Software Driven Expertise

Karl Jeffery and Dimitris Lyras
London, August 2016

August 2016

Published by Software for Domain Experts Ltd
39-41 North Road, London, N7 9DP, UK

Sign up for our newsletter at

www.softwarefordomainexperts.com

Software Driven Expertise

By Karl Jeffery and Dimitris Lyras

Foreword

This book is about how to make software that helps experts do better work.

Help them better understand their working environment, continually develop their expertise, and achieve their goals (which are probably set for them by others).

Here's an example to try to explain what we mean – software for primary schools.

Primary schools in an area of North London have recently installed new software which helps keep track of test scores, and can be used by teachers to fill in school reports.

The software can be used by the school head teacher, or regional education authority, to analyse results for pupils of different genders and ethnic backgrounds, and make sure (let's say) black boys are doing as well as Turkish girls.

It can be used to plan a school's finances, manage attendance data, and manage parental communications.

This sounds like a good story. But is it really helping the relevant experts?

School head teachers have many demands on their time, including urgent problems of the day with certain pupils, staff members or even parents. They may have a specific goal to work towards, such as improving school performance in maths. Analysing test results by gender and ethnic group is probably not too high in the priority list. Test results are one component of a complex web of situation awareness a head teacher needs, and she doesn't need fancy software to access them.

Having a computerised system to fill in school reports does not offer any obvious benefit over writing reports in a paper book by pen, as teachers did in the pre-electronic era, unless you count the ability to copy-paste the same entry into many different reports.

There are benefits to being able to share information easily to authorised people, but this could be done with free software such as Google Docs, or just a shared network drive – and it might be easier for everyone to understand what was happening if it was done that way.

And this software sounds like it is basically a group of relational databases. Its functionality is basically putting information into databases and retrieving it from databases. The software company has selected elements of school life which can fit into relational databases because they are already in a database format. Attendance records, e-mails to parents, financial data, test data.

In the 1970s software was mainly putting information into relational databases, because that was all software could do then. Bank transactions, library indexes, financial data, airline tickets, company purchases.

But in 2016 software is capable of doing so much more.

Let's imagine educational software which is designed to support the situation awareness which the head teacher actually has.

Coming back to our school head teacher example. Her ideal software would fit around her own mental model of the school. That might include progress towards a specific goal, such as improving performance in mathematics, or improving pupil attendance; it might include 9am checks on whether all the staff have arrived and the building temperature is in a comfortable range.

It may include tracking a small number of 'problem' children; it may include preparing for an upcoming inspection; or she may spend 90 per cent of her time dealing with 'urgent' issues and prefer software which can help her keep track of what these are and prioritise which ones she is working on.

There will be commonalities between the needs of different head teachers, but also many differences, so the software will need to be customised for each individual head teacher. This is not a one-size-fits all software market.

But in 2016, software can do far more than try to constrain real life into relational databases. We have massive advances in 'modelling' practise, which try to build a model of the relevant part of the real world so software can be built around it; we have 'low code' technology which automatically creates software code from a model, promising much lower costs of building code, code which is more reliable, code which can be customised to individual expert users at a viable cost.

We have a range of simulation and analytics tools which can support and contribute to the model of the real world; and we have the possibility that the people who will use the software (who we call 'domain experts' to be directly involved in its creation, perhaps with software customised just for them.

Perhaps even more interesting, this software would help her to continually improve her mastery of her task of managing a complex inner city school.

Real world mastery does not come from formal education, it comes from – as the military say – the OODA cycle. Observe, Orient, Decide and Act. Get information, determine what it means and what you can do about it, make your decision, do something, and then get information about how the situation is changing.

Software can't do much for the 'decide' (would you like an automated system making decisions about your child's education). It can support the 'act' (which in practise might just be handling the communication of the decision). But software can do far more for the 'observe' and 'orient' - by helping gather information and present it in a way which makes it easier for an expert to make a decision with.

To make this happen, it needs software people to understand the potential, it needs organisations which employ experts to understand the potential, but perhaps most of all, and it needs the experts themselves to drive the movement to better software.

Perhaps, rather than go to a large software house, she could be served by a small software company – with 8 employees including former head teachers as well as software experts, and with staff members who would spend time in her office watching exactly what she does, providing a service not software in a box. These 8 people would customise the range of platform tools available, to provide her with the most useful possible software which would help her be aware of what she needed to be aware of.

Even more exciting, a move to this sort of software means a move to a move expert-centric society – where experts are more valued – rather than a process-centric society where experts are valued less and less.

This is where our story begins.

Table of Contents

Foreword 4
Part 1 - Software Driven Expertise 11
Introduction 11
The value of expert work 12
The software solution 13
How would you like to be valued for your expertise? 14
What expertise means 15
What expert life should be like 16
All expertise is different 17
Goals 18
The organisation and you – the expert 18
Call centre example 20
How we develop expertise – the seafarer model 20
Let's introduce "Software for Domain Experts" 21
Coming back to software 22
Software for Domain Experts is not designed to impress 23
Why this doesn't happen 24
The belief that automation and analytics is the future 25
You are my hero 27
Note 1: This is not an academic book 28
Note 2: where we're coming from 28
Part 2 – How to build software which drives expertise – the expert perspective 30
Introduction – software which makes experts more valued 30
The expert mindset 31
We understand systems 31

The environment for expert work 33
The caveman mindset 34
Expert’s working mode 34
The expert’s mental model 35
Experts use expert language 36
Work sectors this applies to 38
Police 38
Cybersecurity 40
Complex travel planning 41
Fintech 41
The control system operator 43
Part 3 – how to build software which drives expertise – the software developer’s perspective 44
Introduction 44
Changing the technology culture 44
Like a butcher shop – a small service company model 45
Doesn’t software change people’s jobs? ... 46
Software decentralisation 47
For software engineers 48
Designing around situation awareness 48
Domain Driven Design - Building software around a model 49
Transparency – understanding how it works 51
Ontologies 52
Low code 53
Analytics and artificial intelligence 54
The internet of things 55
Part 4 – looking at the obstacles 58
Part 5 - This is worth doing 63
Introduction 63

People who will pay for SFDE 63
Can’t we love experts? 64
Use the same software for training 65
A pathway to rescuing the news industry .. 66
Helping India 66
Contributing to social inequality 67
Climate 68
Conclusion 69

Part 1 - Software Driven Expertise

Introduction

You probably don't feel that you are valued enough for your expertise.

That's to say, most people don't and you're probably part of most people.

By expertise I mean the understanding you have about how something works, which you've built up over years of watching it, seeing what happens, seeing how the changes you tried changed this thing, and learning from that.

By valued for your own expertise, I mean people recognise the understanding that you have, not what the understanding enables you to do.

Note – we are not talking about software experts here, who are reasonably well valued for their expertise in 2016. We are talking about people who have expertise in everything else.

Life tends to value us for what we are able to do, like filling a slot in a corporate structure, being able to do a certain task.

That's not inherently bad - so long as you find a way for life to value you like that, which means a job you are happy in, and those can be hard to find.

Perhaps - perhaps probably - even if you're in a job, you often feel that you have the understanding to know what the organisation should do, and you're not being listened to.

Our society is not very good at valuing expertise - with the possible exception of software development expertise.

Perhaps you feel that life is about a hunt for a good 'gig' - a good position where you can be paid well for what you understand. But that's not very satisfactory either - and being able to hunt for a gig is not the same as being able to develop expertise.

You might feel the need to 'dress to impress' aka come across as someone with valuable expertise, harden your personality, be someone you're not because everyone else is doing it, because you know you can't be valued for what you know, you're valued by creating a perception that you're valuable.

The value of expert work

But in order for society to run smoothly, and provide everything which we depend on, such as eggs, energy and education, we need people to bother developing expertise, and we need to make it worth their while, or at least provide them with a living and some respect.

There are many visible problems which more of the right expertise in the right place might solve, which are clear if you look at any news website. Terrorist attacks, poorly managed hospitals, long queues at border control, congested roads, poor political judgement.

That statement is not intended to mean that the professionals running all of these services are not experts, but just to illustrate the areas in our life which are dependent on expertise, and often lead to results which are worse than we would like.

Then there are plenty of invisible expertise problems – the problems many of us encounter, but which do not make it to national news. When we can’t get what we need from hospitals, schools, police, transportation companies, telecom providers.

Even more invisible are the millions of people who can’t find work which they feel stimulated by and which fits with their childcare needs or anything else. Expertise is needed to run viable businesses which provide healthy employment.

Even more invisible are the services which are generally provided reliably, which we are dependent on, but which would cause big problems if it were not available.

Groceries, electricity, petrol, safety, national deference, water.

Also invisible are the millions of people living in bad conditions around the world because the country lacks the expertise to provide them with basic needs like water, food, sanitation, ability to get around.

Yet more invisible is the damage we are doing to the climate, which will require colossal expertise of the right sort and in the right place to fix.

The software solution

The solution is to shift society – and our working culture – to one which values expertise more. We can't force culture to change, of course – but if we could encourage the use of software designed to support expert work, that would be a step in that direction.

A great deal of software which experts use ends up constraining expert work. It hasn't been designed specifically to constrain expert work, but since it is not designed specifically to support expert work, that is what results.

This solution starts with the idea of building software in a different way.

So much software which is used for the working world is built around a mindset of improving automation – so the computer can do more, and the people are reduced to entering information into online forms and reading what the computer tells them. You could say the person is working for the computer not the other way around.

