



Productivity and legacy IT architectures...

By Cliff Moyce, Global Head of the Financial Services Practice, DataArt

When computing became ubiquitous in administrative environments in the late 1980s and early 1990s, it was welcomed as an opportunity to improve the efficiency and effectiveness of business processing. Manual or semi-manual business processing at that time was noted for its inefficient hand-offs, checking, and duplicated effort as well as storage problems. And yet 30 years later we look at extant ('legacy') IT systems architectures as representing the biggest barrier to productivity in some types of organisation. For example, large banks now spend nearly 50% of their operating budgets on IT – and yet it is IT configured in ways that would horrify any student of process design with the IMS. For example, multiple systems (sometimes meaning 20 or 30, not just two or three) doing the same thing; forced 'integration' between systems requiring software, middleware and hardware that should never have been required in the first place; inconsistencies between systems

Attempts to rationalise the architecture by building a single new system to replace multiple old systems often simply results in yet another system being added to the pile.

meaning reports have to take an aggregate of all outputs rather than relying a golden source, etc. Attempts to rationalise the architecture by building a single new system to replace multiple



old systems often simply results in yet another system being added to the pile. Support costs are high as people struggle to manage and resolve the complexity, risk and issues. What to do about these problems is a long running debate^{1,2,3}. One approach that is often espoused is to design and implement a new, more modern architecture using a radical clean slate/blueprint style approach⁴. While recognising the temptation to start again, this article asserts that big-bang approaches to legacy IT systems replacement can be naïve, expensive and fraught with risk. Instead, pragmatic approaches that can deliver improvements using what exists currently are preferred and recommended. As well as discussing technologies that can enable such approaches, this article considers the cultural and organisational implications of adopting these methods.

The debate

The debate on legacy systems in some organisations is intensifying as expectations for cost efficiency, flexibility, and usability increase. Legacy architectures are typically described in articles and presentations as unplanned; complex; poorly understood; slow and expensive to operate, support and enhance; old fashioned in their interfaces and reporting capabilities; hiding redundancy; difficult to monitor, control and recover; susceptible to security problems; and, hard to integrate with newer models and technologies such as cloud computing and mobile devices: 'Even minor changes to processes can involve rework in multiple IT systems that were originally designed as application silos'⁵. Getting old and new applications, systems and data sources to work seamlessly can be difficult, verging on impossible. This lack of agility means that

Explanations for problems associated with legacy architectures include excessive complexity arising from a post-hoc need to integrate systems that were originally designed to be autonomous.

legacy systems in their existing configuration can be barriers to improved customer service, satisfaction and retention. In regulated sectors, they can also be a barrier to achieving statutory compliance. Pressure to replace these systems can be intensified by new competitors who are able to deploy more modern technologies from day one.

Explanations for problems associated with legacy architectures include excessive complexity arising from a post-hoc need to integrate systems that were originally designed to be autonomous; poor knowledge of systems due to lack of documentation and loss of original development teams; individual applications growing 'like Topsy' as new functions and modules are bolted on to meet customer demand; use of technologies, models and paradigms that are now outdated; duplication arising from multiple systems doing the same thing, etc. 'Local initiatives' are sometimes argued to be partly to blame for the situation⁴ as business lines or functions commission their own system builds or buy package implementations, perhaps with little regard to integration and support issues. Many of these explanations for the problem could be summarised as 'customer requirements taking precedence over architectural integrity', but many people (especially the customers) would prefer that to the converse. Amusing analogies such as the possible negative consequences of living in an unplanned house that has been extended many times are sometimes used to encourage audiences to take a complete re-design approach to solving the problem⁴. By such an approach, it is argued that customer service can be improved and complexity, duplication and risk reduced. These are all highly laudable and valid aims, but how easy is it to design and

Trying to fix all of the problems at the same time is a logistical impossibility in anything but the smallest companies, and bears a high risk.

