someone working in the real world requires an understanding of how the real world works. Not the entire real world of course, but the section of it which they are creating software for.

Every domain in the real world, from policing to the music industry, has its ways of working, the things which the professionals working in them need to find out, the ways they have worked out of doing it.

Police departments everywhere have methods which have developed over decades, centuries in some cases, for solving cases and reducing crimes. Helping people to understand what is going on – what might have happened, what trends are happening in a certain region, where the trouble spots are and how this is changing. In other words the systems have evolved to enable people to use their curiosity within a system.

Similarly the music industry has evolved to have professionals capable of working out which music acts might make it to the top, finding potential popstars, testing them out in different situations, developing them, making records, promoting them, trying them out on different audiences. The professionals involved don't know who is going to make it big beforehand, but they have established systems for figuring it out.

Any software to support curiosity in these fields must be structured to the way people think. Lists of cases, events, potential popstars, models of how things are progressing. And the people developing the software need to understand this too. The software must support the way people (already) work, not demand people work in a new way to fit the software.

Right now, police and music professionals work in the only way they can – with mental models, documents, spreadsheets and e-mail. Think what they could do with better software designed to support how they work and think.

Digital technology people and the status quo

Digital technology people often talk eagerly about 'disruption'. But they are rarely enthusiastic about disruption when it comes to their own business models.

Look at what is happening, rather than what people say, and you see a very different picture. For example software companies' sales teams are focussed on trying to tie clients down into purchasing complex software packages they cannot easily extricate themselves from, in subscription arrangements. Thus avoiding disruption in the software company's revenue stream for decades.

And how often do software people question the contribution of software in general to society? They seem to not notice that software in 2018 is often much more complex than it needs to be, leading to massive challenges with both interoperability of different software tools, and a cybersecurity nightmare.

The software industry is a key component in a society which is seeing large scale dis-satisfaction among the public, leading to votes for populist leaders and causes which people believe can stop them from feeling like losers.

But software people are too happy with whatever the software giants are delivering them, believing they leave no gap unfilled.

HOW SOFTWARE CAN SAVE CURIOSITY

How can software best serve curiosity? By getting out of the way.

Or more precisely, by being as invisible as possible.

Just like an office building where people come to work and get on with their work, without having to think about the building itself. It provides everything they need of course, but it does not obstruct them.

The customer should feel in charge of the software – the way a builder feels in charge of his tools.

Ideally, the customer has been involved in constructing the software itself, so it works exactly the way the customer wants it to – equivalent to a person who has been able to design their own workplace.

Continuing the building analogy, we can say that making a building which serves our needs invisibly takes much more effort than just creating a building. Everything needs to be just so, and do it reliably.

It is pretty difficult to make the perfect building with imagination alone – to do it well, you need to have seen lots of buildings and how well the design "works". Probably you'll follow a number of models or mini-designs to make up a larger design, in the way that most kitchens or bathrooms are similar.

And software design is much harder than building design in that it will need to anticipate what kind of questions the customer might want to answer with the help of the software, including questions at different steps of a multi-stage process, like booking a holiday involving flights, hotel and activities for family members.

The customer will want to understand the ‘logic’ which the computer follows, in the way that we understand that a flight booking system will offer higher priced flights at holiday times and closer to the date of travel.

The customers’ needs are not static – they may vary with what the computer system has to offer – in the way that someone’s preferred flight depends on the prices and how important it is to travel on that specific day.

And software design is much more flexible than building design is – in that a piece of software can be changed or merged with another software at some point in the future. That ought to be a possibility, anyway.

Always remember the customer is completely analogue

The world outside software has always served “completely analogue” customers. Restaurants, artists, flower shops, composers, writers, luxury goods firms, have learned how to serve customers who do not function through computer logic. Perhaps the vast majority of human beings still live entirely in this totally analogue world.

Yet there is a tendency of software people to build tools as though the customer is a computer or robot, rather than a person. You tell me your needs or provide fixed information, and the software will add it to its database, compare with something in its database. If the input information is incomplete or wrong, well that’s not the computer’s fault.