There are good reasons why software is built in this way. Many people genuinely believe that this is the way the world is heading, with more automation. The people who specify software say what functions they want the software to have, and the people who build it, will build those functions and try to get them all working properly with no bugs.

We suggest starting with a different approach – instead of thinking about functionality, we start by thinking about what kind of software would best help the expert working with it to develop and use their understanding of the domain and build out from there.

Nothing in these ideas is original, although they have not been put together in this way before.

Some software developers have been thinking about 'domain driven design' – designing software around a domain – and ways to continuously improve software until the 'users' are happy with it, for over a decade now.

Meanwhile the way in which people use and develop their expertise is well understood – by understanding a situation and seeing how the situation changes based on the decisions that are made.

How would you like to be valued for your expertise?

So we come again to our opening question – how would you like to be valued for your expertise?

Your expertise is your understanding of how something works. You know how to manage a factory making a certain kind of microchip. You can manage the maintenance program for a city district water supply. You can run a school or a classroom. You can manage a police station.

This is not just for senior roles. Managing a classroom, managing a police service (which could be just for a village), managing a company or a department in a company, all requires people who understand how that system works. Managing the maintenance of a car, or the maintenance of anything. Providing assistance to someone over the telephone (otherwise known as working in a call centre).

Companies and governments need expertise at all levels, you want to develop expertise, and you want to work for an organisation which values you for your expertise.

So we offer an approach which can help companies build software tools to support their expert workers – and in doing so improve the visibility of the work the expert workers do within the organisation, and the perceived value of the work.

Another angle on this is to say – perhaps with a touch of cynicism – as a society sometimes we don't value experts, but we do value software. So now we are talking about software driven expertise. Perhaps people will value this sort of expertise more.

What expertise means

We're not talking about a particular sort of expertise, because we think all expertise is like this, but you may disagree - it revolves around understanding cause and effect.

The definition of expertise we're using here is of someone who understands how a system works, they understand what is going on, they understand if what is happening is satisfactory, they understand what levers they can pull to bring operations in a satisfactory range, or improve the performance in other ways.

Of course no-one has absolute knowledge of these things, in the way that the best classroom teacher in the world could find herself in a situation she doesn't know how to deal with. And similarly a novice might start with some understanding. So we are talking about understanding in relative terms.

Continuing further, an expert will have a range of comfort zones. There could be a 100% comfort zone where everything is going well and nothing needs to be done. If activities are moving out of the comfort zone, the expert will have a good idea whether anything ought to be done, and if so what.

If we get completely out of the comfort zone, when an expert has never seen the conditions before and has no idea what kind of levers will work, at that point the expertise is useless.

The expert builds up an understanding of how the situation works from an internal and external sense.

From an internal sense, the expert can see what is usually leading to what and why. We can call this 'scenarios', telling a story about what is going on.

From an external sense, the expert can see what usually leads to what, without necessarily understanding why. We can call this 'patterns'.

[Note: this understanding of expertise is taken from Emery Roe's excellent book 'Making the Most of Mess'.]

And before any of this can happen, the expert needs to understand what is actually going on. We can all this 'situation awareness'.

An expert will see a situation in a different light to a non-expert when presented with the same information – in the way that an experienced classroom teacher will make a completely different assessment of the state of a classroom of children than someone who is not a teacher.

An expert will have better idea of what kind of situation awareness is necessary. If the expert is running a website, she might think that direct response the website is generating is far more important than anything the web traffic statistics say.

Coming into a digital realm, an expert will understand what of the available data is telling something useful, how to best work with the data, which data shouldn't be trusted, or which should be ignored.

What expert life should be like

Expert work should be fun, or at least, not miserable.

There are many reasons why society is better today than it was 100 years ago. But expert work is one reason society could have been better 100 years ago than it is today.

Imagine - a village or small town where everybody had a trade, everybody had a shop, fixing bicycles, making shoes, pianos, growing food, helping mothers give birth.

There was space for so much valued expertise - because if you wanted your shoes fixed, and you couldn't pay for new ones, you had to go to the person who knew how to fix shoes, and pay them to do it.

But it is not too hard to see that living in a society where people can work for themselves, they don't have a boss, they don't have a risk of the company getting bought, and not much risk of the company going under, because the town probably has business for a fairly finite number of shoe repairers, and you are one of those.

People have expertise which the society values directly (by being willing to pay for it),

These are perhaps dangerous comments, because this shouldn't come across as nostalgia for the past that is not the intention.

But perhaps we can take some lessons from the past into today. How can we set up a society for expert work, as fulfilling as that one would have been?

All expertise is different

The cause and effect is pretty different everywhere, which means that expertise isn't very transferrable.

A doctor – someone with a good understanding of the cause and effect of a body – has reasonably transferrable expertise, if bodies in one part of the world are pretty similar to another. But the health services, and how to get on working in them, won't be.

Someone who has done well as a police officer or teacher in one part of the world will find part of their abilities transferrable, but part of their abilities definitely not.

Goals

Goals are a critical factor in this picture. An expert will usually be assigned goals to achieve, which could be keeping the service provided reliably, or achieving improvements in performance. Let's say the goals are set by someone else (like a boss).

As well as showing which direction to go in, the goal will drive learning. We don't learn by being just told things, like a safety announcement on an aeroplane.

We learn by having a direction we want to move in, and by figuring out how to get there, if we are getting there, what we do which helps us to get there, what we do which doesn't take us there.

The organisation and you – the expert

Experts and organisations should fit well together.

It can be hard to earn a living as an individual expert, unless you are in the lucky situation of having special expertise and a queue of clients who need it.

Imagine if you have skills to repair a certain sort of vehicle. Customers typically pay you by the hour. So you only have an income if you have a stream of customers with problems with that sort of vehicle, which take time to fix. Problems you can fix in a few seconds (by advising a certain part which needs replacing), and problems you can't fix at all (if it would take more hours than the customer is willing to pay for) bring you no income.

You may be more likely to have a steady income stream if you work for a larger company, which has a larger pool of customers than you can achieve as an individual, which can spend money on advertising if necessary, and which can take a risk on your salary, paying you a certain amount monthly to keep you on the company's books, even if business is slow.

There are many expert-centric organisations, who treat the experts well, and are rewarded by good work, leading to happy customers.

But there are many organisations who do not treat their experts very well, in all fields, from teachers to engineers.

The people who run organisations respond to short term imperatives – and that rarely involves the experts, who are employed full time and not interested in going anywhere. They see that the expert needs the organisation, and its salary, more than the organisation needs them – and many people also have the same expertise.

Experts may end up trying to contrive the situation to one where the organisation is forced to value them – for example by being restrictive about sharing information which only they know, or trying to keep the organisation using old software which only they know how to fix. It isn't desirable for either side, just a matter of survival.

To add to this, experts like to work in a certain way, and that is not difficult to understand. They usually want to work to a regular schedule, and have the materials they need readily available, so they can get to work on the value adding stuff, understanding a complex situation and doing what is necessary to keep it on the desired path.

You don't want to be bothered, told that you are about to lose their jobs because the organisation is losing money, be told that you are not reaching the organisation's targets even though you know you are doing a good job.

You don't want to have two bosses providing conflicting direction. You don't want to have to go out and find customers or new employment. And especially you don't want to have substandard tools.

The best answer should be that the organisations and the experts value each other and can get on with doing the work as effectively as possible.

Perhaps if we move to a world where experts have software designed to help them use and develop their expertise, then the relationship between the organisation and its experts will be much healthier.

Call centre example

Say your company runs call centres – or has people working on the telephone responding to customer enquiries and problems.

As the call centre manager, you could try to automate everything, so that the people on the telephone have a standard list of questions to ask, which they enter into software, and the software advises on the next step. You can run "analytics" on your call centre staff to improve their productivity, looking to see how many problems are solved with one phone call, and how long the calls are.

Or you could rely on human expertise, and try to do everything you can to support your staff, so they can provide the best possible support to the person calling. Can you give them better computer tools so they can understand what is going on with the caller's account?

Can you monitor the most common reasons for calling, and put information about how to fix them online, and make sure your staff know how to deal with it? Can you provide rapid 'escalation' to provide access to someone who really knows how to deal with the problem if the first person can't?

How we develop expertise – the seafarer model

The way we develop expertise is pretty well understood – we have a goal, we have a situation, we do something, we see how that changes the situation.

The military calls it OODA – Orient, Observe, Do and Act. They could add 'learn' to the end of this but that part of it is probably well understood.

Imagine an ancient seafarer crossing the oceans. There are critical elements of situation awareness – the working atmosphere onboard, the condition of the vessel, the wind and other elements of weather, how much food is left, a basic idea of the location from the stars or perhaps from what you can see.

In some circumstances, the seafarer can improve the working atmosphere, fix problems with the vessel, make the most of available wind, adjust food rations, and see if that is making progress towards the goal of getting to the destination. Over decades, the seafarer will learn which measures lead to the best results.

In your mind, you have a model of the situation. You have a picture of the critical elements of the situation. You know what you need to be advised about, to keep your model updated.

You cannot learn how to do this through formal training, simply because just about every real life situation is different. You can go to a seafarer training school and learn how to navigate by the stars and fix a sail (or perhaps you could in ancient times), but then you discover your boat, your crew, your voyage, have unique properties about it, which you can only figure out when you are there. Formal training can only work if someone has done the path ahead of you, and it is pretty similar to the path you are about to go on. Much of life is always changing, so formal training usually needs someone who has done the path ahead of you quite recently.

Let’s introduce “Software for Domain Experts”

We use the name "Software for Domain Experts" or SFDE to describe software which is designed with the priority of making it easy for experts to use. This doesn't mean software designed around the database, or software which has seen tweaks on the front end to make it 'user friendly'.

