SPE 38120

Buy, Build, Beg or Borrow: Delivering Applications in the New Age of Software Development

Todd Little and Steve Webb, Landmark Graphics

Abstract

Changes over the last decade in both the oil industry and the computer industry have substantially complicated the process of delivering software applications. The oil industry has gone through a significant downsizing, while computing advances have users constantly demanding more complete and integrated solutions. Several oil companies' response to this dilemma has been to establish a policy of "Buy, Don't Build." This mindset is often useful, but does not directly address the real question of how to maximize and leverage limited resources in order to efficiently deliver the necessary applications to the user community.

This paper delves beyond the bipolar buy versus build question to present experiences with various approaches that have been used to deliver software. Among the methods discussed are software tools, research institutions, consortia projects, alliances, and industry standards.

Introduction

Both the oil industry and the computer industry have gone through significant evolution during the last decade. The challenge of delivering software applications to the oil industry has been intensified by the rapid advances in computing technology and further complicated by the downsizing of the petroleum E&P industry. The old paradigm where oil company internal R&D labs deliver the software applications to their internal customers is difficult to maintain with these contrasting forces. R&D departments have needed to change from pure research organizations to value added entities.[1] As a result some oil companies have established a new mindset of "Buy, Don't Build."

While this is often a useful distinction, buy versus build is far too polar. The complexity of delivering software in today's rapidly changing environment requires maximizing limited resources. In our efforts as a software vendor we have had experiences with a number of techniques to leverage our software development resources, among them software tools, research institutions, consortia projects, alliances, and industry standards. These alternatives all fall somewhere in the buy-build spectrum. This paper presents some of the benefits and limitations of these approaches.

Changes in the E&P and Computing Industries

Oil companies have always recognized the need for computer software. Ever since the first vacuum tube computers, oil company R&D departments have been right behind providing software technology. Access to software solutions was

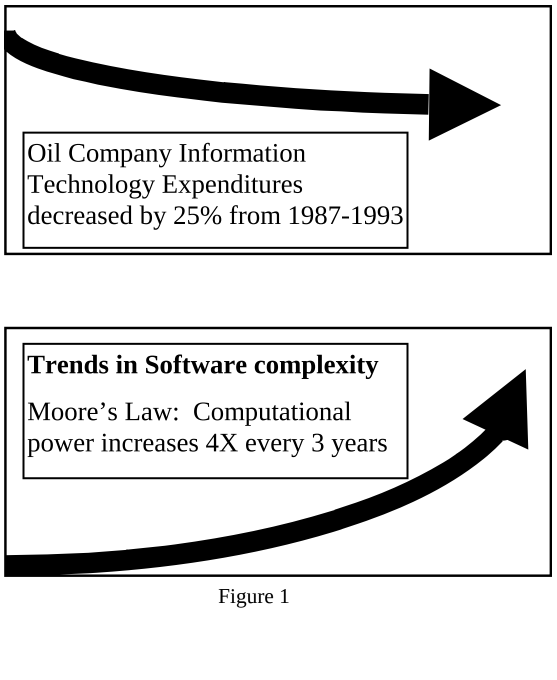
Oil Company Information Technology Expenditures decreased by 25% from 1987-1993
Trends in Software complexity
Moore's Law: Computational power increases 4X every 3 years

Figure 1

considered a competitive advantage. However, over the last decade both the oil industry and the computing industry have gone through major changes. Decreasing or flat oil prices have resulted in significant downsizing in the E&P industry, particularly in oil company R&D organizations.

As oil companies searched for cost cutting targets they realized that the cost of developing their custom applications was quite large. Software development projects repeatedly overran their initial estimate, often twofold[2]. In 1996, 73% of corporate America's software projects were canceled, over budget, or late.[3] Just maintaining software is expensive and often as much as 2/3 of the software cost is incurred during the maintenance phase.[4]

As shown in Figure 1, it is estimated that the oil companies reduced their information technology spending by 25% from 1987-1993[5]. Meanwhile, computing technology has continued to advance, with major growth created by the personal computer and the computer workstation. Ironically, the advances in computing technology and software solutions is perhaps one of the factors that enabled the oil industry to survive during this downsizing era.

Thus the need for software applications grew while the resources available to develop those applications shrunk. Specialized software vendors sprung up and grew to respond to this need. Most of these vendors started by simply providing much of the same services that had been provided by the downsized internal R&D. Typically vendors developed a specific solution to a domain need.