As an illustration, consider a flight booking system. The computer system is designed to accept the customer’s choice of flight destination and date, put that into a database of available seats and prices, and then return the customer data about whether the seat is available, what the price is, which the customer can then accept and pay for, or decline. Come back another day, the availability and prices are different.

But here's some possible customer scenarios, which you might recognise from your own experience.

I need to get my family to another country sometime during the summer but can use a range of different airports to depart from and arrive to and a lot of flexibility about the date, but price is important because I am buying five tickets.

I need to get to a certain location at a certain time for a business meeting, on a limited budget. I could fly in the morning before the meeting, or get a cheaper flight the night before, if I knew how much a hotel costs. I have a preferred departure and arrival airport, but could use other airports, if I knew the time and cost of ground transportation.

I'm not sure if I need to go to a certain meeting, so would like to make the booking later, but I would like to know how much the price might increase if I do that.

I booked a flight for Monday evening departing Monday 12.10am, not realising that means the ticket is for Sunday evening. I booked a family holiday in September rather than August. I booked tickets to be posted to the wrong address. I didn't realise I had to apply for a visa to go to that country. That was a silly mistake I made, but surely the software could have warned me about it?

I want to try out many different options very quickly, to see it if might be inexpensive to take my family on a short trip after Christmas. But it is very hard to see what the final cost will actually be, without going through screens trying to sell me insurance and car hire,

Building a flight booking software to accommodate these requests is within the capability of 2018 technology and would perhaps justify its development costs from attracting increased usage and so more bookings.

When curiosity works with data

Many people at work use data together with their curiosity, to better understand what is going on. So the way their software handles and presents data is important.

Data can be presented in ways to support or discourage people's curiosity.

On one hand, easy tools to dig into data, and get a richer understanding of what is going on, can really support curiosity. You can see that there might be a place you can get to.

We use curiosity to learn more about what the data actually represents and what it can do. We learn how data can be put together with other sources of information, such as our conversations, or trends we think we observe. We can make guesses or judgements about what might be happening, and then see if the data supports this idea.

On the other hand, frustrations with data which turned out not to mean what it promised, or too complex data, can turn curiosity away.

Similarly, the 'internet of things' trend can lead to installation of multiple sensors generating data, which initially feels helpful, but can be followed by a realisation that it is very difficult to translate multiple sensor data into an understand of what is actually going on.

People who make decisions solely on data have to treat the data as concrete – because that's all they have. That is something very different.

The software architect

To construct software which supports people's curiosity, a critical role is the 'software architect', the person who designs the software. Nothing kills curiosity and good work like badly built software – in the way that nothing stops people doing good work in an office building than building problems. The focus moves onto something other than the work.

The traditional role of the 'software architect' is usually someone who works on complex software packages, and is mainly concerned with the construction of the software – areas such as security, integrity, performance, and scalability. In times gone by, say over a decade ago, building any software was much harder than it is today, and so it made sense for the software architect to be entirely focussed on technical aspects, guiding the team of developers.

But a software architect's role today could be more similar to a building architect's role.

The building architect focusses on how the building will be like to live and work in, how it will fit with the surrounding environment. The architect will get involved in the basic decisions about the materials and services which will be used to construct the building, in order to deliver this. Non people aspects of the building design are referred to engineers.

Similarly, for software design, the role of the 'architect' could be far more to understand the environment within which the software will be used, and how people work in it, and ensuring that the product fits with this. And also defining the basic commercial products and services which will be used to create the product, alongside custom built products. Then the engineers can take over.

A building architect will have areas of specialism and develop a detailed understanding of the needs of the domain related to that specialism. Some building architects work on large tower blocks in cities, some work on interior design, some work on eco-friendly developments, some work on house extensions where people will live.

Similarly, a software architect should have an area of specialism – the skills of building software for the upstream oil and gas industry, shipping industry, or any other industry are very different, and all require in-depth understanding of how the domain works.