With a bit of practise, we think it might be possible to identify or rate "Software for Domain Experts", software

designed where the expert takes priority, over software designed for someone else to take priority.

This is similar to how we can quickly recognise with a piece of writing, if it was primarily written with the reader in mind, or primarily written for some other purpose, for example, putting a certain point of view across, or persuading you to buy something.

We are all pretty good at judging other people in this way too. Is this person genuinely friendly, or friendly because they have an ulterior motive, for example making connections to someone we have connections with, or our money, or social circle? Most of us become experts at this.

Life is not as black and white as this of course. Consider that a good politician can simultaneously put her view across and make you think she is on your side. Good writing can put the reader first and also sell something. Although the logic does work if you think about it in degrees, a politician's first priority is showing that she is on your side, and her second priority is convincing you of the conclusion she has reached about what is the right policy from your point of view.

Coming back to software

Let’s get back to software now. How can software best help this expert?

A large part of it, perhaps nearly all of it, is in the situation awareness piece of it.

A lot of what is currently going on is held in people's heads. The software can't help there. But it can help with any information which is held in a digital format, and there's an increasing amount of it these days.

We don't mean simply presenting information either. The software can do some analytics on it - but all directed by the expert, not a programmer who doesn't have much understanding of what the expert actually needs to do.

When it comes to modelling, the software can be used to build all kinds of models, or representations of reality, which can help the expert to understand their world and what they need.

Although the most important model will be the one in the expert's head.

The software can help co-ordinate work with other people, which is also part of situation awareness.

Software for Domain Experts is not designed to impress

Much commercial software these days is designed to impress, to create a wow factor when the salesperson is showing it at the exhibition stand, or in the sales meeting.
This is a distraction for "Software for Domain Experts" (SFDE). To be SFDE, the most important criteria is that it works and does not frustrate.

There are many things in life which work well in a range of situations and do not frustrate, and they are perhaps not what you would expect to be, if you had never seen them. A railway line, a chocolate bar, a shift dress, a white shirt.

It is very difficult to create such a thing using your imagination only. Probably you need a lot of trial and error in the real world, and a lot of iterative development. SFDE is the same.

You probably use software every day which just works and does not frustrate you – which has been through a great deal of iterative development. Perhaps your e-mail software, or your web browser, or your favourite website.

All of this software has been created with enormous software development budgets and a mass audience in mind. Companies developing SFDE must manage to create great software without a mass audience.

"Software for Domain Experts" will probably include a 'push' component, where the software 'pushes' useful and relevant information to the expert.

Providing the right information at the right time looks very simple – but of course creating software which does this is extremely complex task, because it involves the software 'understanding' what the expert is actually doing or needing. "Pushing" the wrong information is an unwelcome distraction.

An expert might feel that it is better not to be 'pushed' anything at all than be pushed unhelpful information – like having a (human) assistant who is not very good at figuring out what you need to know.

Why this doesn't happen

Why isn't the current software development industry usually very good at creating SFDE?
Let's start by looking at how working software is usually developed – by getting a list of 'requirements' and building it up function by function.

Programmers seem to like 'requirements', as an agreed list of what is to be built, and how much money is to be paid for it, so the programmers can get comfortable doing their expert work.

Yet many people say that the process rarely works as it should – clients don't know what they want, programmers don't know how long it will take to build it, and there can be 3 or more intermediaries – the CIO of the organisation which employs experts, the sales manager of the software company, the analyst and project manager of the software company. There's very little direct contact between the experts and the programmers.

The software industry loves to describe you, the expert, as a 'user' – and like always, the choice of language gives a great deal away. Drug addicts are called users. The automobile industry does not refer to car drivers as 'users', probably usually they are 'customers'.

Besides, probably no-one is thinking of software as a tool to help people develop and use their expertise. They are

thinking of software as a tool to do some complicated calculation, or information movement, or co-ordination task.

A great deal of software for the working world is basically built around a relational database with logic and calculations around it – and the programmers spend more time trying to constrain the highly complex real world into a relational database, than they do making software which reflects the complexities of the real world.

A second reason why the software industry is not good at making SFDE is to look at the organisation's perspective. Big companies are not in the mood for helping experts use their expertise better. They would prefer things to be automated, and not dependent on any specific individuals.

A third obstacle is the current obsession with analytics, automation, big data and internet of things. It creates the illusion that it is possible to run society without expertise at all. Plenty of people are falling for this.

It isn't possible to make much of an argument about what technology might one day be able to do because we don't know. But you draw your own conclusion. Can you imagine technology running a police station?

Can you see where too much automation is counterproductive, like when your phone line is fixed and you are talking to endless automated systems (even if they are disguised by having people in call centres reading information on screens at you)?

A fourth obstacle is that developing custom software around the needs of one expert can be very expensive. But the cost of doing this is dropping.

The belief that automation and analytics is the future

In 2016, many people believe that the future involves more and more computer automation (ie computers doing more tasks), and analytics will become more powerful, and this is the way technology – and society – is going.

Planes will fly themselves, we will use robo-lawyers and robo-doctors, news articles will be automatically written from news reports, our accounts systems will file themselves, payments will automatically be made to suppliers once certain criteria are met, robots will care for the elderly, fridges will order their own replacement produce which will be farmed and delivered by robots.

Perhaps this idea has particular appeal to software people, if they imagine that it will mean that while other people's jobs will disappear, the need for good software programmers and data scientists will only increase.

Software for Domain Experts sits on the other side of the fence – the side which says, most of the tasks which could be automated have been automated, and we still need plenty of human experts. Perhaps we need experts even more than 20 years ago. And these experts need tools.

The argument for whether there will be more automation has many complex components to it, including an assessment of what technology can do today, what may happen if technology advancement continues its current rate (ie what will happen if current technology trends continue, something we are not sure about but can accept as a possibility).

We could also look at why the idea of continual technology advancement may be a mirage anyway (is your office computer better at handling e-mail and making documents than it was 15 years ago? How about your e-mail spam filter?) Also acknowledging that while computers may be able to put words together to make a sentence, the role of a journalist is a fair bit more than that.

And the current obsession with 'systematising' the working world, reducing individuals to people who have to achieve multiple specific 'key performance indicators' and are not given much autonomy, can also reduce work to jobs which are capable of being done by computer. But that is probably not a world we want to live in, because real life is usually more complex than that. That is something we see when companies give all their staff 'key performance indicator' targets attached to bonuses, and then discover that staff

did manage to meet their targets at the expense of everything else.

Or if you really believe that automation is the future, perhaps think carefully about what is happening next time your home phone line or internet connection stops working and you have to go through a hellish experience with the (probably) over-automated but incapable system your phone company has put in place, where you have to speak to multiple people in India to get them to call your local phone engineer to have a look at your local phone exchange, where someone has perhaps put the wrong plug in the wrong socket.

You are my hero

This is what we want you to do, if you believe our story.

If you are an expert, you can see what good software can do, and try to get it built – enabling you to do wonderful work which you find massively satisfying, and which you are very good at, and where it is clear that you're good at it – so much that if you take a break from work, you can get back into it reasonably easily.

If you are a software developer, you can see how what you are building could perhaps do more to help experts, and you can help push your company projects in that direction.

If you run a small software company, or considering setting one up, you can see a new business opportunity, serving a specialist market, with knowledge of how experts in that market think, combined with your software expertise, and knowledge of relevant software platform tools, taking advantage of new technology products freely available, and cutting edge academic research.

If you are an investor, you can see opportunities to fund great small businesses to do something which is valuable, new, and takes advantage of recent technology development and research.

Note 1: This is not an academic book

This is not an academic book – in the sense that most of academic study usually ends up looking at a small subject in great detail.

Here we are doing the opposite – looking at a huge subject, the world of expertise, in less detail than it is usually looked at – in order to try to find a pathway to serving expertise differently with the help of software.

We will touch on many subjects which academics look at in great detail – including software modelling, software engineering, psychology, learning science and management.

This book raises many questions which it does not answer – although many of the answers can be found in academia or publishing from technical disciplines, and perhaps the book raises questions other people may want to find answers to, or which we can write about in future books.

Please don't blame the authors for not answering all the questions being raised, is the point of this here. The intention is to provide a path, not to answer every question.

Note 2: where we're coming from

This idea has we think many original components to it, which make it justify your time and effort understanding it, but it did not come from thin air.

We draw a great deal from our own experiences.

Dimitris comes from a Greek shipping family and has been developing software for the maritime industry for 20 years. He also runs a software company for call centres, aiming to give staff the best possible information to serve the person they are calling, and which follows (or created) many of the ideas in this book.

Karl runs a magazine called Digital Energy Journal, about digital technology in the upstream oil and gas industry, and

also a magazine called Tanker Operator, about deep sea tanker shipping (not software). He also runs the publishing and events business around these magazines.

Many of the ideas were developed through the Software for Domain Experts conferences we ran together at The Cube, Athens, Greece, in November 2015 and April 2016 - thanks to the participants in both of those conferences for helping us develop to the ideas.

Thanks to Jenny Pantelis and Alex Menounos for the original ideas and discussion in May 2015 which started all of this off.

Many of the ideas in software, learning, artificial intelligence, psychology and butcher shops originated with US expert Roger Schank, who previously wrote a book together with Dimitris “The Future of Decision Making” (2010).

Emery Roe's "Making the Most of Mess" book is a great study of how experts work and understand a system, and how they are best supported. It is based initially on a study of how electricity control room operators manage electricity supply.