However, the constant drive to increase efficiency has led to other important changes in the way we find and produce oil and gas. Reservoir studies are now often performed by integrated teams of geophysicists, geologists, petrophysicists, and reservoir engineers.[6,7,8,9] Sometimes these integrated teams are made up of representatives from several oil companies that are partners in the field. This has increased the need for higher degrees of integration between software applications. This demand for integration added another major complication to software development and raised the cost of developing software considerably. The larger vendors responded to this need by improving their own integration and by acquiring other vendor applications to bring under their fold. The consolidation of the vendor industry over the last two years is remarkable. Of the 9 vendors involved in the Simulation Application View of Epicentre (SAVE) project, all but two merged, were acquired or were the acquirer of one of the other participants.[10]

Yet the need to develop new and more integrated software application suites seems insatiable. Oil companies and vendors still must optimize their scarce developer resources. In our effort to produce software applications we have experimented with a number of different options which have the potential to increase the leverage of our developers.

Software Tools

As a software developer one method that we have used with success is to look at the question of buy versus build with regard to our own application development. In many cases we have found that development time can be substantially reduced by purchasing components which can be used and reused in software applications. As this is covered extensively in a previous paper we will not go into great detail.[11] Suffice it to say that this method can be very effective in reducing software development costs. It is not without problems, however. Even though the tool is being supported by the tool vendor, once it is included in an application it is the application developer's obligation to support the overall package. This can be frustrating, particularly if release cycles are out of phase. We have found that the best solution to this problem is to establish good relations with the software tool provider. Another concern is overall software integration issues. One software tool may be very suitable to a particular task, but if it is incompatible with another tool or other parts of the application suite, then the gains realized may be eroded by substantial rework in order to make it compatible.

Sometimes it is not a question of buy versus build, but rather a question of take versus build. Today the Internet opens up a great wealth of public domain tools. Sometimes these tools can be used directly as is. Often what is available is a prototype of a code fragment which can be used to jump start a project. We have successfully made use of both types in our applications. But while the price is right in one sense, there is no such thing as a free lunch. There is no doubt that there is an abundance of public domain software. The downside is that a lot of it is junk. It may take a lot of "surfing" in order to find something that fits a particular need. And do not forget the high cost of maintenance. Once a software developer incorporates the public domain software into their application, they have the obligation to maintain it. As opposed to a purchased tool, a public domain tool is unlikely to have a tool vendor available to help with support or maintenance. This free software may end up costing a significant amount in the long haul. Caveat Emptor.

Research Institutions

One of the big advantages of the Internet is that it opens the door to the activities of other industries. We have also had similar benefits from working with national laboratories. In 1995, the Department of Energy formulated the Advanced Computing Technology Initiative (ACTI). We were involved in a number of these projects and found that we had particular success from one of the projects.[12] There were a number of reasons for this success. This project involved a very limited number of participants and was very focused in its objectives. That focus, combined with the ability of the lab to leverage some of the technology from other industries, such as aerospace, led to a very productive project. We have also been involved in a number of other consortia projects with research organizations and universities. Our experience has shown that the effectiveness of these projects degrades significantly as the number of participants increases and the scope broadens.

Consortia Projects

While research institutions can help leverage technology from other industries, often the research we need is very specific to our industry. Whereas in the past each individual oil company's R&D department was seemingly in competition to do this research, now often the desired objective is simply getting the technology to the business units and doing so as efficiently as possible. In some cases a software vendor conducts its own research and comes out with a product hoping that it addresses the market needs. Another model is to conduct the research as a consortia project. Here the software vendor has an idea for research technology with a fairly good indication that this technology is feasible. The vendor solicits oil company sponsors who participate in funding the project and monitoring the results.

We have had a number of successful consortia projects in the areas of reservoir simulation solver and parallelization technology.[13,14,15,16] The consortia members have benefited by getting the technology to their users in a timely manner and by having a say in what went into the technology. In some cases the participants use our software directly, and in other cases have incorporated the produced software components into their own in-house software development. In general, they have also benefited by obtaining the technology at a lower cost than if they had waited for a vendor to produce the product on their own. Indeed, oil companies often take the view that if funding is not provided the products they want will not be produced. As a vendor, we were able to reduce the research risk and were able to receive direct feedback from the oil company during the research development. A result is that we have produced products usable by a broad market.