Just as with building architecture, software architecture does not need to be technically complex. It is more about having a better understanding of the right thing to build in a certain place for a certain need.

A difference between software architecture and building architecture is that it is technically possible for software to be fluid – just like knowledge itself. It should be possible to continually improve the software, providing an ever better fit with people's needs. It could be joined up with other software and melded into something else completely, providing new ways to support people's curiosity and understanding of the world.

In practise though, todays software is rarely thus changeable, because it is usually based on rigid database structures. It makes today's software as rigid as buildings.

New approach to software sales and marketing

Software which does the most for its customers is not necessarily big and complex. So it isn't necessarily what the sales and marketing people want. But perhaps the answer is to change the way software is sold. Less about whizzy technology, more about actual value.

The software industry is not the only industry to provide overly complex products in a quest for higher margins. Automotive, tourism and luxury goods are sectors where customers are routinely persuaded to buy much more complex / expensive products than they actually need.

A difference is that in the software industry, the price of the purchase of non fit for purpose / overly complex products is paid for by people other than the person who made the purchase decision.

A price is paid by technology staff, who have to meet the challenge of making this complex thing work and integrate with whatever else is going on in the company. A price is paid by staff members, who have to figure out how to use this software to do their work, and the overall company performance is not as good as it could be.

A price is paid by the company, who will have to pay for many hours of expensive IT and data management time if it ever wants to switch away from using a complex software package, or make it interoperate with other software.

A question, then, is how should a software company make money if it is only selling smaller, simpler, more standardised software components which a customer can easily stop using, rather than tying customers into complex expensive systems.

Perhaps the answer is to look at how other industries do it, such as groceries, which sells many small components which a customer can easily stop using, with no cost of switching to another provider at all.

Software traceability

Software transparency and traceability is very important for building software products to support curiosity.

People want to feel they understand their tools as well as a builder understands his machine tools, or a car driver understands his car.

Understanding a tool does not mean understanding how to build it, or how it all works. But it does mean understanding how the tool will respond and its internal logic.

Furthermore, the clearer the software is built, the easier it is to develop it (including by people who did not build the original software), thus maintaining the transparency.

If people improving the software do not fully understand it, it is too easy to get into a mode of “let’s see if this works”, further adding to the lack of traceability.

Data traceability

Curiosity will also be better supported if it is able to understand how the data was put together. Any ‘data’ passed around as part of a decision making process is much more than a number – it is a compilation of numerous statuses.

For example, you might know how many Air Miles you have. But you don’t know if your Air Miles were awarded as part of the transaction you are currently making. If you buy a flight and hotel together, can you use the Air Miles from the flight as part of your hotel booking?

As another example, economic data can be much more interesting if you have a sense of how it is compiled and where the strengths and weaknesses of the compilation method are, and what else may be going on.

Cybersecurity

Cybersecurity is becoming a major commercial force in software development, in that the threat of a hacking has more power to change how software is written than the customers.

Hackers love complex software because nobody understands it fully. They can insert code into it and no-one knows it is there. They can find pathways for using the software in ways other than how it was designed.

The way to combat this is to keep software as simple and clear as possible. Nobody has ever managed to hack Notepad.

Curiosity and advanced digital technology

Advanced digital technology, such as analytics and artificial intelligence, can work in harmony with curiosity, if we get a few things right.

Apart from the world of algorithmic trading, all big decisions are ultimately made by people. But computers can play varying roles beneath that, automating some of the smaller decisions and serving up data to the decision maker.

So the computer is a servant to curiosity.

The computer is helping its curious customer to understand the situation, spot patterns, continually learn, and be aware of problems emerging to which the person should apply attention.

Along the way, there could be a lot of labour intensive work which a computer could do better than curiosity. For example, in oil and gas subsurface analysis, there are many complex studies of the subsurface, getting a picture of the geology built out of large volumes of seismic data. A person can show a computer how to do the interpretation, starting something off which the computer can copy the way the person does it.