Eric Evans book and software development approach "Domain Driven Design" (2003) is a very close partner to Software Driven Expertise. One big difference is that Domain Driven Design is a book for software developers, this book is targeted more for experts.

Badass: Making Users Awesome by Kathy Sierra has great ideas about how expertise actually works.

Seth Godin has a wonderful worldview about expertise, and many of the ideas expressed in his wonderful blog and books are expressed here.

Part 2 – How to build software which drives expertise – the expert perspective

Introduction – software which makes experts more valued

We began this book talking about how you, as an expert, could be more valued for your expertise – and our solution is having software which is more expert-centric.

This logic does not all flow smoothly – just having software does not change the way an organisation feels about its experts.

But an organisation which does value its experts better will be willing to pay for more expert-centric software.

If the expert-centric software is available, companies will be more likely to use it.

If experts are aware of software which can do more to help them, and have a better idea about how to get it implemented, they are more likely to push for it.

Perhaps our principal end goal is more accurately expressed as an expert-centric culture – since this will lead both to experts feeling more valued, and more use of expert-centric software. But culture can't be changed directly.

We see our goal as similar to the goals of the Economist magazine, which was founded in 1843 to "take part in a severe contest between intelligence, which presses forward, and an unworthy, timid ignorance obstructing our progress".

The battle between expertise and ignorance is perhaps the same today as it was in 1843. Perhaps it was the same in ancient Greece. Perhaps it is the same in every society. Perhaps it can never be ultimately resolved – but the side of 'expertise' needs all the help it can get.

We can say that software which doesn't directly support people's expertise could be unwittingly supporting the other side of "unworthy timid ignorance", and unfortunately, that's arguably most software which experts use in 2016.

Any software which is built as a collection of functionality is probably not expert centric.

The expert mindset

The expert mindset is pretty well understood, but not well understood in the realm of software development.

Experts continually learn – not usually from formal training alone, but from their experiences.

They have goals they want to achieve (or are assigned to achieve), they want to understand the current situation to see if they are getting where they want to be, and they learn how the situation works. This could apply to anything – a policeman, marketing manager, school head teacher, school teacher, and soldier.

Learning has a circular relationship with work, in that the more they learn and master their domain, the better they can work, and the more they learn the more motivated they are to continue.

If they can be left alone to get on with something (ie have autonomy), that can also improve the motivation. Working with others can be fun, but not having one or more 'boss' who keeps asking you to change direction.

It also helps if the goal is something linked to something they want to achieve at a deep level, such as improving safety, improving education, providing a reliable electricity supply, not just an abstract 'key performance indicator'. Then they have a sense of purpose.

Autonomy, mastery purpose, the key motivators, which all lead to increased expertise. Thanks to Daniel Pink for that theory.

We understand systems

Humans are very good at systems, although we don't usually see things in that way.

Your house is a system, and you have a mental model about what you have in the fridge, what maintenance work might need to be done, what state your household relationships are in. Your body is a system and you learn what leads to what which leads to you feeling stressed, upset or ill. If you run a small business, that's a system too.

Expert work is the same – we understand the system of the environment we work in – how it works, what leads to what, what certain indicators mean. Our brains are very sophisticated this way.

The capability of people to understand a part of life as a system is colossal - we have so many systems in organisations - economy, government, government services, employment, and politics. All have cause and effect and ways to get better at it.

The police understand their part of the world, and the policing it demands, as a system - including the sorts of phone calls and what type of response they need - judgement of whether a call needs no response (same guy often calls to say he is about to jump out of a window) and all possible response.

Actors (the theatre type) also figure out a system. They work out a role which they can play well, and which finds them regular work, they stick to it. They gradually improve the voices, body shape, mannerisms and behaviour that goes with that role, so they can get into a new role quickly. They may pretend they learn every role from scratch, but even we in the audience, seeing the actor playing a similar role in different places, know that's not true.

But our working environment structures, and our software, don't do anything for 'system man' or 'system woman' at all. They are structured as though the best way to get the most out of us is to get us to work faster, make our tasks simpler, and be better at checking what we're doing.

Software for Domain Experts aims to take a path out of this. It is built around an understanding that everyone has a 'domain', which they have expertise in, and they understand

what is happening in it, and how it works – and they need certain information to maintain that understanding. Software built with this aim in mind can provide it.

The environment for expert work

The ideal environment for expert work is not difficult to explain. We want to able to focus our minds on the part of the work which demands deep thinking - and have everything else as simple and undemanding as possible.

Here's one way to explain it.

My mother used to complain about her father who was an electrician and carpenter, saying that in order to be persuaded to do carpentry work on the family home, he would expect someone else to find all the tools and materials he needed, and clear up afterwards.

This illustrates the expert mindset, in the sense that experts enjoy the hard part of the work where they need to think hard – it is everything else that they find frustrating. (If they are working for free, such as a carpenter working on his own home at the insistence of his wife and daughter, perhaps the less interesting part of the work is even more frustrating).

Whilst being paid to do work, the frustrating part of the work is just as frustrating, but most people have learned that complaining at work does not go down well. But this does not mean that frustrating people at work, such as by giving them frustrating computer software, is a good idea.

To take another example - a legal expert is probably very happy discussing some complicated legal point with another lawyer, using her specially trained and developed legal brain to find the right answers, using a language that only lawyers (and perhaps only lawyers in that particular domain) would understand. Perhaps she'd happily do this on a Saturday night in the pub.

But for a legal expert to explain a legal issue to a lay person takes far more effort and is not usually so much fun.

Don’t expect the same lawyer to be so keen to explain complex legal issues to a layperson on a Saturday night.

Sometimes when people reach superhero status at work, they will also not tolerate spending five minutes doing anything frustrating – such as when a top film director says he will only work if he has food delivered exactly the way he likes it to the set, and won't tolerate any interruptions.

The caveman mindset

Another useful way to look at expert work is to recognise that we are all, in an evolutionary sense, the same humans who were around 10,000 years ago. The factors which engage our brains the most strongly are the same as the factors which engaged our brains 10,000 years ago.

We all understand the caveman brain. But perhaps we don’t design our working systems and software with the caveman brain in mind.

As an example, our 2016 society routinely gives experts masses of written information in e-mails and documents and expects us to absorb it all perfectly and quickly. This is something we are no more capable of doing than we are of remembering the airline safety information which is ‘fed’ to us at the beginning of every flight.

Expert’s working mode

An additional factor to the expert's model is the working modes.

If everything is working fine, perhaps the expert does nothing at all, and keeps an eye on things. If someone is going a little wrong, perhaps the expert will still do nothing.

If something is obviously wrong, perhaps the expert knows exactly what to do, because he’s been that place before, like a head teacher who has a method to deal with troublesome pupils which generally works.

If the situation goes further outside the comfort zone to a place where the expert has never been, the expert is probably still more likely to come up with the right approach, based on her understanding of the situation.

If the situation goes further out of the comfort zone, the expert and her expertise provides no value at all – an example being the US financial experts in the 2008 financial crash, as Emery Roe documents in detail in his book 'Making the Most of Mess'.

The expert's mental model

All the time, the expert is maintaining and continuously improving a mental model of what is going on. This model is very different to a non-expert's.

Continuing the school example – when visiting a primary school as a parent, you might notice let's say the condition of the buildings, whether the teachers and children look happy or not.

Being a head teacher of that primary school, your mental model might involve the next government school inspection, issues with staff, the school maintenance program, serious complaints which have been made by parents, balancing the budget, although the condition of buildings and whether teachers and children look happy could also be a factor.

As a parent, you also have an in-depth mental model of the condition of your family and household – what is working well and what isn't – which will be very different to what any casual observer would see. A musician has a very different mental model of a piece of music she is currently performing than anyone in the audience.

Putting the mental model together takes the full extent of our experience with that thing. A head teacher with 20 years' experience will have a far richer mental model than a head teacher with 5 years' experience.

The idea of the mental model transfers roughly to software construction – because the ideal software would be designed around the same mental model which the expert uses. This is almost impossible to do completely, since the mental model is continually evolving and software takes time to develop.

But the mental model will have simple elements to it which probably all experts in that domain will follow.

Experts use expert language

Experts develop a language for talking about expert work.

People who know about something need a way to talk about what to someone else who also knows about that something, without all the intermediate steps in between of explaining stuff. One doctor passing a patient onto another doctor at the end of a shift, or two engineers talking about a project. Two lawyers can update themselves on a court case, two postmen can talk about what needs to be done and what they have done, teachers can talk about where they are with a class.

For a non-specialist it is easy to dismiss this language as jargon, and say why don't people get better at explaining themselves. They don't want to or need to, is the short answer.

Experts can also use language as a way to keep non-experts out of the conversation. It is pretty frustrating when you want to discuss a complicate point with another expert and you're expected to talk at a level a novice can understand. It uses much more brain power and doesn't really take you where you want to go. There's some compensation if you're helping someone to learn, which is satisfying to an expert, but often you're not even doing that, you're just trying to explain something really difficult to someone who doesn't understand or particularly want to.

Language can be used as a way to judge someone else's understanding. Someone else can learn the jargon, but an expert can see whether they use the terms in a way which illustrates they understand the models behind the jargon.

By language we don't just mean this word means that, like people talking in a foreign language. Because to understand a language, and work with it, you need to understand the concepts behind it.
Beneath the language is the model which experts use in their heads to understand the situation.

For example geology has an enormously rich language, which geologists understand. One geologist can explain a rock formation to other geologists using this language. It includes terms for the geological time eras, the different ways geological formations came into being, different sorts of rocks, how they formed, how they moved.

