Regent's Working Papers 2019
Working Paper 1907: RWP1907

Enterprise Software: Buy or Build? Part 1: Making the decision

Michael Talalay
Management Consultant & Senior Lecturer, Regent's University London, Inner Circle, Regent's Park, London, NW1 4NS, UK
talalaym@regents.ac.uk

Abstract: This article uses two case studies to analyze the buy – build issue. Should organizations attempt to build their own software systems, or should they buy an existing package. The focus is on what are called Enterprise Resource Planning (ERP) systems, the complex, all-encompassing packages that handle all of an organization's commercial operations: financial, sales and marketing, manufacturing, human resources, logistics and so on. The two case studies are both drawn from my experiences as an IT consultant in the manufacturing and logistics sectors. The one company was a manufacturing conglomerate, the largest such in the UK, and the other was the world's leading supplier of ship management services. The former chose a package, whereas the latter opted for building its own system. The article looks at the reasons for the decisions and evaluates whether they were correct. A follow-on article looks at the implementation processes in these two different scenarios and tries to see what differences and similarities can be found.

Keywords: ERP; buy or build; enterprise software; package software; bespoke systems

Word count: 4,127.

1. Introduction

All organizations need IT systems to facilitate transaction processing and to enable fast and accurate information retrieval and analysis. Many companies buy what are known as Enterprise Resource Planning (ERP) systems from software providers such as Oracle, SAP or Microsoft to provide these functions.[1] Other organizations develop their own systems, possibly combining these with some "best-of-breed" packages. The waters are muddied by complexities such as cloud computing, software as a service (SaaS), and "building block" applications. But the fundamental question remains the same: buy or build?[2]

Both approaches have seen significant successes and catastrophic failures.[3] The objective of what follows is to shed some light on these alternatives on the basis of two case studies.

One case study involves a multi-national manufacturing conglomerate. The other involves a global ship management company. In each case, a successful decision was made with respect to ERP packages. One company bought; the other built. Though each decision was based on specific circumstances, the lessons to be learned are general in nature. They indicate when it is appropriate to buy a package and when it is advisable to build a bespoke system.

Though each decision was correct in principle and though the end result proved the decision correct in practice, neither project was guaranteed to be successful. The follow-on paper to this one compares the implementation of these two decisions and asks whether the factors that made each successful were different or identical or a combination of the two.

2. Background: The Choices Facing Acomarit

A few days after I started a new job in London, my boss told me to grab a flight to Glasgow. We had a prospective client to meet.

The client was Acomarit, one of the world's leading ship-management companies. By this time in the late 90s, Glasgow's maritime future was well in the past. Ship building on the Clyde was in terminal decline, and running away to sea was no longer the glamorous adventure of years gone by. But Glasgow was a major location for ship management; it was teeming with experienced ship's captains and engineers who understood the industry and could ably fill the necessary and crucial roles in the shore-based offices from which the world's merchant fleet was run.

My boss (a fellow Canadian by the name of Tim) had been invited to Glasgow a few days previously by one of Acomarit's two IT managers to advise on whether onward development of the company's financial systems was better done in PowerBuilder or Delphi (at the time two of the leading software-development tools). Tim declined to venture an opinion on which might be the better option. Instead, he pointed out to the CEO of Acomarit that his company had two separate IT departments, with two IT managers who weren't really talking to each other, and two incompatible IT systems. Tim added that the computer room was unlocked with the door wide open and was used to store paper and soft drinks.

Tim suggested to the CEO that perhaps the company needed to consider more fundamental issues before choosing a new development tool. So, Heathrow airport in the morning and the British Airways shuttle up to Glasgow. The CEO wanted a quick analysis on what the issues were that needed addressing. Tim wanted IT expertise; he was ex-Royal Navy and could talk boat, but he needed systems knowledge.

