allan's blog - Agile & Digital Business

I help companies and teams that create software.

Monday, September 12, 2016

Conclusion: Agile & ERP, part 5

The story so far…

1. Agile & ERP?
2. The Good News about Agile & ERP
3. ERP Culture v. Agile Culture
4. Th Bad News about Agile & ERP 5.

Where does this leave us?

For a modern software developer, who understand agile and is well versed in the current ways of working encountering an ERP system is like stepping back 20 or perhaps 30 years.

The mindset of in ERP is more akin to the software development mindset when I started working in IT. It is as if the last 20 years have passed them by - the technology is dated, the agile revolution hasn't happened, users should be grateful for what they are given and the vendor is all powerful.

The deal with ERP is: the ERP system does so very much, *why reinvent the wheel? Writing software is expensive (and fraught with failure) so why not use one we made earlier?*

It's the software reuse argument written BIG.

The downside is: the ERP is all encompassing, you need to use particular technologies and these technologies haven't really advanced in 20 years. The skills are in short supply. Most importantly: all the ways we have found to make software development more effective are absent.

Let us be very clear: there are massive benefits from using an off-the-shelf system. ERP systems are big, really big, and complex, general ledgers, multi-country/multi-currency payroll, etc. etc. Why would you want to rebuild this stuff?

But: there are also massive costs.

The ultimate question is: *are the benefits greater than the costs?*

Its worth repeating what Capers Jones says at this point:

> "As a rule of thumb, if COTS [Common Of The Shelf] packages require modification by more than 25% it may be less expensive in the long run to build the same kind of application." Applied Software Measurement, Jones, 2008

Unfortunately he doesn't elaborate on how the 25% is calculated, he probably means function points. He continues:

> "Package modification is a high-risk activity with a significant chance of failure and disaster. Not only is productivity normally low for modifying purchased packages, but if the vendors are reluctant to provide support, modification can easily be more expensive than customer development." Applied Software Measurement, Jones, 2008

When answering this question please remember licenses for ERP systems are far from cheap, and they reoccure. Plus, vendors may well charge extra fees to support parts of the system which have been changed or to ensure backward compatibility.

Once installed clients can be very, very, reluctant to update ERP system. This is hardly surprising once you realise that "configuring" a system like SAP may mean having developers change SAP code in the base system. If SAP release a new version they may have changed that code, you are lucky if you can get away with a merge from hell.

When you buy an ERP system you buy an existing legacy system. You then attempt to configure it to be your legacy system. (See my Peugeot post.)

Contributors

- Allan Kelly
- allankelly2015

My books

- Continuous Digital: the alternative to projects
- Xanpan
- Little Book of User Stories
- Business Patterns
- Changing Software Development
- Agile Reader

Newletter sign-up

Sign-up for my newsletter

I am at...

Upcoming events with Allan Kelly

Planning for Value
Private: 15 September at Oxford

Agile Cambridge
Software Acumen: 27-29 September at Cambridge

Agile Bristol
Agile Bristol: 12 September Evening at TBA

Agile Dice Game
TechCityCoffee: 25 July 2017

widget @ surfing-waves.com

Speaking events

- Upcoming events
- Events RSS feed

my Linked in profile

More

- OnTwitter: @allankelly.net
- My Homepages: allankelly.net

Twitter feed

This sounds like a bad deal but its a deal that makes sense for a lot of companies and has done for 20 or 30 years.

Now, here is my conclusion:

> 20 years ago buying a legacy ERP system and configuring it to be your own legacy may well have made financial sense.
>
> But in the last 20 years technology has advanced, not just the CPUs but the programming languages, the programming tools and even the programming processes.
>
> Buying a legacy ERP system today means buying into a legacy development process with legacy tools. You step back 20 years.
>
> In the last 20 years the tools and techniques have got a lot better. While buying an ERP system may look like a sensible decision you also shackle yourself to 20 year old tools and processes.
>
> The tools and processes have improved so much in the last 20 years that the equation has changed. Modern approaches (i.e. agile and test driven) using modern technology means the cost and time gap between a) developing your own system using modern techniques (including microservices, existing libraries, open source, etc.) and b) choosing legacy ERP system (reoccuring license fees plus configuration costs) is far smaller than it was.
>
> I conjecture that the gap is getting smaller as we speak.
>
> Since ERP systems last decades the 20 year old ERP system you are buying today might be expected to last another 20 years. Who knows how much better things will get in two decades.
>
> This also means there is a market opportunity for “modern ERP” systems which use modern technologies (Java, JavaScript, Micro-services, etc.) to support modern working practices. Microsoft, Oracle and SAP are all trying to move forward but the legacy is very profitable.