If membership of these consortia is sufficiently large, projects of this kind can have a very high leverage of resources and this is the real attraction of these consortia. Often, for no more than the equivalent cost of a man year of internal effort, the oil company can have access to research or software that is the product of many man years effort. Moreover, the software vendor can then take to market a fit-for-purpose product than already has an established user base, thereby leveraging the value of the software far beyond its original funding.

Alliances

Another of the today's buzzwords is "alliance." The definition of "alliance" in The Concise Oxford Dictionary is "union ... joining in pursuit of common interests."[17] But most alliances are not born with union of purpose in mind. Rather they are an uneasy marriage of diverse objectives. These objectives will vary depending upon the type of partnership. For the software vendor the alliances can come in several forms: they can partner with oil companies, hardware vendors, or even with other software vendors. We have been involved in each of these three types of alliances and have had experiences to indicate what works well and what does not.

Oil company / Software vendor. Our alliances of this type have fallen into three categories: funded development where the oil company participates in the specifications of the software but leaves the software development entirely to the software vendor, outsourcing where the oil company builds a prototype or possibly a complete product before turning it over to a vendor for maintenance, and joint development where both oil company and vendor actively work together to develop the software.

Funded development is the most traditional vendor-oil company relationship. Often an oil company has a specific need, but does not have the capability or the desire to develop the software internally. In our experience with funded development, the oil company has participated closely in the specification phase but has relied entirely on the software vendor for the development of the software. The oil company has participated again prior to software release in the latter stages of testing. The benefit of this model for the oil company is that they get early access to software designed for their specific purpose. For the software vendor the benefit of this approach is that all software development is carried out under its control. However, to provide the custom software development services, often at short notice, means that extra resources have to be found or other planned developments get delayed. The real issue here is leverage. If all that has happened is that one-off development is now done by software vendors rather than oil companies, then in the big picture we have done nothing but shuffle resources. On the other hand, if the developed software fits into the software vendor's strategic offering, then this can be a big win for both sides.

An outsourcing relationship has a lot in common with the funded development model, but it starts in an entirely different way. Usually the oil company has developed software that they to want to have available to their users but which they do not consider proprietary. At some point, they consider licensing or outsourcing it to a software vendor in order to make the technology generally available and to reduce their maintenance expense. The oil company wins by keeping the desired technology available at a reduced expense. Again the software vendor gains if they are able to leverage this technology and merge it with their existing software offerings. There are a number of caveats for the vendor, however. Often what is developed is a prototype or simply an algorithm. Turning that into a product and integrating it with an existing product offering can be a significant effort, perhaps as much as nine times the original effort.[18] Furthermore, maintenance is now the responsibility of the vendor. These problems are exasperated if this relationship takes the form of licensee-licensor rather than as a partnership or alliance. If the relationship is formed primarily as a cost cutting attempt, then it will rarely be effective. However, if the product fits the vendor's strategic direction and can be leveraged appropriately, then the other costs may be worthwhile.

Joint development is more of a true alliance. In this model both the vendor and the oil company participate in all phases of the software development project. Resources are contributed from both participants. A very close and long term relationship between the vendor and the oil company can develop which can be very beneficial for both participants. The oil company gets fit-for-purpose software and the vendor has a ready made market for its product. For the vendor an additional benefit of this relationship is the access to well-qualified staff within the oil company. These staff tend to be well in tune with the needs of the oil company and the needs of users in general. On the other hand, joint development is very hard to coordinate, especially at different sites. To prevent two versions of the software emerging, good communication is paramount and sharing of resources desirable. We have found that to keep the code development properly managed it is necessary to rotate our development staff through the oil company's office on a regular basis. Ownership of the oil company developed code is another issue. Usually the oil company does not want the ongoing responsibility for code maintenance and is only too happy to relinquish ownership. However, that too creates problems because an effective technology transfer mechanism has to be found to teach the software vendor how to use, modify and understand the oil company developed code. This is a particularly challenging task.

The key challenge to making these alliances successful for the software vendor is to find a way of leveraging the work. If these developments can be added in a way that makes them applicable and attractive to a larger user base than the original clients then adding these enhancements is a sound business proposition. If this is not the case then support costs for these enhancements will soon erode any benefit. Indeed unless a development is in the long term strategic direction of the software product, the software vendor should always say no to the oil company no matter how well the original development is funded.