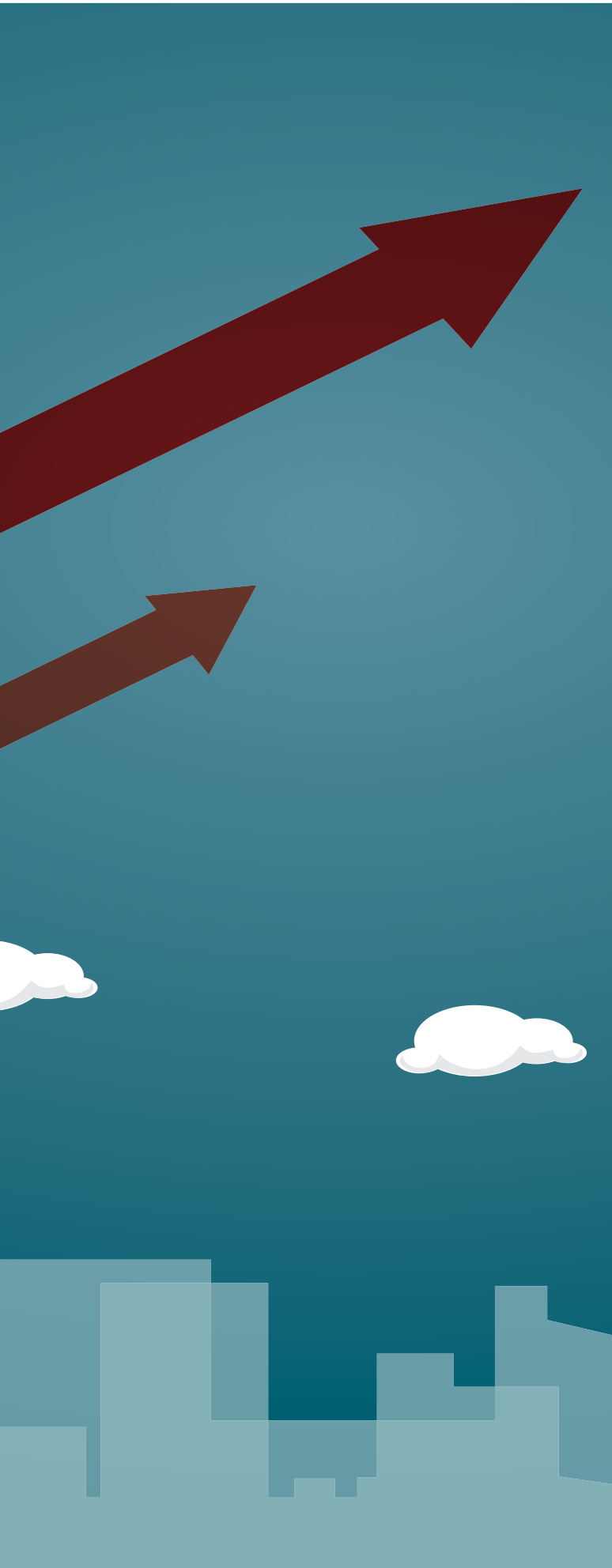
implement a new IT architecture in a large mature organisation with an extensive IT systems estate? For example, in a large bank with huge real-time transaction processing demands that has grown organically, and also grown by acquisition? Rather than the unplanned house analogy, a better analogy might be a ship at sea involved in a battle. Imagine if you were the captain of such a ship and someone came onto the bridge to suggest that everyone stop taking action to evade the enemy and instead draw up a new design for the ship that would make evasion easier once implemented. You might be forced to be uncharacteristically impolite for a moment, before getting back to the job at hand.

Redesign?