We interviewed various managers and staff and I wrote a short report setting out the problems.[4] The CEO liked this and asked us to do a more detailed investigation into the problems and put forward a set of solutions. I spent several weeks in Glasgow with various members of a small team that Tim and I put together to do an intensive analysis. Again, the CEO approved of our findings and he asked us to go ahead, to put our money where our mouths were, and to implement our recommendations.[5]

He also told us "not to do an Arthur Andersen" on him. He did not want us to flood him with youngsters who had just started and would learn at his expense. He wanted us to train his people and transfer our skills to them.

The key question was buy or build? Three basic options confronted us. The first was to buy a package, an integrated system that encompassed all, or at least most of, the functionality required. The second option was to opt for "best of breed": to buy a selection of separate packages, each of which was a leader in its particular area, and then to have them interface with each other and exchange information. The third option would be to build from scratch. The two last options could be combined – buy best of breed or a partial solution and build the remainder.

3. BTR and the Case for Buying an Integrated ERP Package

BTR (British Tyre and Rubber as it formerly was) was a global industrial conglomerate, which by 1997 had become the largest UK manufacturer outside the pharmaceutical sector with sales of £9.5 billion.[6] When I joined in the early eighties, as the twelfth member of the Management Information Systems (MIS) department, the company had just junked its centralized, mainframe computer system and was replacing it with smaller mini-computers (what today might be called servers) distributed to the operating companies themselves. We had found a hardware/operating system combination that provided both a single, integrated database and a powerful, easy-to-use natural enquiry language. The final decision was between two good software packages. Each was comprehensive in that it covered all the major functions of a company: financials commercial, inventory control, logistics and manufacturing. One, however, excelled in manufacturing; while the other has its strengths in the financial areas.[7]

BTR was a manufacturing company, primarily supplying other businesses rather than consumers. It made items such as vacuum-blasting equipment, aircraft deicers and landing gear, hydraulic hose assemblies, North Sea oil rigs, and printers' blankets. Hence, the obvious choice was the package that was stronger in manufacturing. Despite this, we opted for the other package with its financial capabilities. Why? Because BTR was managed as a decentralized organization, with a flat management structure, small business cells, operational autonomy, and very tight financial controls and reporting. It was this requirement to analyze and provide financial information to an extraordinarily tight deadline that was key to the package selection. Commercial benefits rather than technical features were the deciding factor in our choice.

The successful growth of BTR was down in large measure to the triumvirate of very capable senior executives. One of these was Don Tapley, the head of BTR Industries, which was the division into which Europe and the UK fell, including the MIS department. Our remit was to implement the new system into as many of the operating companies as wanted it and could financially justify it. The new system would be of substantial benefit to almost all our subsidiaries. Most had very little by the way of IT functionality (we're back in the early 80s). Accounting software and payroll running on a bureau were the norm. The benefits of introducing a comprehensive package, with a single, integrated database and real-time updating were clear. Moreover, the operating divisions were on the whole engaged in discrete manufacturing, which suited the software. The package was a natural fit. Nevertheless, there was often pressure from clients to modify the software to fit their "unique" requirements. Don Tapley issued a directive. He said that if the package did not fit the way that the company operated then the company should change the way it worked.

Initial thought suggests that this directive is illogical. A case of tail wagging dog. IT systems serve the company; they facilitate transaction processing rather than forcing processes into the straitjacket of generic systems. Initial thoughts, in this case, are misleading. Existing processes are likely to be a mishmash of out-of-date standard procedures amended in an ad hoc fashion over time and modified by informal working practices. This is not an ideal place to be. Existing processes are unlikely to be efficient, effective, or up to date. Any good package, on the other hand, will incorporate "best practice". On the whole, changing processes to fit the package is likely to be productive and innovative.[8]

A further argument in favor of not altering the package is cost. The issue is not just the up-front price but rather the total cost of ownership (TCO). Every time the software supplier issues an update, it will need to ensure that the company-specific version is also updated.[9] This running cost can be high: not just the cost of coding but also the extensive testing required. And, of fundamental importance, there are likely to be delays in getting this done, because modifying clients' specials is not likely to be among the top priorities of the software house.[10] There are significant costs in terms of both money and time.