As a non-geologist you could buy a geological dictionary, or look up terms online, so you know them all. That would help, but only to a certain level. Because a real geologist has an innate understanding about how rocks came to look like they do, or what we can work out about rock from the information available to us, and that takes years of study followed by intense work with actual rocks. That is probably deeply satisfying work, and valued by society (in the oil and gas industry). That is to say, understanding geological language will only take you so far in geology.

For software to be useful to domain experts, the people building the software need to understand and respect the language, and build tools which fit with the language.

The software needs to contribute to this model, not try to get them to build a new model completely.

Software developers are not meant to be defining a new way to work with the domain from scratch. There are very few life situations where this might be - it would need to be a field where new software completely changes the way that domain experts work, and they don't come across too often. (One of the few examples is - again - geology, where subsurface interpreters use computer systems to understand the subsurface in a fairly completely new way. Although probably not a completely new way).

As a software developer, if your software tool doesn't fit the language which the domain experts use, that's an indication you're on the wrong track.

Work sectors this applies to

The concepts here are designed to support any expert anywhere - defining an expert as someone who needs to understand the cause and effect of a system - so long as some of the situation awareness that person needs can arrive as data. (It won't work if your situation awareness comes from other people's minds, or from looking at something directly).

It is probably most useful for people who are managing something, with responsibility for keeping something reliable, or achieving a certain goal.

Many of us are responsible for doing something reliably - people providing services (oil, electricity, water, good maintenance, happy school classes, groceries, security, hospital management, transportation, shipping, telephone customer service).

People who run companies, or have roles in organisations, are often given goals to achieve (improve sales, achieve targets).

To work out where this would add value, you can start with the question - how would it help this expert, if she had the most useful possible information which the organisation might be able to make available, displayed in a way which is easy to work with, so she has the best possible situation awareness?

Police

How could the police and government security professionals be better served by software?

A hint to the answer can be seen reading news reports about investigations into why events happening. They commonly say

that a key piece of information had been in the police systems, but wasn't acted on by the right person at the right time. Another government passed on a message that a person had been to Syria and may have been radicalised. Someone crossed a border and the border force were alerted too late that they should have stopped them.

In a fantasy world of security expertise supported by software in the best possible way, a computer might be able to help sort out all the vast amounts of information in security databases, so everyone has the right information at the right time.
Police forces typically have databases which they use to store information about crimes, vehicles and people. But these databases might not be integrated. So there is no automated way to let a police officer know if they are reporting a crime involving a vehicle, and that vehicle is recorded in a separate vehicle database as stolen. The vehicle registration number needs to be looked up separately in both databases to find this out.

A police software commonly used in North America is CompStat (short for computer statistics). This has a Wikipedia page which describes it in detail.

The computer tries to work out the optimum place for police to be at any time based on analytics, and works out how well individual officers are doing. It is credited with both reducing crime and killing morale in the police force. And its effect on reducing crime is disputed, because crime could have been reducing for other reasons.

What if we take a SFDE approach and begin by asking, what is the ideal software for a police officer, which enables them to observe and orient?

Perhaps it would be imagined as a mobile phone tool, where the officer can login at any time and see the most relevant information, taking into consideration where she is, what else is going on, what non-emergency tasks she has been assigned.

It could inform the officer, if there something relevant happening nearby, if a person of interest about to drive

past, having been automatically tagged by license plate recognition or other tools. It could show the officer an updated list of non-urgent calls to make during the shift. Perhaps the officer has been assigned to be at a specific place at a specific time (for example, outside a certain Underground Station around pub closing hours.

One way to use technology for security is to build models or normal behaviour, and compare it with behaviour of people under interest. Individuals generate enormous amounts of data today in their day to day life, much of it available to the police.

The behaviour of someone planning a terrorist attack could look very different to anyone else's behaviour. With some behaviour modelling, together with expert work, it could be possible to have far more effective security without being intrusive.

Cybersecurity

Cybersecurity is commonly thought to be about catching the bad guys – but is perhaps far more about situation awareness. When you read reports about security breaches, they are often not particularly sophisticated, such as the story of when infected USB drives are dropped around a staff carpark and someone puts them in a PC on the corporate network.

Security systems can be a big pain and obstruction to expert work – which suggests many cybersecurity experts do not have the situation awareness they should have about the impact of their systems. A good security system both achieves security and doesn't disrupt normal life.

Many corporate security efforts put too much emphasis on the firewalls and antivirus (the wall to keep out the bad guys), but very little effort on understanding what is actually happening on your network.

Situation awareness tools can help you visualise your corporate cybersecurity very differently. Can you see how much memory all the PCs on your network are using (because a

compromised computer might use far more of its memory)? Can you see the traffic on your network by IP address / location, and spot any trends there, for example you have much more traffic from China than you usually do?

Complex travel planning

This software approach could be useful for people who plan complex travel arrangements. We are talking here about travel which is more than just booking individual hotel rooms and plane tickets.

For example – if a flight is cancelled, working out the most effective way to book an alternative path to the destination at the last minute, taking into consideration accommodation and different travel options, can be very time consuming to do using modern software tools, which are designed mainly for buying individual plane tickets and hotel rooms.

The level of complexity can quickly reach the point where a human being with travel expertise can achieve far more than a computer can, particularly if she already knows much of the information (like how easy and costly it is to get a taxi from one airport to another).

So perhaps travel computer companies should stop trying to make more powerful computer systems, and instead build better tools to serve human travel agents - perhaps like the sort which were around before the internet came along.

Fintech

Fintech – "financial technology" – is a buzzword in 2016. The world of technology for finance is of course enormous – although some commentators say that the term 'Fintech' is actually used for a very narrow range of applications, such as providing 'robo' advice.

There is some "Fintech" which aims to bring process to financial tasks previously handled by experts. For example one hot Fintech company at the time of writing aims to make loans to small companies guaranteed against invoices the

company has already issued, so the company can be paid instantly, rather than wait for the standard 30+ days, on payment of a fee.

The company has developed a complex database driven system designed to try to drain all the risk for the company making the loans.

One of your writers tried this service out for his company, and perhaps unsurprisingly, it did not work. The client company did not meet all the database-driven criteria and there was no scope for human judgement in the system to over-rule the rules of the database.

Another area of Fintech is automated decision systems – such as where a bank is able to approve loans automatically if the client meets pre-designed criteria. But this is not the computer 'deciding' – this is a human deciding, and setting rules for the computer to follow.

For the purposes of this book, let's look at what kind of software can drive financial expertise – by helping them observe and orient.

Financial experts can include individuals and company financial managers looking after their own cashflows, investors managing portfolios, banks making decisions about loans, banks managing their own cashflows, and everybody managing their risks.

There are many tools on the market to analyse financial data. But they are usually presented as 'financial analytics', which is not quite the same as situation awareness. Let the analytics system tell you what you need. Like all analytics systems which promise useful answers, it probably won't.

To take a personal example from managing a small business finances, predicting how much cash will be in the bank from day to day is extremely hard. We can see the volume of invoice issued, and the invoices already due with our accounts software.

It doesn't tell us what our experience tells us – some clients pay on time, some always late. Payments come later at the end of the month when our customers are paying their own salaries, and payments come later during summer and winter breaks, when our customers receive their own payments later, or have a lower volume of sales. Our costs – mainly salaries – do not vary much during the year, so that makes a certain point of summer (when spring revenues dry up and autumn revenues have yet to start flowing in), particularly hard for cash.

All of this could be put in a computer system, if it was built just for our company – and the computer could also analyse our specific accounts patterns and come up with something better. No-one is building software for that.

The control system operator

In the control systems world, an operator needs to make very important decisions about what to do based on information the software provides. He could be running a ship, an offshore oil platform, a nuclear plant, an electricity grid.

The room for error is small and the potential damage for error can be huge.

Yet often control systems are pretty poorly put together, for example with alarms de-activated and no-one even knows they are de-activated, or with so many alarms that no-one can respond to them all. They are often put together function by function, with no centralised planning around what the operator actually needs at all.

In an ideal control room, the software would be completely transparent, providing the operator with exactly the situation awareness she needs at any time, with the operator understanding fully how the information provided to her has been generated.

This takes enormous research and development work, and is tricky when the systems have been provided by many different companies. There are legal issues involved, such as when a company wants to be able to demonstrate that its systems are

capable of warning about an emergency situation by sounding an alarm, but not considering that the operator may have 100 other alarms sounding at the same time.

Part 3 – how to build software which drives expertise – the software developer's perspective

Introduction

How should software companies, software developers, company CIOs, experts themselves, and anyone else interested in building this sort of software, start with making this sort of software?

Continuing with the military model of 'observe, orient, do and act', which we introduced at the beginning of this book, perhaps a good start is to ask, what situation awareness does this expert need? What situation do they need to observe, and how does the information about that situation need to be presented so she can make her decision? Then ask, what can software do, to provide that situation awareness?

The software does not provide situation awareness by itself. Situation awareness is ultimately in the expert's mind, and is developed through a variety of information sources and mental processing. Not all of the information sources can be provided electronically, particularly if they involve assessing what is going on in another person's mind.

Ideally the information would be presented in a way which fits around the expert's mental model – and only tell the expert something she needs to know to keep her mental model updated.

Changing the technology culture

Looking at this question from a different direction, the ideas in this book go against the dominant technology culture of 2016, and that will make it hard for them to gain traction.

Usual themes covered in technology magazines are: this computer can do far more than computers have done before;

this person is really smart and ambitious, and all the investors are all over her; this is an interesting idea about how to make a business with computers doing more; here's how to spend your money, here's how to change the world.