At some point, many large organisations have attempted the enterprise-wide re-design approach to resolving their legacy systems problems. Many such initiatives are abandoned when the scale of the challenge or the impossibility of delivering against a moving target become clear. Time has a nasty habit of refusing to stand still while you draw up your new blueprint. Re-designing an entire architecture is not a trivial undertaking, and building/buying and implementing replacement systems will take a long time. Long before a new architecture could ever be implemented the organisation will have launched new products and services; changed existing business processes; experienced changes to regulations; witnessed the birth of a disruptive technology; encountered new competitors; exited a particular business sector and entered others. All of these things conspire to make your redesign invalid before it is live. If you are lucky, you realise the futility of the approach before too much money has been spent. Furthermore, the sort of major projects required to achieve the transformation are the sorts of projects that run notoriously high failure rates: 'In just a twelve month period 49% of organisations had suffered a recent project failure'⁶; 'Only 40% of projects met schedule, budget and quality goals'⁷; '17% of large IT projects go so badly as to threaten the very existence of the company'⁸.

So if wholesale blueprinting and re-engineering is impractical, what can be done to solve the problems of legacy architectures? The first thing to say is that trying to fix all of the problems at the same time is a logistical impossibility in anything but the smallest companies, and bears a high risk. Many organisations would not have the resources to accommodate the large spike in project effort. Problems always need to be tackled in priority order as there is rarely a silver bullet for the whole job. Luckily there are some practical and cost effective approaches that can mitigate many of the problems with legacy systems while





obviating the need to replace any of the systems. Two of these approaches are service oriented architecture (SOA) and web services^{9,10,5}. Used in combination, they offer an effective solution to legacy systems problem.

SOA refers to an architectural pattern in which application components talk to each other via interfaces. Rather than replacing multiple legacy systems, it provides a messaging layer between components that allows them to co-operate at a level you would expect if everything had been designed at the same time and was running on much newer technologies. These components not only include applications and databases, but can also be the different layers of applications. For example, multiple presentation layers talk to SOA and SOA talks to multiple business logic layers – and thus an individual presentation layer that previously could not talk easily (if at all) to the business logic layer of another application can now do so.

Web services aims to deliver everything over web protocols so that every service can talk to every other service using various types of web communications (WSDL, XML, SOAP etc). Rather than relying on proprietary APIs to allow architectural components to communicate, SOA achieved through web services provides a truly open interoperable environment for co-operation between components.

The improvements that can be achieved in an existing legacy systems architecture using SOA though web services can be immense, and there is no need for major high risk replacement projects and significant re-engineering. Instead organisations can focus on improving cost efficiency by removing duplication and redundancy through a process of continuous improvement, knowing that their major operations and support issues have been addressed by SOA and web services. Another benefit is that the operations of the organisation can start to be viewed as a collection of components that can be configured quickly to provide new services even though the components were not built with the new service in mind. This is the principle of the 'composable enterprise'¹².

People????

Addressing the issue of legacy systems in a way that makes good sense is not just an IT issue, it is also a people issue. It requires people to resist their natural inclination to get rid of old things and build new things in the mistaken assumption that new is always better than old. It requires people to resist the temptation to launch 'big deal projects', for all of the reasons that people launch big deal projects – from genuine belief that they are required (or the only way), to it being a way of self-promotion, and everything in-between. It requires people to take a genuinely

Systems are business critical – not only to the organisation that own and operate them, but also critical to the businesses of their clients their clients.



objective view of the business case for change, while operating in a subjective environment. It requires people to prioritise customer service over the compulsion to tidy up internally. And, it requires the default method of change to be continuous improvement rather than step change projects – which can be counter intuitive in cultures where many employees have the words ‘project’ or ‘programme’ in their job titles. But this is all easier said than done when you are dealing with people in a real life organisation, where certain skills and behaviours have been valued highly for years. It is not an overnight job to get people to realise that it is those skills and behaviours that are contributing to their problems. Resistance to change should be expected. In fact, as long as resistance is overt it is a good thing because at least people are engaging and opening themselves up to discussion and the possibility of learning¹³. Getting to the point where legacy IT architecture issues can be handled in the best possible way will involve many of the common aspects of organisational change – education; developing new skills; adopting different mindsets; using multiple rather than single methodologies; and, basing the choice of method on the reality of the situation rather than on custom and practice. The popularity of agile methods means that continuous improvement using iterative rather than step-change approaches is in vogue again.

