

Key Findings

June 24, 2004

Complexity Trumps Law Suit

As I watch the Oracle U.S. Department of Justice proceedings unfold, I wonder if these two entities are even on the right page. The *same* page? Sure. But the *right* page? I dunno. I personally don't believe competition would dry up if Oracle is allowed to acquire PeopleSoft. But in this wild world we call the free market I wonder how Oracle can come to the conclusion that the acquisition would be good for it.

Right now lots of people are forecasting a period of consolidation in the enterprise software industry and they use this case as emblematic of the era. But when I hear so many know-it-alls predict a slam dunk, I get wary and tend to run in the opposite direction. Let's grant the assumption that the DOJ will lose the case and Oracle will buy PeopleSoft. I think it would be a bad thing for Oracle and PeopleSoft if Oracle were to prevail. Believe it or not, if all this were to come to pass, I think the average consumer of enterprise software would barely notice because I think the market is moving in another direction. Now, before you start wondering if I have been out in the sun too long, consider the following.

The Rise of Complexity

Enterprise software has reached a level of complexity where it has become too difficult and too expensive to do much with it. It is expensive to buy, costly to implement, and flocks of greenbacks disappear into the Bermuda Triangle of customization, maintenance, and upgrades each year. This has become the classic argument for larger, monolithic application suites and it is an important driver of the Peoplesoft acquisition.

A good deal of the cost of software is tied to labor. In this automated, post-modern age, very expensive and smart people — who in alternative careers might have cured cancer or landed us on Mars by now — still more or less hand-craft the solutions that millions of us use and take for granted — except when they break. And they break often because there are so many interconnected moving parts sometimes working at cross purposes.

Putting two or more disparate sets of source code together really *is* rocket science or brain surgery (see above). A case in point — in the course of the trial so far, Microsoft and SAP have announced that they had at one point contemplated a merger but that both sides had scuttled the idea at least in part because of the complexity posed by the integration task.

Design Rules

A few years ago MIT press published a book called “Design Rules” by two Harvard Business School professors Carliss Y. Baldwin and Kim B. Clark. The title embodied a dual entendre communicating both the idea that there are inherent rules for designing complex systems and a more hip and sage expression when “rules” is read as a verb.

What I learned from that book is that complexity is a relative thing and often the end of a paradigm or a model for how we do something. When we reach levels of high complexity a paradigm shift can not be far off and the enterprise software industry today is a prime example. The solution to complexity is, almost invariably, modularity — breaking the problem down from a monolithic one to several or many that adhere to organizing standards for how they all fit together.

The NASCAR Approach

When the hardware industry got too complex for a single vendor to build and control all the components, manufacturing of memory, disk, and tape drives separated out and the third party peripherals industry was born. The result was better, faster, and cheaper peripherals, which in turn made the mini-computer revolution possible and further modularization and differentiation led to the distributed systems we have now.

Today the insides of even name brand PCs look like the outside of a NASCAR with all the logos of cooperating products serving the same set of standards. A better example might be a racing bicycle because while the car carries a lot of sponsor logos that don't actually run in the car, the bicycle actually uses the components it advertises. Whatever.

Hardware was not the only venue where modularity spawned a new approach. The same modularization process was responsible for operating systems, databases, workflow and business rules engines, and other software innovations that continue today at the application level. The enterprise software applications industry has been quick to develop modularity — look at all the best of breed vendors out there — but relatively slow to come to the integration standards table, but that too is changing, and that is what makes the PeopleSoft acquisition so treacherous for Oracle.

The New Paradigm

Just about every industry we know of has gone from being vertically integrated to becoming distributed over the last few decades, except for enterprise application software. Think about it. Cars can be assembled anywhere from parts made all over the world. Engines and transmissions might come from Canada, glass and batteries from Mexico and eventually it all gets assembled in Detroit or Marysville, OH. Shoes might be designed in North America but they might be built in the Far East, Brazil, or Italy. And movies might be distributed by Hollywood but independent film makers live and work everywhere.

It's time for the software industry to undergo the same kind of de-verticalization which will result in a two tiered structure in which there will be publishers and developers. The publishers will aggregate demand, develop and enforce standards and do all of the sales and marketing. Developers will turn to what they do best — developing software — and turn over what they like least — marketing and sales. The model will be predominantly a hosted one; the better to keep the installed base on a single version of code that can be easily serviced and sold into.

Imagine the attractiveness of a software publishing company with millions of deployed seats in a hosted environment. Such a company will have a ready-made distribution channel for new software and developers with good ideas will flock to it to sell their applications.

From here to there

To be sure all of the pieces are not in place yet to make this new model a reality, but there are tantalizing glimpses of the future if you know where to look. Web services, hosting, and integration servers are key components and they are arriving on the scene now. The ubiquitous presence of the Internet has already shown us that national boundaries are no impediment to good ideas and bright developers can live anywhere and make meaningful contributions. Still, much work needs to be done to get so many disparate applications to work together as one and we shouldn't downplay the contractual and legal challenges that must be worked out. But this direction of change is developing a momentum of its own.

Perhaps the best example of a system in action today that incorporates many of the tenets of the new paradigm is Sforce from Salesforce.com. Right now Salesforce has over 1,000 developers who use standard tools to build extensions to its core applications or who use its core data model and services to build whole new applications that run seamlessly on its network. Most importantly, those developers are working for free, simply to have the opportunity to sell their wares to Salesforce's installed customer base. One thousand free developers enable Salesforce.com to play David to the industry Goliaths.

How to win the law suit

The move for a company of Oracle's size today is not to gobble up other companies and their code sets or to wage a never ending battle with the Justice Department. We have reached a level of complexity that serves as an asymptote to further combinations. The challenge now is to diversify by building the largest ecosystem of compatible applications possible, hosted by an infrastructure that is so customer friendly, so robust, and so secure that no customer would think of using software from any other vendor.

It's still not too late for Oracle to win the law suit. All they need to do is quit.

###