Software vendor / Hardware vendor. An alliance between a software vendor and a hardware vendor tends to form in reaction to some market force rather than be planned as part of an overall long term strategy. Often in order to break into or maintain market share in a target market segment a hardware vendor will shower the software vendor with free hardware and resources. Likewise a software vendor may try to get a competitive advantage by teaming up with a hardware vendor to exploit a new hardware innovation or development. We have succumbed (and continue to succumb) to both these temptations. However, the hidden cost is high to the software vendor in both these cases. Free machines often come with new or even unreleased systems software which require significant debugging (by the software vendor usually). And the free resources do not usually have the required skills and experience to be useful without significant investment of effort on the part of the software vendor. In this age of rapidly changing hardware technology, any new hardware innovation is usually short lived, and if it is of any significance, taken up by another hardware vendor almost immediately.

We have, however, had some long term highly productive relationships with some hardware vendors. One such joint cooperative development agreement has lasted over five years and has evolved as the hardware and software markets have evolved. This relationship was put together for the long term mutual benefit of both parties rather than aimed at a short term tactical objective. During the relationship, there have been some short term tactical projects, but these have been executed efficiently because we already had the infrastructure, and more importantly the personal relationships in place. No steep learning curve by the hardware vendor was necessary and the resources offered were efficiently used. Machines were already in place or could be added easily to our existing infrastructure, and the programming help was already familiar with the demands of our applications. From the hardware vendor's perspective, it had a long term supply of applications ready to go on its latest hardware - the most important requirement for selling new boxes.

Software Vendor / Software Vendor. It is also usually market forces which prompts a software vendor to pursue an alliance with another software vendor. As a software vendor is building an integrated product suite they will realize that they cannot do it on their own. By forming a relationship with another vendor with expertise in the missing critical areas both parties hope to leverage off the other. In order for these partnerships to be successful it is important not just to have a collection of products but to also have them well integrated. This requires a close relationship with strong commitments from both sides. However, these relationships can be difficult to manage as either party may view the other as a potential competitor ready to steal its market.

Standards

One trend over the last decade common to both the oil industry and the computer industry has been the strive towards standards. The proliferation of computing architectures and software implementations almost guaranteed an integration nightmare. There were hopes that standardizing on common interfaces would drastically reduce the effort required to produce integrated software.

In the computing industry this worked fairly well with organizations such as X/Open and OSF (Open Software Foundation). One of the primary reasons for success was that the computer hardware was evolving with the standards. In fact, it could be said that those companies that tried to retrofit their older hardware and operating systems were the ones that did not fare well in this environment. Overall the computing industry did not face a large "legacy" problem.

In oil industry software development we do not have that luxury. We have both legacy data and legacy applications. Our data is a major asset that we cannot ignore, and modifying legacy applications working off legacy data stores is a major undertaking. Furthermore, in the oil industry we have a lot of data and data types. The Petrotechnical Open Software Corporation (POSC) has taken on the monumental task of defining standard data mappings for all of the data that we use in the E&P industry. Industry take-up of these standards has been slow for a number of reasons but primarily because the enormous cost of legacy migration. This is perhaps ironic given that one of the purposes of defining standards was to reduce the cost of delivering integrated solutions. While this is likely true in the long term, there is no doubt that migrating legacy applications and legacy data to any standard is very expensive. Studies have shown that the cost of making a change in an application once it has been deployed can be over 100 times what it would have been had that change been included in the specifications.[19]

One of the POSC related groups we have been involved with has had some success working with the POSC standards. Clear business drivers led a group of oil companies and vendors to form the S.A.V.E. alliance.[10] They of set out with the specific objective of validating the POSC data model in the area of reservoir simulation. While the project was relatively successful at evolving an improved data model for reservoir simulation, industry take-up of the model has been slow and some might argue that since the S.A.V.E. alliance has disbanded that it was not successful. We believe the contrary. The S.A.V.E. members realized that the business environment had changed and they had accomplished as much as could be accomplished given the business drivers. Short term gains were unlikely, but they recognized the potential for the long term and remain committed to the objectives. In that vein, the findings from the project are of great help in planning the evolution of existing and future applications.