Advanced software tools can help someone understand what is going on, if it means they can ask more sophisticated questions of the data and get replies, get ideas about trends and see if they match what is actually happening.

A hurdle to overcome, if AI is going to nurture the curiosity of its customers, is in encouraging the developers of AI to accept that as their role, rather than making machines which can use their intelligence by themselves. Is the computer competing at playing Go, or is the computer helping the person play Go?

People who work in the field of AI are excited about what AI can do, which makes complete sense. They see a pathway for AI to be able to do more than people can (and think AI is perhaps already there). If you engage these people in a discussion of the strengths of people vs machines, they can quickly tell you how fast AI is evolving and it is only a matter of time before machines can do everything people can do.

Very few AI professionals appear to be interested in judging what AI is actually capable of today. You can make your own judgement of this by seeing how good the AI tools you are provided with actually work, for example the tool on your computer or phone which offers to help you if you get into trouble, or perhaps the tool which is supposed to move the scam e-mails out of your inbox.

A common argument from AI people is that many elements of many people's jobs could be done by a computer today. For example the element of a bus drivers' job which involves driving the bus down a straight road with no obstacles, passenger interaction or prediction required. Or the element of a radiologist's job which involves spotting where an image is similar to another image it has seen before, but not explaining this to a customer or filtering out the false positives.

In the real world, we usually expect people to have decades of experience working in a specific domain before they are able to make decisions. Perhaps that indicates the level of sophistication required. Unless, of course, these decisions can be made purely on numbers.

Can the customer be involved in building software?

One way to end up with software which supports curiosity is if the customer is able to get involved in building it or defining it.

The customer has more motivation than anyone else to get the tool just right, and more understanding of what is needed.

We have analytics packages which are designed to be operated by people who are not data scientists – anyone can take some data and try to find a story from it. Plenty of people who are not professional data scientists can also write code in Python, the data scientists' preferred language.

We also see a growth of 'low code' software, which is software generated automatically from an outline plan (as a simplified explanation). Currently low code is largely used for very simple applications, such as online forms, but there's no technical obstacle to low code being used in far more complex tasks.

Could we one day see workplace software with a 'maker culture' – where it is commonly place for people to build their tools for their specialist needs. Just like in the 'maker culture' which already exists for electronics and physical components, the software components would be standardised, designed to be put together by people with amateur levels of expertise.

Just like in the 'maker' world, there could also be blueprints of something someone else has put together which has worked, which you can follow, showing how well it worked.

-

SOFTWARE MODELLING AND CURIOSITY

The most important factor of software to support curiosity is how well ‘modelled’ the software is against what curiosity actually needs.

Curiosity does not live on her own, developing an understanding of something like a Victorian scientist. Curiosity works within a domain – anything from marketing, nursing, organisational management, policing. The individuals have roles to work out the most effective way to apply their limited resources, keep services running reliably and avoid problems – by understanding how their domain works.

So software for curiosity must be designed to tell the customer what she needs to know, including things she doesn’t yet know she needs to know, for example emerging problems. The software shouldn’t distract her with things she doesn’t need to know, or demand she spends time fiddling with software.

That’s really hard to do, because it involves understanding what (for example) a nurse might most want to know at 2am based on whatever data is in the computer system at that time. Perhaps nothing at all.

The customer might be co-ordinating one task with another one, for example a purchasing manager who needs to ensure that the right spare parts are in the right place for upcoming maintenance work.

Building software to serve the needs of the domain takes a lot of initial imagination, guesswork, and continual improvement. But a compensating factor is that the actual domains we work in – are often not changing at all. Tourism, oil and gas production, law, work in the same basic ways, sometimes for centuries.

So once a model is built for software to serve how people work in a certain domain, it could be usable and extremely valuable for many years.

What modelling means

The definition of a model is a representation of something with some of the detail taken out, so it is easier to mentally manage than the actual thing.