A very different argument against making changes is that they are likely to be the wrong ones. Implementing a new system is more than a technical issue. Setting up and configuring the software, testing it, and then training users are key functions (with the latter two often neglected in an attempt to meet an impossible time constraint[11]), but they are relatively easy in comparison to the associated change management. Only after the system has been bedded in will it become clear what modifications if any may be necessary. It is probable that these are not the ones that were originally perceived to be required. Don Tapley was correct. Change the operational processes rather than the software system.

The case for preferring an integrated package to either best-of-breed or bespoke software is clear. Everything links seamlessly; best practice is built in; no bespoke interfaces are necessary; testing is pre-done; information retrieval and reporting work smoothly; costs are known.

4. Acomarit and the Rationale for Rejecting an ERP Package

So, for Acomarit, I recommended against an ERP package. Why? Given my very successful experience with integrated packages and given the clear and compelling reasons in their favor, why did I recommend that Acomarit not buy one?

There were three major reasons for this decision.

First, Acomarit marketed its services at least in part on the quality and distinctiveness of its IT. This was one of the company's competitive advantages. These IT systems did leave something to be desired. However, they were still better than most of their competitors, (the state of IT in the industry not being wonderful). If Acomarit were to buy a package, then it would become a follower. That marketing advantage would be lost.

Second, no appropriate package existed. There were packages to do part of what was required, particularly with respect to on-board planned maintenance, but there was no package specifically geared to managing ships.

Given Don Tapley's directive, however, the question needed to be asked as to whether a generic ERP package might be a viable option: would a package that did some of what was needed be better than building bespoke software? The problem came down to the specialized functionality within ship management. Thus, when we specified our requirements, the key modules included crew rostering, risk and safety (including ship and mariner certificates), and vessel management. Moreover, even conventional areas such as purchasing, accounting, and planned maintenance had specialized quirks, including ship-shore replication. These key functions made using a conventional ERP package problematic.[12]

Third, Acomarit did have development expertise and experience. There was a total of eleven IT personnel, and the financial and the operational systems had been developed in house. The skills and the knowledge were there to handle both the integration of best-of-breed packages and bespoke development. This was not just an unsubstantiated hope. As part of our detailed investigation, we performed very detailed analyses of the strengths and weaknesses of the IT team.[13]

Taken together, these three reasons argued strongly against an off-the-shelf ERP package and made in-house development feasible. Against all my principles and my background with integrated packages, I advised the client not to buy an ERP system.

5. The Best-of-Breed Option

The second option was to use best-of-breed, possibly in conjunction with in-house development. This was the current situation. Acomarit had developed financial and operational systems in house and interfaced the latter with ShipNet, the industry-leading planned maintenance system from the Norwegian software house of that name.

One strong possibility moving forward was to buy a financial package and then link it with the operational systems. Acomarit needed fairly sophisticated multi-currency functionality, because it had to be able to produce its reports in both dollars[14] and sterling and also because of its large number of foreign currency invoices and disbursements. A further complication stemmed from the fact that every vessel was a legally separate company with its own set of ledgers. Any package would need to include multiple ledgers and a consolidation function where individual ship Profit and Loss Accounts (P&Ls) could update the corporate P&L. These requirements restricted the options available but did not exclude this possibility.

However, there was one further requirement. Acomarit needed to report to the owners in their preferred currency and format. This provided a strong incentive to us to write our own financial module: there were distinct marketing advantages to be able to go to clients and say that they could structure the account presentation in virtually any way they wanted. When we finally designed the general ledger, we did so in such a way that clients could log into the system, access the accounts for a particular vessel, and then assign their codes and structure to our standard vessel general ledger. They could have immediate access to their vessel's accounts in their preferred currency and their specific chart of accounts structure. We viewed this as a marketing advantage well worth having. Close to twenty years later, the CEO of the cloud communications platform provider Twilio was quoted as saying, "One of the line of business shifts we're seeing is putting people in power who want to ***build great customer experience and use software to do it***."[15]

Similarly, we could have continued with the existing ShipNet planned maintenance system and structured the rest of the package around that. However, as this functionality is so central to the industry, we felt it was best to have our own system. Moreover, we judged that we could write a better one.[16] There were also two further downsides to keeping ShipNet. First, we would be using a package available to all our competitors – again where is the competitive advantage here? Second, we would need to confront the issue of writing interfaces.