Conclusion

There is no doubt that we have entered a new age of software development in the exploration and production industry. The software needs of the users in the industry has long outgrown what a few scientists in an R&D department can provide. Oil company R&D departments can still play a critical role in the software development process, but their role has changed from technology developers to technology providers. They now must leverage their efforts off others. Likewise the software vendors have realized that they can not do it all on their own. Fortunately, software providers have a portfolio of potential software development methods which can help leverage developer resources. Software tools, research institutions, consortia projects, alliances, and industry standards can all provide effective means to help deliver software applications. However, the cost of leveraging these tools should not be overlooked or underestimated. The investment required to start leveraging these tools can be significant. Not surprisingly, in nearly every case where we have used one of these methods and had a positive experience, the project objective has been consistent with our long term strategic objective. Likewise, most of our problems have come when we have looked to these methods to provide a quick fix.

References

1. Clementz, D.: "Company R&D: Does It Add Value to the Bottom Line", *Journal of Petroleum Technology*, Vol. 49, No.2. (February 1997).

2. King, J.: "IS Reins in Runaway Projects", *Computerworld*, Vol 31, No 8.

3. DeMarco, T.: Controlling Software Projects. Yourdon Press, Inc., New York, NY, 1982

4. Zelkowitz, M. V.,:"Perspectives on Software Engineering," *ACM Computing Surveys*, June 1978.

5. Cambridge Energy Research Associates: The Quiet Revolution: Information Technology and the Reshaping of the Oil and Gas Business, Cambridge Energy Research Associates, 1996.

6. Caamano, E., *et al*: "Integrated Reservoir Interpretation", *Oilfield Review*, (July 1994) 50.

7. Balough, S., *et al*: "Managing Oilfield Data Management", *Oilfield Review*, (July 1994) 32.

8. Dria, M.A. and Aronstam, P.: "The Use of Integrated Software for Improved Reservoir Management, paper SPE 28934 presented at the 1994 SPE Annual Technical Conference and Exhibition, New Orleans, LA, Sept 25-28.

9. MacKenzie, A.S., "Trends in Reservoir Performance Prediction, paper SPE 28387 presented at the 1994 SPE Annual Technical Conference and Exhibition, New Orleans, LA, Sept 25-28.

10. Haringa, H., Little, T., Aydelotte, R., and Austin, A.: "SAVE: An Alliance for Reservoir Simulation Software Integration", SPE 36759, *Journal of Petroleum Technology*, Vol. 48, No. 6. (June 1996).

11. Little, T,, Sinclair, C, and Rahi, M, A.: "Buy Don't Build: What Does That Mean for a Software Developer", SPE 28255 *Journal of Petroleum Technology*, Vol. 47, No 6. (June 1995)

12. Bethel, W., Jacobsen, J., Austin, A., Lederer, M. and Little, T.: "Implementing Virtual Reality Interfaces for the Geosciences", paper presented at the 1996 Virtual Reality in the Geosciences Conference, Halden, Norway, June 24-26.

13. Wallis, J. R., Foster, J. A., and Kendall, R., P., A New Parallel Iterative Linear Solution Method for Large Scale Reservoir Simulation", SPE 21209 presented at the Eleventh SPE Symposium on Reservoir Simulation, Anaheim, California, February 17-20, 1991.

14. Wallis, J. R, Nolen, J. S., "Efficient Linear Solution of Locally Refined Grids Using Algebraic Multilevel Approximate Factorizations", SPE 25239 presented at the Twelfth SPE Symposium on Reservoir Simulation, New Orleans, Louisiana, February 28-March 3, 1993, No. 12 (December 1976), pp

1226-41.

15. Killough, J. E., Foster, J. A., Nolen, J. S., Wallis, J. R., and Xiao, J., “A General Purpose Parallel Reservoir Simulator”, presented at the 5th European Conference on the Mathematics of Oil Recovery, Leoben, Austria, 3-6 September 1996.

16. Killough, J. E., Camilleri, D., Darlow, B., and Foster, J. A., “A Parallel Reservoir Simulator Based on Local Grid Refinement”, SPE 37978 presented at the Thirteenth SPE Symposium on Reservoir Simulation, Dallas, Texas, June 9-11, 1997.

17. The Concise Oxford Dictionary, Sixth Edition, Oxford University Press, Oxford, 1976, 26.

18. Brooks, F. P. Jr.: The Mythical Man-Month, Addison-Wesley Publishing Company, Inc., Reading, MA, 1975

19. Boehm, B.: “Software Engineering”, *IEEE Transactions on Computers*, Vol C-25