Programmers can see the future, and it is more automation, more companies like Google, more data science and analytics, and less people getting in the way. They want to be on the side of history, not victims of it.

Software for Domain Experts is very different. The software company is either providing a service to experts, or one the same 'level' as the experts. The most important part is the expert work, not the software. The emphasis is what computers help people do, not what computers do.

There may not be much money around, since experts (other than software experts) are not usually paid very much, they don't have much money to spend on software. Investors are not usually interested, because this is a business model around serving individual clients, not one size fits all unicorn building.

But changing the world we can do - through better ways to develop and use expertise with the help of technology - not technology itself.

Changing the technology culture is probably the hardest part of this.

Like a butcher shop – a small service company model

The business model we envisage for providing the software is a small team (within a big company) or small company, comprising both software experts and domain experts, with the team or company specialising in one domain – such as police, security or education.

This is a business model like a butcher shop on your high street, where you (as an expert customer) can talk to the owner if you want to, and he knows your name. None of this

business of being passed around multiple people in ‘customer support’ and setting up ‘tickets’.

As a customer, you might buy your meat from a chain supermarket, but if you go to a butcher shop, perhaps you like to go to a shop which is run by the owner.

It could be the same with the Software for Domain Experts business model. The software could be provided by large software companies, but there are also opportunities for small software companies run by their owners, where the customer can speak directly to the owner.

In the software industry of 2016, large software companies are typically looking for large markets – either very large individual customers, or where they can sell exactly the same software hundreds or thousands of times. The Software for Domain Experts market will not be like this, because every expert's needs are different.

That doesn't mean that these ideas are not relevant to large software companies. It is possible for big companies to serve specialist markets taking advantage of their scale – like a coffee shop chain or supermarket chain which offers a service specially adapted to a small community while simultaneously taking advantage of its large supply chain or other development capabilities.

This business model will also take advantage of the full range of platform and ‘low code’ tools, which make it possible for a small team to build customised software.

Doesn’t software change people’s jobs?

One argument is that software itself changes people’s jobs. The role of a supply chain manager, a geologist, a publisher, a librarian, a banker, is so much different in 2016 than it was 20 years ago, and so are many other jobs. It is not the role of software developers to design software around the way people worked in the past.

The counter argument to this is that many jobs have in fact not changed too much. If you mastered teaching, law,

journalism, ship operations, policy development, heavy industry, policing or politics in 1900, probably the basic skills are still relevant in 2016.

To continue – if we are saying that the world revolves mainly around expertise, then we need to consider where that expertise comes from. Experts learn their understanding from the people who came before them. They adapt to take advantage of the changes in technology and circumstance – but the core of their understanding is passed from expert to expert.

The role of a supply chain manager, or a geologist, are extreme examples of jobs transformed by technology in 2016. Today's supply chain manager uses enormous amounts of analytics tools to make sure the company (which could be a retailer, or industrial plant) has the optimum items in its inventory at any time, and anything it needs isn't too far away. Yet doing this role, even with the help of analytics, takes an enormous amount of skill – which can only be learned from other supply chain managers.

Today's petroleum geologist will be understanding the subsurface of the earth with enormous amounts of computing power. But the most important parts of the role – understanding how to transform data from sensors (seismic recording in this case) into an understanding of how the subsurface formed, still need the same technical understanding as a petroleum geologist from decades ago.

Software decentralisation

It is possible that large software companies will never be interested in Software for Domain Experts markets – because they are principally interested in selling their consultancy and existing product portfolio.

In that case, the main pathway forward for Software for Domain Experts might be looking for more software decentralisation.

There is a big push in some sectors of the software industry for this – encouraging a future which is made up of many

small ‘apps’ controlled by different people, communicating with each other using data standards and protocols, rather than a future dominated by large software companies.

For software engineers

For software engineers, SFDE will probably need more of an ‘agile’ and iterative approach.

The working paradigm of being given clear ‘requirements’ of what needs to be built, and a clear timescale to build it in, probably won’t work here. The requirements are not clear, nor is the budget or time scale.

The general mindset of building software which ‘does stuff’ will probably need to change – the priority here is not what the computers can do, but what the computers enable experts to do.

Thinking of the person who works with the software as a ‘user’ is probably something which needs to change. Language is important, and the word ‘user’ suggests that the person is passively ‘using’ the thing which you create. Actually, the person is doing expert work and your software should be helping them.

The word ‘user-friendly’ probably also needs to leave the lexicon. It suggests that you are making some effort to make your creation palatable to the ‘user’ as an after-thought to the software construction, rather than building the software around the person who has to work with it at the outset.

There might be far less coding and far more designing, using low code tools.

Designing around situation awareness

When designing software one starting point can be to consider what situation awareness the person in that role needs and whether software can help to provide it.

For example there may be daily checks the expert makes, such as a company manager getting an update from sales staff, or a finance manager checking the company bank account, an airport border control manager checking expected passenger numbers and planned staffing, an oil production manager checking the past day's oil production.

There are other aspects of situation awareness linked to alerts, where someone needs to be warned about something bad going on – for example an alarm in industrial equipment, police emergency response, a business responding to an urgent customer complaint.

Domain Driven Design - Building software around a model

We see Software for Domain Experts as something of a sister idea to Domain Driven Design, the idea of building software around a central model of how the domain works, described in great depth in a 2003 book by Eric Evans.

In a speech in Brussels in January 2016, Eric Evans said that the ideas in Domain Driven Design could be more relevant now than they were in 2003, because there are new technology tools to make it easier to build.

Domain Driven Design can be seen as an alternative to database driven design, the idea of building software around a relational database.

A great deal of software for the working world in 2016 could be described as 'database driven design' – including most software for any kind of transaction or records management.

The problem with database driven design is that there isn't much in the real world which fits neatly into databases – apart from aspects of the real world which are constrained into a database type format to begin with, such as a list of people who have bought tickets for a flight, a list of library books or financial transactions.

For other aspects of the real world, such as what a company finance manager or city water authority maintenance manager

need to be aware of, are probably too complex to fit easily into a relational database.

So trying to build database-centric software for real world data makes for very complex software, with complex logic to put the real world into the database, and get results from the database which help someone in the real world.

Programmers put most of their effort into trying to make their database work with the data they have, rather than making the software work in the real world.

The complexity means that even programmers don't understand it well. The software company management and investors don't understand it and struggle to communicate with their programmers. The experts struggle to work with it.

The idea behind "Domain Driven Design" is that the core of the software is a model of how the real world works, or how the experts usually do their work. You can include data repositories as part of the software, but they are not at the core of the software.

The term 'model' means a representation of reality. It can't be defined in more detail than that – but, as Eric Evans says in his book, you usually know when you come up with the right model.

Making a software model is a little like making music. Music is and always has been at the core of many societies. It expresses how people in that society think, and it helps people in that society to understand their society more. We don't know how it does it - it is a very abstract art, and developed through a great deal of trial and error on the part of the people who create it. But when society feels that the music reflects how they feel in a way they haven't seen before, the music has enormous power.

Building the model requires both domain and software development expertise. It needs to be a simple as possible but not too simple. It covers the core tasks which someone needs to do. For example a geologist might want to see all the literature and papers available to the company about a certain field.

The model sits right in the middle of the software, with everything else – the data infrastructure layer, the applications layer, and the interface layer, built around it.

You have to keep the domain model in the centre of the software, all the way through the software development process. The longer the development process, and the more people involved, the harder this gets.

This is a little like starting a business around a model such as making the world a better place by improving education. But a few months later, the challenge of covering cashflows and responding to customer demands gets so overwhelming, the project team lose sight of the original 'model' or reason the business was set up. That means that you may have a business, but it isn't doing the great task which you set it up to do.

Transparency – understanding how it works

A benefit of 'domain driven' software is that should be much easier for everybody to understand – whether they are working with the software or working to develop the software.

This is because the software is built around a central model which everybody understands.

We can call this 'transparency' – we all understand what is going on.

To illustrate what 'non-transparent' software feels like, perhaps you remember a time when you were trying to submit an online form, and the software would not accept what you were trying to enter, but it was not clear why the software would not accept it.

Perhaps it was a company accounts payable system and you were trying to submit an invoice so you could get paid. For some reason, accounts payable software is often exceptionally badly designed.

The information you were trying to enter came from the real world and was correct, but there was some logic in the software you did not know about which was telling the computer not to accept it.

This is a deeply frustrating experience. Perhaps you went on to try to 'game' the software, working out what the software would actually accept. You may have succeeded in submitting the form that way, but the effort had nothing to do with your main work and what you were trying to understand.

The same thing happens to a less obvious degree when the computer presents information to you which you do not understand, like a flashing light on a car dashboard, which could mean anything from 'stop immediately your car is in danger of exploding' to 'we recommend a service visit in the next 6 months'.

"Understanding" a computer system is a relative term – for example, you can 'understand' a computer spellchecker without understanding the statistical processes behind it.

If you are a pilot of an aeroplane and the computer is making suggestions to you, you probably need more understanding of where those suggestions came from, than you need of how your spellchecker suggestions are generated.

Ontologies

Ontologies – or structured data sets – can be important, because many experts – and many people – see the real world in this way. But the software needs to be designed around the ontology that the real world person uses – you can't expect an expert to change how they work to fit the software.

An ontology could be explained by thinking about how your mother in law organises her kitchen implements, or how your father in law organises his garage tools (apologies for the gender stereotypes).