Conclusion

To summarise, resolving the problems of legacy enterprise IT system architectures can provide significant gains in productivity, efficiency, agility, and customer satisfaction. For these reasons the endeavour should be a high priority. However, there are many risks attached and this type of work needs to be approached in a way that is highly mindful of those risks. After all, the systems are business critical – not only to the organisation that owns and operates them, but also critical to the businesses of their clients their clients. Luckily we now have technical tools and approaches available to effect radical improvements without having to incur the expense, effort and risk of major replacement projects. But using these tools comes with a change of mindset and approach that may be counter-cultural in some organisations. It can mean a move away from step-change and ‘long-march’ projects, and a move towards continuous improvement. Education and engagement will be one of the keys to making it happen.

References

1. Souza, B de. (2015). *Enterprise architecture and the legacy conundrum*. CIO (13284045). Last retrieved 16 July 2015 from <http://www.cio.co.nz/article/563662/cio-upfront-enterprise-architecture-legacy-system-conundrum/>
2. Preimesberger, C. (2014). *Updating legacy IT systems while mitigating risks: 10 best practices*. Last retrieved 5th September, 2015 from <http://www.eweek.com/enterprise-apps/slideshows/updating-legacy-it-systems-while-mitigating-risks-10-best-practices.html>
3. Matei, C.M. (2012). *Modernization solution for legacy banking system: Using an open architecture*. *Informatica Economica*, 16, 2, 92-101.
4. Marchand, D.A. and Pepper, J. (2015). *Firms need a blueprint for building their IT systems*. *Harvard Business Review* (June 18, 2015). Last retrieved 22 July 2015 from <https://hbr.org/2015/06/firms-need-a-blueprint-for-building-their-it-systems>
5. Serrano, N., Hernantes, J., and Gallardo, G. (2014). *Service oriented architecture and legacy systems*. *IEEE Software*, 31, 5.
6. KPMG (2005). *Global IT project management survey*. Last retrieved 5th September, 2015 from <http://www.kpmg.com.au/Portals/0/irmpmq-g-global-it-pm-survey2005.pdf>
7. IBM (2008). *Making change work*. Last retrieved 5th September, 2015 from <http://www-935.ibm.com/services/us/gbs/bus/pdf/gb03100-usen-03-making-change-work.pdf>
8. McKinsey and Company in conjunction with the University of Oxford (2012). *Delivering large-scale IT projects on time, on budget, and on value*. Last retrieved 5th September 2015 from http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value
9. Cabrera, L.F., Kurt, C., and Box, D. (2004). *An introduction to the web services architecture and its specifications*. Last retrieved 30th June 2015 from [https://msdn.microsoft.com/en-us/library/ms996441\(d=printer\).aspx](https://msdn.microsoft.com/en-us/library/ms996441(d=printer).aspx)
10. Li, S.H., Huang, S.M. Yen, D.C., and Chang, C.C. (2007). *Migrating legacy information systems to web services architecture*. *Journal of Database Management*, Oct-Dec 2007, 18, 4, 1-25.
11. Mahmoud Q.H., (2005). *Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)*. <http://www.oracle.com/technetwork/articles/javase/soa-142870.html>.
12. Murray, J. (2013). *The composable enterprise*. Last retrieved 22nd July, 2015 from <http://www.adamalthus.com/blog/2013/04/04/the-composable-enterprise/>
13. Moyce, C.L. (2015). *Resistance is useful*. *Management Services*, 59, 2, 34-37.

About the author

Cliff Moyce is a Global Head of the Financial Services Practice, DataArt. He has taken his skills into financial and professional services, environmental protection and utilities. Cliff is a fellow of the IMS and hold an MSc in organisational psychology from Birkbeck, University of London. www.cliffmoyce.com or cliff@cliffmoyce.com