A map is a model – it is a representation of that part of the world, but with detail taken out so it can fit on a piece of paper or screen.

Building a model is important in software because it gives you a structure to build code around.

Conversely, if you start building software with a design for a database, and try to constrain it to what the real world wants, you'll end up with .. the sort of bad software we usually see in today's workplaces.

Open source code, or APIs, could be defined as "code without a model". So they are often very difficult to work with.

If we designed cars the way we designed software, we'd start with an engine (equivalent to a database), then design a frame around it, add in something to keep passengers warm and dry, put in some controls – like a car from the early 1990s. It would drive, in the hands of someone who thoroughly understands the machine. But would not be a great or invisible experience.

Modelling a tribal village

Here's another analogy for a software model. Consider a village in tribal times, with a chief and 10 advisors. The chief understands the expertise of all of the advisors, and at the tribal meeting, knows who to ask for, for advice on each question.

The chief has a model in his mind about who knows most about what – based on what they have shown to be best at understanding in the past.

Modelling for software can be thought of in a similar way. Is there an analytics algorithm that can be brought to play when there is a certain problem which needs addressing? Does the chief (or expert) understand how the algorithms work and what assumptions they make? Which algorithm has proved to be most useful in the past?

Lawmakers are also modellers

Lawmakers are also modellers – they build mental models about how people behave, what shortcuts they might look for, what would prevent them from taking that shortcut, and design rules to try to constrain behaviour to the right outcome for society.

It requires being precise and methodical, thinking clearly about how the world works. Also an understanding that the real world is more important than the law itself, and the law should not impede the right sort of behaviour in the real world.

White board activity models

An activity model can also be drawn on paper, or on a white board, as a series of boxes connected by arrows.

These can show what happens when, how a decision is made.

For example the process by which a health diagnosis is made, or the process a police investigation is made, or the process a passenger takes through an airport.

Standard conceptual models

A "conceptual model" is the design of something at a "conceptual" level, explaining what the elements of it do. Making it a 'standard' means that it can be shared.

For example a standard conceptual model of a car could explain that it has pedals which do specific things, a steering wheel, indicators, an engine, front seats and back seats.

The benefits of having a standard conceptual model for a car are so obvious no-one has stated it as such.

But in the software and data world, the benefits could mean that different companies could build software around the same conceptual model and it would all easily interoperate. Or two companies could build software for related tasks, based on a standardised conceptual model, and the two software modules would work together straight away. Not just in the data parameters they use, but also in the way they 'understand' the world.

A conceptual model could be defined as a model for software without any code – so it might include illustrations of screen interfaces, how people would work with it, and what it would do.

In future, perhaps standard conceptual models for companies could be as important as the software industry is today. Software would just become a commodity, with code generated automatically from the model. The real value would be in the model itself.

Imagine if you own a rich model for how an airport should operate, or a shipping company. Someone setting up, or improving, an airport or shipping company, could purchase your model and follow it. The price may be high but the value a good model would deliver to a large organisation would be in the billions of dollars.

Modelling the interaction

Part of the modelling work is to model how people will actually work with the software – this is where we get to the "user experience part".

Modelling the user experience becomes far more critical the smaller the screen, or the more interface limitations they are.

Making software which someone works with, using a keyboard, mouse and large screen is relatively easy. Making software which gives out the right information at the right time via a text message spoken at you from a watch, or a voice activated head mounted display, or on a limited communication bandwidth smart phone, has much less room for error.

Abstraction and rigidity

One of the critical skills when modelling could be called 'abstraction', or our ability to reduce an activity into the core or processes behind the immediately visible details.

Consider a queue of people waiting in a passport control at an airport. Do we view this as a limited number of data points – people, nationalities, passports, space, inspectors – or do we see this as a rich story for each person about where they are going and why?