It makes complete sense to your parents in law, but when you try to visit, and try to help, or just find a tool or a kitchen implement, the ontology makes no sense to you at all.

That doesn't mean they use the wrong ontology, it just means you don't understand it.

Similarly a teacher may have a structured system in her mind for where the class is with a certain learning program, which pupils are the most troublesome, what to do at a certain stage in a lesson, probably all of these.

A police station chief may, in his mind, sort the staff into their various ranks and levels of experience. He categorises crimes and makes sure the serious ones get the attention they warrant. He might also have categorisation systems for physical assets, investigations under way, local police priorities, covering future staffing needs, local individuals which need special attention and more.

If you make crime reporting software which lists the most recent crimes by date, that is not helping him much with his situational awareness, if he lists crimes in his mental model by seriousness as well as date.

Low code

"Low code" technology – computer systems which write code themselves – have seem a great deal of technology development over the past few years.

The basic idea is that the designer creates a model of how the software should operate, and all of the code is created automatically.

This is a great benefit to 'Software for Domain Experts', because it means that the software developers can spend more time building and improving models, and less time creating code.

The code created automatically is far more reliable than hand-written code, and it can be automatically updated every time the model changes.

It can become viable to build software customised for one individual expert, with no more complexity than they need to do the task they specifically need to do. This is a much better alternative to software developed iteratively over decades, with more and more functions added to it in response to every customer request.

Perhaps building software using 'low code' can become so easy that it can be done by the experts themselves, with no need to use software developers at all.

Analytics and artificial intelligence

Analytics systems and artificial intelligence can make a great contribution to helping an expert maintain situation awareness. But the common challenge with these tools is making sure the analytics is assisting the expert, rather than a data scientist who creates the analytics believing that analytics can make the decision.

Most experts have more "data" than they know how to do with, and 'artificial intelligence' type tools can reduce it into something which humans can work with and make judgements with. These tools can cluster data and spot trends which a human cannot see.

There's no way to make a cast iron argument that analytics will never replace humans, since it involves predicting how fast technology will improve beyond what is possible today.

Here's one story which illustrates what is possible with analytics alone, and how analytics might be able to help experts.

A few years ago a major search engine made announcements that it could monitor the spread of flu viruses by watching what people were searching for, and where they were located. This upset the health professionals whose job it is to do this sort of thing, but without the benefit of the data. But

as it turned out, the search engine was actually unable to track the spread of viruses – and a spike in search terms was correlating with winter. So the search engine was predicting winter.

However if the search engine makes its data about searching for flu available to health policy professionals, perhaps they can do something useful with it.

"Artificial intelligence" is a misleading term, here. It has no clear definition. It seems to be usually taken to mean, computers doing something which people previously did, with a goal to gradually remove 'human intelligence' from the process. So people building 'artificial intelligence' are not motivated to think much about the human expert.

If 'Artificial intelligence' means tools which continually learn about the data and which data is most useful to experts, by looking at what is changing and what data experts usually look for, that could be very helpful indeed.

The internet of things

The internet of things is a buzz phrase of 2016. But when looked at through the lense of an expert, or someone who has to actually make business decisions based on the data generated by sensors, it looks very different.

For example, the shipping industry has had 'internet of things' discussions for decades now, with talk about fitting sensors over engines and other shipping machinery, allowing condition to be monitored.

Sending temperature and vibration data from a piece of maritime machinery to an onshore office is a relatively easy technical problem to solve. But actually getting value from this data is far more difficult.

It sounds promising, considering the enormous amounts of money shipping companies spend on fuel, and the environmental hazards of having a badly configured engine, if it can lead to higher emissions of soot from uncombusted fuel, and perhaps regulatory penalties.

But shipping companies don't usually have in-house experts who can work out how to improve an engine configuration from a vibration sensor.

Perhaps the engine manufacturer has staff who do know how to make value from this data. But they still need to know a great deal more about the engine in order to make a recommendation of a specific adjustment which will improve operations. And do they know exactly how the engine is configured and where the sensor is – or do they just have a stream of vibration data?

Looked at in this way, 'internet of things' starts to feel like a sales pitch from sensor, computer network and communications manufacturers, who want the shipping company to buy stuff without being able to contribute to helping them to use it.

Software for Domain Experts suggests an alternate pathway. If you start by understanding what a specific expert needs to know, then you can work out how to build it, and what sensors you will require.

And whilst you may not have expertise in your company about sensors, you probably will have expertise in what is most important to be aware of, and how this can be gathered, and what sensors will help.

Coming back to the shipping example, a shipping expert might determine that the best way to manage fuel consumption over a voyage is to come up with a voyage plan showing how much progress the vessel should be making at every day, so it arrives at its destination port at the best time, taking into consideration the conflicting objectives of reducing fuel consumption (so reducing speed), but not taking too long for the voyage and increasing the costs to the cargo owner.

Once this plan is in place, the shipping expert could decide what sensors are necessary to check progress against the voyage plan. In this example, perhaps no sensors are necessary at all – just asking the ship crew to send a 'noon day report' of where the vessel is at noon every day.

Part 4 – looking at the obstacles

Let's look at all of the obstacles which are preventing Software for Domain Experts software being implemented.

The biggest is the status quo culture in the software industry – which is geared towards more automation, trying to get computers to do more tasks which were previously done by experts, taking human decision making off the table.

The software industry is dominated by large software companies, who are looking to sell products which they have already made, because they make most profit on the 1,000th sale of a piece of software, and a big loss on the first few sales. They are not particularly interested in moving in new directions, and they are not usually interested in customised products.

The most profitable software products, which get the most promotion, are usually database driven, because the largest, most profitable software packages have been database driven for the past few decades.

Then we have the status quo in large organisations, which is also leaning towards trying to manage with less and less experts – if it is possible for computer software to do the work of an expert, that sounds very attractive.

We have a culture in society in general which is does not particularly value experts much, with the exception of software experts, with their ability to produce more robotic and analytical tools.

We have a culture in software development which is geared towards building software to a set of 'requirements', and usually building it around a database.

In 2016, we have anti-expert movements gaining ground politically – including 'team Brexit' in the UK, where one of the leaders said that no-one listens to experts any more, and Donald Trump in the US, who appears happy to try to twist the truth if it makes him more likely to win. The truth, which can be otherwise known as situation awareness

of what is actually going on, is a fairly core component of expert work anywhere.

The idea of automating society seems to be gaining ground. People don't mind a society geared around rules, so long as they are not too expensive or inconvenient. It saves the headache of having to think.

This automated society becomes very unequal. At the top, you have the people clever enough to design the rules, create the software, or persuade the public that they are worth voting for. At all other levels, you have people whose jobs become increasingly to serve the software, or to serve the system, and who will be paid less and less.

There will always be lots of jobs for lawyers because the system of rules becomes so complex only a professional lawyer can navigate it. The small elite have big egos, so they would rather try to win any fight using lawyers than come to a compromise.

The legal system gets so complicated that people win or lose any legal battle based on how much they can afford to pay on lawyers, not any real world consideration such as who is right. This means that anyone with money can basically do whatever they want.

If you do want to make the world better, Facebook will give you free tools to set up a 'group'. But Facebook knows that real power in the world is always held by the person who controls the platform, not the person who can gather together a bunch of protesters, and the platform here is Facebook.

The brightest people in the world want to work at Google, because Google seems like the company of the moment. The future will be Google getting bigger and other companies getting smaller. Google has the best programmers and the best data scientists, and they will have all the power in future. Google staff are treated like kings and have the highest status.

The idea of analytics having all the answers, rather than turning to pesky human experts to figure it out, is

compelling to politicians, company managers, software people, and the public – everyone, really, apart from the experts themselves. This idea has an enormous amount of traction.

If anything else is happening in society, such as people being unemployed or living in poverty, it is very hard to find out about it, even if it is happening in front of your eyes (which are not open, because everyone is glued to their mobile phones whenever they walk around). People get their news from Facebook, which uses algorithms to pick news which people actually want to read (unlikely to be about levels of unemployment in their neighbourhood). News companies which aim to reveal the truth and help experts get drained of funding and close.

There is very little chance of solving problems based around 'inconvenient truths' such as climate change because this requires both a public which understands the truth, and a supply of experts to work out how to fix it, both of which are unavailable.

The elite of society becomes smaller and more powerful, and everyone else's work becomes more like serving a machine. Professionals are given "key performance indicator" targets to meet. These are demanding targets which require their entire mental focus leaving little room for anything else.

Companies are increasingly choosing staff based on their ability to hit targets. This usually means 25 to 45 year olds with no children who can focus 100 per cent on their work. It isn't prejudice because this is based on performance. Those people are basically slaves.

This is a great world for narcissists - people who want to get power for the sake of having power or who need it for self-validation. Because they will make a point of going to get power. Having power is a good survival technique because usually the powerful get to eat first and get killed last, just as it usually was in the Middle Ages.

Narcissists don't like experts much – they don't bother to understand anyone else's world, they don't like anyone else having control of what's going on – so they'd like experts

to be underpaid and under supported, although preferably bound to their jobs with a salary they depend on so they don’t make too much fuss.

The rest of us suddenly realise that the people who really wanted power have gone out to get it, and sewn it up so they have their hands on all the right levers, and there's no chance of anyone else getting any. The rest of us weren't interested in power itself but we're not interested in having empty bank accounts either. But that’s what we’ve got.

Software programmers could save us from this future, with a small nudge towards creating software geared more to serving experts rather than automating them out of the picture. But they probably won’t, because programmers are pretty comfortable as it is.

And besides, software which tries to automate expert work can get very complicated, and software programmers are the only people who understand it, so it keeps them in their jobs.