Finally, one might ask: *given all these problems, and the changes in technology why do companies buy these systems?*

Nobody ever asked my opinion on this so I can’t really say but let me float an idea.

Only companies of a certain size, say $1 billion revenue can afford these systems. When a company reaches that size they are expected to buy such a system - having an expensive system is a sign that your company has arrived.

These systems are driven by the finance departments, the CFO. As companies get big they need such systems to control the company. But also, the CFO of a $1bn company wouldn’t look serious if he didn’t have one - after all, all the other $1bn companies have an ERP system so why not yours?

If you are the CFO of a $1bn company without an ERP system and something goes wrong it will look irresponsible. Having an expensive ERP system allows the CFO to say “We followed best practice and bought Oracle”. It reduces the risk of the CFO losing their job.

And buying an ERP system allows the CFO to claim more resources for his department.

And it allows the CFO to say at his next job interview “I introduced an ERP system to a $1bn company.”

So the next time you get appointed CFO of a company which as just broken the $1bn revenue barrier and you find there is no ERP system in place what are you going to do?

Engineers might think: I’ll set up a small team and start growing our own ERP system using modern technology.

To the engineering mind that approach stands the best chance of working and is financially the best route.

But it is also high risk simply because nobody else does it.

Everyone else says: “I’ll call SAP, I’ll spend a lot of money with them, Accenture, Cap-Gemini and CSC. When it goes wrong I can blame them. After all, thats what everyone else does so doing anything different would be high risk.”

Posted by Allan Kelly at 10:42 am

Tweets by @allankellynet

allan kelly Retweeted
Anna Filina
@afilina
Pro tip: if your method is 225 lines long, then you should probably refactor your code ASAP. I frown at methods over 50 lines.
18h

allan kelly Retweeted
Multinewmedia
@Multinewmedia
Get the new book from Allan Kelly (@allankellynet) for just $5 on LeanPub for

Embed View on Twitter

Follow by Email

Email address... Submit

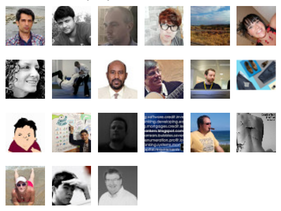
Subscribe To

Posts

Comments

Followers

Followers (39) Next



Follow

Alltop

Alltop ALL THE TOP STORIES

Popular Posts

Software has diseconomies of scale - not economies of scale

Some things can never be spoken

Programmers without TDD will be unemployable by 2022 (a prediction)

Why do devs hate Agile?

Dear boy, have you ever tried programming?

Banking systems stink - the pain of an botched HSBC release

Blog Archive

► 2017 (27)
▼ 2016 (43)
 ► December (3)
 ► November (2)
 ► October (3)
 ▼ September (5)
  Conclusion: Agile & ERP, part 5
  The Bad News: Agile & ERP, part 4

3 comments

Add a comment as Allan Kelly

Top comments

Claysnow Limited 10 months ago - Shared publicly
"having an expensive [ERP] system is a sign that your company has arrived" allan kelly http://buff.ly/2cTdrxT

· Reply

Dan Haywood 10 months ago - Shared publicly
Thankfully, there are one or two billion dollar companies that think differently.
http://isis.apache.org/powered-by.html#_powered-by_estatio

· Reply

Nicolas U 10 months ago - Shared publicly
That is very true. Nothing to add :)

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Culture problems: Agile & ERP part 3
The Good News - Agile & ERP, part 2
Agile & ERP? - part 1

- ► August (2)
- ► July (4)
- ► June (7)
- ► May (4)
- ► April (4)
- ► March (2)
- ► February (3)
- ► January (4)

- ► 2015 (42)
- ► 2014 (41)
- ► 2013 (43)
- ► 2012 (45)
- ► 2011 (53)
- ► 2010 (70)
- ► 2009 (90)
- ► 2008 (85)
- ► 2007 (79)
- ► 2006 (77)
- ► 2005 (62)

JavaCodeGeeks

Syndicated to Java Code Geeks

(c) Content Allan Kelly. Simple theme. Powered by Blogger.