As a result, the second option, best-of-breed, was also rejected. We embarked on the process of building our own ship-management ERP package from scratch. Of course, creating a new system was not going to be a trivial exercise. There were a number of key issues that needed to be addressed. These will be addressed in Part 2 of this paper – and compared with the implementation issues of a package – as successfully practiced at BTR.

6. Conclusions

Both the BTR decision to buy and the Acomarit decision to build were vindicated by events.

At BTR, the operating companies were uniformly keen to adopt our package. And on the whole they obeyed Don Tapley and did not demand core modifications. Our hardware suppliers were so impressed by our package and our approach to implementation that they began to bring external clients to us. We ended up spinning the MIS department out into a wholly-owned subsidiary, thereby turning a cost centre into a profit centre.

For Acomarit, there were three indicators of success. The first, and most straightforward, is that the system was developed and used to run the company. Second, Acomarit was acquired by its major competitor, V.Ships. The reason for the takeover – and the price paid – was in substantial measure due to the quality of the IT systems that were being built. Finally, the ShipSure ERP package for ship management has been successfully sold and implemented into other ship managers.[17]

There is no *a priori* answer to the question of whether to buy or to build. However, four broad factors seem to suggest which course of action is appropriate: marketing, capabilities, functionality, and responsiveness to change. On the first point, if a company's IT systems provide a competitive marketing advantage, then there is a good case for developing them in house rather than purchasing a package readily available to competitors. For Acomarit, this was fundamental; BTR, on the other hand, in no way needed unique systems for its competitive advantage. The second point is whether the company has the capability and the capacity to build its own systems. Acomarit did. BTR did not, and the time and expense to develop the expertise would have been prohibitive. (The alternative of commissioning a bespoke system from a third party is a recipe for disaster). The third factor is functionality. Put simply, if there is no package that meets a sufficient amount of the required functionality, then building a package or integrating best-of-breed solutions are the only remaining options. Finally, there is the issue of responsiveness to change. When alterations in circumstances require significant modification or enhancement of the software, the ability to respond quickly is essential. Surprisingly, this does not necessarily mean that in-house development has an advantage. Thus another client of mine with in-house systems had an IT department so overwhelmed with work that they could not address a fundamental problem in a mission-critical application. Instead, they hired expensive temporary staff to work around the software failure. It took external intervention (by me) to sort this issue. On the other hand, BTR had access to the source code of the package we bought. We were able therefore to respond quickly to any change in the operating environment.[18]

Circumstances will suggest which strategy is appropriate. However, as with all strategies, execution is everything. Part II of this article will look at how BTR and Acomarit successfully executed their respective strategies. And we will address the issue of the extent to which their successes rested on similar factors or on different ones.

References

[1] B. Perkins, "What is ERP? Key features of enterprise resource planning systems," CIO, 22 January 2019.

[2] M. Korolov, "AI technology: When to build, when to buy," CIO, 9 April 2019. M. Branscombe, "How to decide when to buy software and when to build it," CIO, 5 July 2016. C. Doig, "How to determine when to build or buy enterprise software," CIO, 5 November 2015. A. Bajaz, "Large Scale Requirements Modeling: An Industry Analysis, a Model and a Teaching Case," Journal of Information Systems Education 17, no. 3 (2006): 327. P. S. Traylor, "To Buy or to Build". InfoWorld, 13 February 2006. F. Daneshgar, G.C. Low and L. Worasinchai, "An investigation of 'build vs. buy' decision for software acquisition by small to medium enterprises," Information and Software Technology, Vol. 55, Issue 10 (October 2013): 1741-1750. P. Hung and G. C. Low, "Factors affecting the buy vs build decision in large Australian organisations," Journal of Information Technology, 23 (2008): 118-131.