When people say ‘the geeks run the world’ what they mean is, software gets so complicated that only ‘geeks’, who can think through how a computer follows instructions, can actually understand the software. That’s fine if you are a geek of course.

There are a few people tasked with finding the best way to organise software for an organisation and seeing what works and what doesn’t overall. They have job titles like ‘chief information officer’ and ‘software architect’. But their background is usually IT, so they are more interested in traditional IT type roles, like keeping IT costs down and keeping IT infrastructure running reliably.

We are continuing a battle which the Economist magazine has been fighting since 1843, when it was founded to take part in a severe contest between intelligence and ignorance, as it states on the masthead of every issue. This is not a battle which can ever be won.

Part 5 - This is worth doing

Introduction

Yet it is still worth trying to build better software for experts, even if just because it is hard to find a project which is a more worthwhile use of our time and energy. While the battle can never be won, it is certainly worth fighting.

And Software for Domain Experts probably is on the side of history.

Consider the switch from mainframe computers to PCs. It looks obvious now, but probably didn't in the 1970s – it took people like Steve Jobs and Bill Gates to work out where the future was going and take everyone else onboard.

2016 enterprise software is analogous to a 1960s mainframe computer in that it is extremely complex, and can only be worked on by specialists. The computer was designed as a processing machine, not to help someone with a task.

1960s mainframes were pushed aside by 1970s PCs, which were modular, built up with standard components. Although the overall complexity is higher, the complexity which anyone needs to work with is much lower. Broken parts can be replaced by plugging in a new one. Yet the way we all work with a PC is completely customised – everybody's set-up looks different. The PC is designed principally to help people to do things – not to process data.

People who will pay for SFDE

There are answers to the question of who wants to pay for SFDE – although they can be hard to find.

One perhaps unexpected answer is the 'mobilisation' of the workforce.

Many companies are talking about helping connect employee's mobile phones into the corporate software infrastructure.

The logic is quite clear, if it enables employees to "work" more when they are not at a PC.

It isn't viable to build a mobile phone app and think about 'usability' as though it is the icing on the cake at the end of the software development process (as many desktop software companies seem to).

The usability needs to be built in – which is another way for saying, have a core model for the software which matches the mental model of the people who work in the domain.

Mobile phone software cannot be complex.

But also, there is a limit which can be achieved with mobile phone software – it can be OK for photographing a receipt and uploading it to a corporate expenses system, but not for troubleshooting an offshore oil installation.

Another example where companies might be willing to pay for SFDE type software is call centre staff.

Many companies run their call centres by trying to reduce staff to machines working for the computer. Staff have targets to achieve, and enter data on a computer system, which tells them what to say. In theory it sounds good, but in practise the result is like your last bad experience with a call centre was.

Instead, imagine a computer system designed to put the best information right in front of staff, and help them to learn so they can diagnose what is really going wrong much faster and come up with the right solution to solve it.

Can't we love experts?

Political and media dialogue isn't entirely anti-expert.

Many cowboy films had an underlying theme of the expert cowboy takes on the system, which doesn't appreciate him.

The UK has ended up with what looks like an expert-centric prime minister, Theresa May.

Some Americans are about to vote for Hillary Clinton, mainly on the basis of her expertise running senior levels of the US government.

Angela Merkel of Germany is well respected as a thoughtful expert, Germans make fun of the way she thinks carefully though the options, calling it to 'Merkeln'. It is usually used in a negative sense, meaning someone who thinks carefully through options, and not necessarily do anything if that seems like the right choice. That doesn't fit with a narcissistic view of the world which expects leaders capable of making snap decisions, but it does fit with an expert-centric view.

Use the same software for training

There are more side benefits to this sort of software, which justify the effort and investment needed.

One benefit is that the same software could be used for training.

When fed with real world data, but not used in the real world, the software becomes a simulator of what the real world is like.

Most training and e-learning software is not like this. Most e-learning software, like any formal training, takes a student through structured steps, put together by a trainer, who has been down the same world before.

Formal training will always have its place, but it cannot train a student how the cause and effect works in the real world, because the real world is always changing.

But SFDE can be used teach a student how to understand how a specific situation in their domain works.

Roger Schank, guru of many topics related to Software for Domain Experts (including artificial intelligence, human learning, psychology and computing), has suggested that in his ideal educational environment, instead of children being

taught ‘physics’ and ‘chemistry’, they would be able to pick a vocation, such as a firefighter, and spend a few weeks trying out software simulation tools which show them what it is actually like to be a firefighter.

Then they can decide if they like it, and stick with it, or don’t like it, and try something else. By the time they leave school, they have a very good idea what job they want to do, and they have the basic skills in it, which is a long way ahead of how children leave school today.

Software for Domain Experts tools could be used for that.

A pathway to rescuing the news industry

If you value your expert-centric news – which gives you carefully considered ideas of what is going on – and are worried that the business model for that seems to be declining, perhaps we have a short answer – everything in this book is promoting an expert-centric culture, which values journalists and expert analysis far more.

That doesn’t mean they can necessarily make a living. But it ought to be easier.

Also bear in mind news’ role providing situation awareness to experts – people do use news to assist them in their work, not just as general awareness of what is going on. If cultures change so people are willing to pay more for software which enriches expertise, perhaps they’ll pay more for news as well.

Helping India

Software for Domain Experts could make a big contribution in India - helping the country meet its two main challenges, as described by its Prime Minister Modi.

One challenge is helping find youth employment – the other is helping improve Indian manufacturing, so the country imports less from China. Software for Domain Experts sits between these two challenges – helping young people develop

the necessary skills and understanding to run manufacturing plants.

India needs a means for young professionals, to get quickly to a point where they can run complex factories and water systems.

India takes education very seriously and has great educators. But formal education reaches limits because (except with research) it can only teach someone what someone else has understood ahead of them.

The challenge of running Mumbai's water supply is something which can only be understood by running Mumbai's water supply and the specific challenges there.

India also has great software developers.

Contributing to social inequality

SFDE can help contribute to reducing social inequality in two ways – by helping anyone become an expert, and by giving better tools to the policymakers who make decisions about how our society is run, and the experts who decide which individuals get additional state support.

Currently state support is something of a blunt force, having to make the same support methods available to everyone. But if decision makers could get a higher resolution view of where government funds would be most effectively targeted, they could use the funds more effectively – providing help getting out of a bad position in life to the people who can do the most with it.

It can also enable governments to provide a more sophisticated safety net – making the government's scare resources available in the best way to prevent individuals falling into (for example) homelessness.

SFDE can be a great democratising force if the tools can be available to everyone who has an interest in becoming an

expert – understanding how a certain sector of society or business really works.

This can make it much easier for (for example) mothers to return to employment after a break to have children, or for people without university education to catch up with people who did.

Climate

Climate takes maximum expertise, maximum understanding. There are no simple ways to fix the climate. Anyone who promises any is wrong.

There is no global dictator who can tell everyone what to do, so we all have to figure out the right way country by country.

The people who run our countries are usually pretty constrained in what they can do. They may support action themselves, they may have support, but they'll generally be a large constituency of people who don't believe that the trouble to fix the climate is worth it, and they often have a point, if the demanded price is high (if they are on a low income for example) and they have what seem like bigger concerns.

The only solution is to move up the lever bit by bit – and gently explore where it can be moved – and understand how it can be moved. We've seen a great deal of lever moving. Constructing large wind and solar farms, environmental rules on buildings, to start with. But so far the lever hasn’t moved enough to make any discernible impact on emissions at all, not really.

What it needs most of all is better software and methods to understand what is working and what isn't.

And better models beneath the software. There currently are no experts in CO2 reduction with a mental toolkit which works, showing how to do it – because nobody has done it yet.

There are scientists, lobby groups for various technologies, companies making products, politicians making judgements. But not people actually making sure a certain bounded society constrains its emissions to a certain level. We'll surely need to have these experts in order to solve it.

We don't have metrics for how well it is going – except the dreaded counting CO2 emission itself, which is really hard.

A simpler proxy for emission would be to count fossil fuel use, where the emissions from the fossil fuel go into the atmosphere (not with CO2 being stored underground). That could let these experts monitor climate emissions in real time, and understand whether the policies they are introducing actually work. They can immediately see that they are not, like in some European countries, building lots of wind farms, which causes gas power stations to be switched off and less expensive coal power switched on, which makes more emissions, more than the wind farms avoid.

There should be plenty of scope for software here – and only software – and experts which the software supports – can solve it.

Conclusion

We've been talking about a world which revolves around expertise, not power - and software drives it.

We think that is a nicer world than the one we are heading towards and we hope you agree.

A world driven by expertise is much better and software can drive that world.

Ultimately you as an expert can be valued for your expertise – and have a far more satisfying life – contributing to big issues in the world and having autonomy over your working world.

We think building better software to help experts is a very valuable use of energy in 2016 – and hope you agree – but it needs more people to get involved to make the change happen.

Can there be any better project than getting this software built and implemented?

Software for Domain Experts is a company set up in London to encourage the development and use of better software to support expert work. We run two conferences a year in Athens and publish a blog, and publications like this one.

If you like the ideas in this book, please forward the e-copy of this book, or the link to it online, www.bit.ly/sfdebook16 to people you think might be interested.

Please sign up to our newsletter, from our website www.softwarefordomainexperts.com

Come along to our conferences, if they are somewhere you can get to

Or maybe organise Software for Domain Experts conferences and meetings of your own.

You can contact Karl Jeffery on jeffery@softwarefordomainexperts.com