Abstraction is important in software development because there is usually far more to life than what is seen from what someone types into a computer, and the software needs to cope with this. Why is someone purchasing this component now? Could the software use this understanding to do more to anticipate their needs in advance, spot errors, identify where a different component would be more appropriate to the needs, or if another component already in the company's inventory may meet the needs? Is the software providing information which helps the person do what they need to do, or is providing useless information which drains the person's mental energy?

The better the understanding of the person's real aims, the better the software can be designed with precisely the functionality which is required – and less unhelpful functionality, which increases the complexity of the software.

The ideal for the customer is if the software fits "like a glove" – doing exactly what is needed in the right way and no more.

If the abstraction is wrong or not done at all, it increases the rigidity of the system – like a road crossing we have to go out of our way to use, or a single bus to our destination which leaves at an inconvenient time. Avoiding the abstraction issue, by having lots of road crossings or buses, increases costs and complexity.

Building this perfect software can take the same amount of programming time as simple, rigid (badly abstracted) software, and much less than software with a multitude of functionality we never use. The difference is it gets the abstraction right.

Doing the abstraction work of building software requires a mental skillset very different to the one standard programming work involves, such as empathy, and an ability to be controlled by the customer's needs rather than your own view of the world.

HOW CURIOSITY CAN DO MORE FOR BUSINESS AND SOCIETY

What particular areas of society which might operate better with an organisational environment better able to support curiosity, and better digital tools?

Much of the answer can be described with the word management – and not just management of people.

It includes people who manage the reliability of services – anything from water, electricity and fuel, to groceries on the shelves, to schools and hospitals. People to manage any aspect of an organisation's delivery.

It includes people who manage something in their work, such as a nurse who manages the patients in their ward, or the manager of a coffee shop.

It also includes organisational decision making, management in the more traditional sense of the word. Curiosity drives the development of expertise we need to make good decisions.

In all organisations, good decision making leads to reliable services, rewarding jobs, happy customers, the ability to achieve more with less resources and be more adaptable. Bad decision making means delays and waiting times, shortages, outages, breakages, queues, business failure, and stressful, unsafe jobs. This can apply to banks, oil companies, shipping companies, retailers, tourism providers, police, hospitals, emergency services, schools, transport, and any other organisation.

Better support for curiosity could nurture many small businesses, from companies who figure out new ways to do things. It could help reduce some of the size advantage large businesses have, if their size gives them more cover to take more risks – smaller companies could do the same with smaller risks but better decision making.

Better support for curiosity would be helpful in all domains where there are large amounts of data, because it would encourage people to work out what the data is telling them. For example cybersecurity, government policy, management, energy, climate, health.

It would help in all roles which require understanding of a changing risk picture – including heavy industry, and complex financial organisations. Safety requires more curiosity than anything else. True safety is more about the mindset than anything else – and the mindset at all levels of the company.

Curiosity in organisational management

Anybody working in organisational management has to understand a great deal of moving parts. Customers, staff, money, the markets, changes to the inputs. Customer queues, staff stress levels, how much can customers afford to pay. There’s a mental model about how the organisation works, which needs to be continually updated.

Many companies have a strategic change management plan. Managing this change will require monitoring the status of the company, and whether the plan is working.

The company might have overall key performance indicators or targets it wants to achieve, for example a school which wants to get a certain overall score from its inspectors, or certain improvements to the financial performance.

In order to reach them, the company staff will need to gain understanding of which indicators are showing them something useful, and whether they are damaging performance by pushing people to achieve one target at the expense of another.

The recruitment department of the company will want to be aware of the roles coming up to be filled, the availability of potential candidates, how the job market is changing. She'll develop skills at assessing whether a candidate is suitable.

A staff manager of any kind will need to be continuously learning about what is working and not working with her techniques to improve performance from her team.

If she is a manager of a hospital, she'll need to understand the complex picture of the organisations resources, and the demands on it, and how this is changing.

If she works in mental health services, it will take a great deal of curiosity to drive out useful insights from the various data available about what is working and how it can be changed.