[3] For ERP failures, see J. Fruhlinger and T. Wailgum, "15 famous ERP disasters, dustups and disappointments," CIO, 10 July 2017. www.cio.com/article/2429865/enterprise-resource-planning-10-famous-erp-disasters-dustups-and-disappointments.html, accessed 23.08.2019. [note: the 15 in the title and the 10 in the url is correct, not a typo]

[4] M. Talalay and T. Giles, "Preliminary Report on the IT Usage, Potential and Capabilities of Acomarit (UK) Ltd", private, 2 October 1996

[5] A. Chapman, C. Potter and M. Talalay, "The IT Function in Acomarit: Description, analysis and outline recommendations," private, 2 December 1996

[6] A. Lorenz, 'UK: The incredible dinosaur – How BTR is re-inventing itself," Management Today, Jan. 1, 1998

[7] The operating system was called PICK and it incorporated within itself a quasi-relational database. The natural enquiry language was "English-like" and could handle requests such as "sort by country all customers with an outstanding balance greater than ten thousand dollars.".

[8] For an interesting discussion of software and innovation in the service sector, see B. Engelstatter and M. Sarbu, "Does enterprise software matter for service innovation? Standardization versus customization," Economics of Innovation and New Technology 22, no. 4 (2013): 412-429

[9] Software providers rarely release source code. Customers are not able to make their own changes to the basic package; they need to rely upon their supplier.

[10] For some trenchant comment on this issue, see Cynthia Rettig, "The Trouble with Enterprise Software," Sloan Management Review, volume 49, no. 1 (Fall 2007)

[11] J. Hammond, P. Kalmbach and E. Bernstein, "Ozark Feed and Ag Corporation: The ERP Decision", Boston, MA: HBS Publishing (8 August 2018): 9.

[12] And there was a case study – a real-time experiment. I was sitting in reception at the Acomarit office in Glasgow leafing through one of the industry journals, when I came across an article describing how TeeKay (a tanker company with its main office in Vancouver) was attempting to implement the JD Edwards ERP package to run its operations – with a significant lack of success. In fact, some years later, in 2001, Teekay was running only the Financial and Purchasing modules in the J D Edwards One World ERP package. An enquiry in 2001 from the Applications Support Manager at Teekay on JDElist said, "My company is currently conducting a CRM package selection. We have narrowed our shortlist to a small handful of vendors. The CRM system will need to interface to our JDE OneWorld implementation (Financials & Purchasing) as well as to yet-to-be-implemented scheduling and operations systems (custom and/or off-the shelf)." www.jdelist.com/vb4/showthread.php?t=4235&styleid=2 accessed 11.08.2019.

I encountered a very similar issue when responding to an RFP (request for a proposal) from a clothing manufacturer; a conventional ERP package would not be appropriate, in large measure because SKUs (stock-keeping units) included size, style, and colour.

[13] See Chapman, Potter and Talalay.

[14] The shipping industry works in dollars – as do all globally traded commodities. Companies normally have to account in the currency of where they are resident; however, exceptions can be made in cases such as this.

[15] Branscombe, "How to decide when to buy software and when to build it,". My emphasis.

[16] Some ten years later, another ship manager client and I compared ShipNet and the planned maintenance module in ShipSure. We concluded that both were good but that ShipSure was the better in terms of usability.

[17] There is in a way a fourth indicator. We were in the process of spinning the IT department out into a separate company (as I had done at BTR some years previously). We had secured a partner in the maritime communications business. It was going to inject working capital and bring its large user base to the table. Acomarit would provide the intellectual property and the personnel. Everything was in place for a very successful spin out. Unfortunately, both Acomarit and our partner were subject to takeover bids (Acomarit by V.Ships, the largest in the industry). Both takeovers were successful, and that put paid to the idea of the spin out.

[18] With care, because we had to maintain upward compatibility with new releases from our American supplier. We could not and would not supply this source code to our clients. The fact that we had the code makes Don Tapley's edict about changing operational procedures even more powerful. Note, by the way, while I talk about buying software, in fact one usually buys a licence to use the software.