Acute care services also benefit from large amounts of curiosity. The hospital promises to provide immediate assistance to people in need, including collecting them via ambulance, and having all the resources immediately available. This is impossible to guarantee, since it requires use of a limited number of assets and an unknown demand. But can the service being provided be improved?

Education services need to provide the best possible education with the available resources. There are many possible inputs – changes to the recruitment strategy, different ways to manage staff. There are different training methods – including technological methods. A curious manager can get better at working out what is working.

If she works for the police, she'll use curiosity to learn both about the changing big picture, and the specifics of any crime the works on. The big picture includes local hotspots and trends, as well as changing police capabilities. There will also be economic considerations – spending on people, vehicles and facilities – how well is it working? Can the assets be better deployed?

A critical factor with a police service is the emergency response capability. Getting the right number of police to an emergency situation quickly seems to happen by magic, but must take a great deal of planning, including deciding what work is less important and can be immediately dropped. If it is a complex event, the police response will need to be co-ordinated.

Financial administration, banking, investment

Most businesses have a major challenge with cash flow, because it involves predictions of when funds will arrive and when they need to leave. This requires understanding of the patterns of payments and invoicing beyond the capability of any software program yet developed.

If the company is managing capital, there will be many different options to consider – short term, long term, safe, risky.

If the company is making large financial transactions, there will be different options for how to do them, and a need to check they all go smoothly.

Companies may benefit from being able to understand their own financial transactions in different ways – find out ways they are spending money with little return, or being tight with funds where they could afford not to be.

Banks need to make decisions about loans. They may make smaller lending decisions with algorithms, but more complex or larger lending decisions rely on staff members getting a quick understanding of the situation of a company or individual.

There may be opportunities to offer better lending products for people and individuals in need. Currently banks usually only offer small loans for a one off expenditure such as a house improvement, on the basis that the person has an income but wants to pay for a large ticket item before the money is all earned.

Perhaps banks are missing out on another lending opportunity, such as people who are out of work for a short time and need a loan to tide them over. Currently such a person only has recourse to the most expensive lending possible, such as a credit card.

There may be better ways to help people on low income manage funds through software, for example someone who wants to save a chunk of their income every month and only spend a limited amount of it.

Operations management

A large number of roles can be categorised as "operations" – which can mean anything from border control to industrial plants to city electricity suppliers. Every operator needs an overview of their operation, and an understanding of the trends which can cause disruption to the service their job is to provide reliably.

The challenge starts with design of the facility, taking into account knowledge about what works and what doesn't work in the past, to build something which can do the job as well as possible. (Engineering is a form of this).

Project management and construction will benefit from curious managers who can spot problems emerging before they become very expensive. Making sure the various resources – machines and people – are used effectively and deliveries arrive on time.

Once the facility is operating, you'll need to understand common problems and work out how to predict them in advance with the help of data. A curious manager will explore better ways to operate things, for example with "servitisation" where more of a management task can be passed onto a supplier (for example a supplier undertakes to keep something permanently stocked up, rather than delivering on receipt of an order).

There will be purchasing required – a curious purchasing manager will learn about how well past purchases fit the needs, performance of different suppliers including both products and delivery. Whether the spending is matching the returns. A curious purchasing manager will develop high performing systems, automating commonly done tasks, and minimising re-entry of data. Also keeping on top of inventory.

If she works in the electricity industry, she'll use her curiosity to help keep demand and supply balanced. What are the trends which change how things work, for example with spikes in electricity demand (kettles in the royal wedding), changes to the supply (renewables and the solar eclipse)? This becomes more complicated with the promised changed to more local electricity networks, with more renewables and "demand adjustment".

Cybersecurity or IT

The field of cybersecurity has real demand for curiosity. What is happening here on the network? Can I see a trend which indicates something malicious going on, for example lots of network traffic happening at the same time as an important business meeting, which could indicate the meeting room video camera has been hacked and is sending a video feed to a hacker?

All network activity – including hacking – leaves a signature – the challenge is understanding that signature in the masses of data which computer systems generate.

In other fields of IT, a curious manager can keep on top of how the system is being used, and where people are finding problems with it regularly, which are not being fixed. A curious manager can make sure that the resources are applied to the problems or demands – so there is enough internet capability available at all times for example.

Marketing and retail

In marketing, sales and retail industries, you need to understand the market, an invisible thing, which only appears to you as a story which you make up or model, behind the sales figures, which you do see. You see other signs, such as activity, website traffic, newsletter sign-ups. You need to understand what drives a customer to spend.

For online sales, you will have huge amounts of data at your disposal – how people are finding their way around your website – but making something valuable from it is another question.

The curious marketing manager will want to understand how a change in how something is presented leads to a change in the response from the market.

She might also want to be aware of what competitors are doing and how they seem to be motivated.

Working in development

The developing part of the world needs more to be achieved from lower resources. Getting more food from the land and get it to market and stomachs, providing healthcare effectively on low budget, provide effective training to people, providing better society governance to take away corruption and crime, providing energy and water supplies reliably, help people do more with less money, reducing the cost of financial transactions.

All of this demands enormous curiosity to figure out what works and what doesn't – and tools to drive this understanding of what works and what doesn't.

CURIOSITY GETS KILLED

Are we are seeing the slow death of curiosity in our society – with an unstoppable force?

We live in a society where banks and investors hold the most power, making decisions about which companies get funded – based on what worked before. They see understanding what has worked in the past as a core part of their job.

Their forces and culture spreads its influence to many other areas of society. Education establishments, including secondary schools, are often managed by people with a financial background, or by management consultants trained from working with investors.

Lots of people in the tech world don't like people much anyway and a world run by robots – or people forced to act robotically - was fine for them.

Schools are chasing targets, being punished for not reaching targets, trying to keep up with education scores in non curiosity centric education systems. They're slavish to management consultant thinking, and the management consultants in turn are slavish to investment banks and their rear view mirror thinking.

Businesses start chasing targets, and start making their suppliers chase targets. Target driven satisfies a certain type of person. It feels right, judging and scoring people and picking the winners. Competition working at its best.

Curiosity gets progressively tightened away until there is no-one in a powerful position left who cared any more.

Plenty of people never had much curiosity or desire to learn anyway, and they don’t miss the curiosity driven people, where there was a small chance they could develop some kind of clever insight for the first time and make them look silly.

Meanwhile mental health problems increase from people who cannot find their way in this word. Nobody cares much about people with low incomes or bad mental health. Nobody even notices them. Curiosity completely dies and everybody becomes machines.

CURIOSITY COMES BACK TO LIFE

But that's not how the future turns out, because curiosity can never be killed – as it was never killed in the depths of communist Europe.

Even the most status orientated, non curious people, like the prestige which only curiosity can start. Prestige from having prestigious universities, theatres and great art. And the economic advantage which comes from developing new products rather than copying other peoples'.

Eventually, after nearly killing curiosity, society learns how to nurture it again.

The education system recovers the art of training people to think for themselves. The government services use expertise to work out how to provide reliable services at very low cost, with funds available to support people who cannot do it themselves.

You can't kill curiosity. Because it lives on inside just about all of us.

But it may not be in our lifetime that curiosity comes back to life – unless we act now to keep her alive.

If you don't believe there's any room for improvement in the services which our organisations offer, think about all of the people dis-satisfied in today's society, or who believe themselves to be losers from it.

All of the people who voted for Donald Trump, Brexit, and support populist parties across Europe. They are dissatisfied and looking for better answers. Their dissatisfaction does not show up in any economic indicators (hardly any anyway).

Curiosity will live on, whether society supports it or not.

If you are curious to discuss more about how society can better support curiosity – with the help of better software, our company, Software for Domain Experts, is considering running a series of events on curiosity driven software.

We'll look at how it can be better constructed, how it can help support people's learning, where the business opportunities are. You can join us and find out more at softwarefordomainexperts.com.