



Workday's Technology Strategy

A Revolutionary Approach to Redefining Enterprise Software



Contents

Executive Summary.....	3
The New Wave.....	4
Ubiquitous Computing.....	4
The Emergence of Standards.....	5
Middleware Is Not Enough	5
Limitations of Current Systems.....	6
A Revolutionary Approach	6
Object Management Server for Data and Logic.....	7
Limitations of Complex Relational Data Models	7
Encapsulating Data and Logic: Object-Oriented Design for Flexible Data Modeling	7
Committing to a Peer-to-peer World.....	8
Completely Definitional Development	9
Presentation (User Interface) Separate from Business Logic.....	9
Summary	10

Workday's Technology Strategy

A REVOLUTIONARY APPROACH TO REDEFINING ENTERPRISE SOFTWARE

Executive Summary

A new architectural paradigm for enterprise software is upon us, and although the industry has not yet settled on a single definitive name for the phenomenon, it is clear that this “new wave” will be defined by certain key technologies and standards including service-oriented architecture (SOA), web services, XML, and universal use of the Internet as an all-pervasive communication medium.

Today, development teams are increasingly embracing open source tools, object-oriented development techniques, XML, and web services. Workday's use of these technologies does not in itself make our approach revolutionary. So, what does? At Workday we are building applications using an object model with no code to do XML processing — as opposed to the traditional way of building applications using an object/relational model with Java to do relational/SQL transaction processing. The benefits to our customers are greatly enhanced flexibility to change as your business changes, reduced complexity of integration with other systems, and radically reduced maintenance requirements.

This white paper begins with a discussion of the new wave of computing architecture and the failure of legacy ERP vendors to adequately address today's challenges with their “heavyweight” relational approach. We then offer some insight into why we think our approach is revolutionary. We will discuss Workday's:

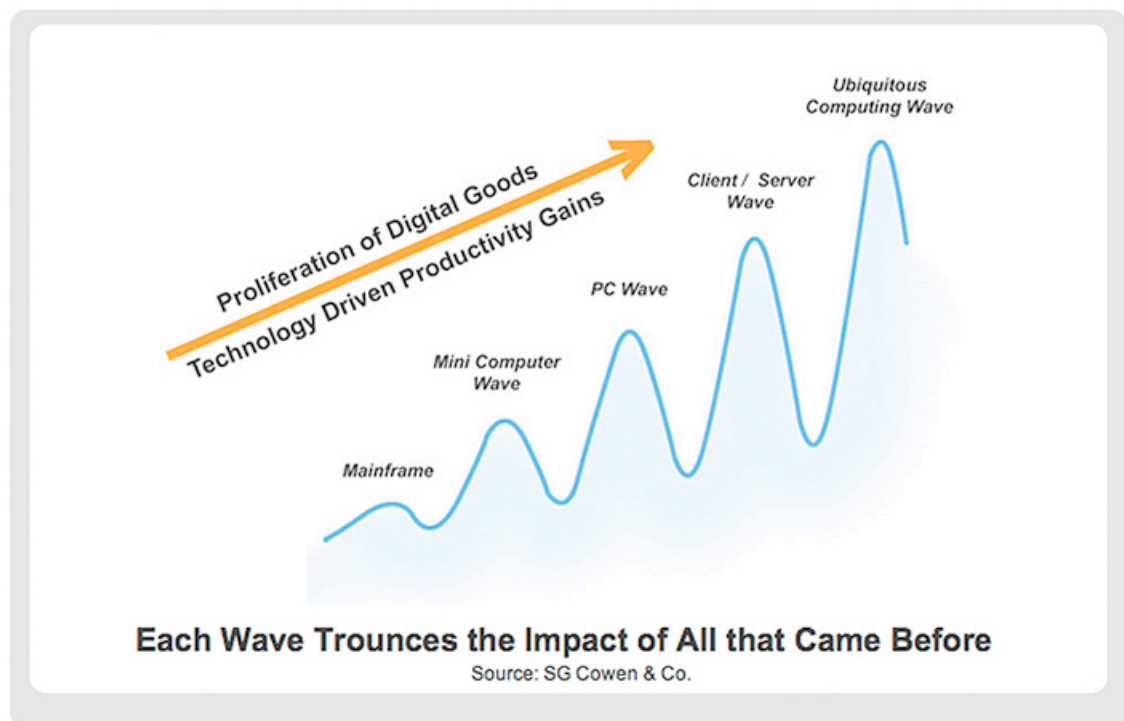
- Completely definitional application development based on an advanced toolset — in other words, development without coding. Definitional development is not only faster and much less expensive than coding in 3GL languages such as Java; it is also far less error-prone, and serves as an insurance policy against code maintenance headaches that invariably lead to application decay over time.
- Presentation layer (user interface or UI). The Workday UI is completely separate from the business logic in our system. As UI standards tend to change rapidly, this enables us to update or even completely re-architect our presentation layer without touching the underlying business logic.
- Pure object-oriented development methodology. The use of an object model to represent the data, relationships, and logic for an entire system means that data, relationships, and logic can be changed — as your business inevitably changes — more easily and cost-effectively.

The New Wave

Ubiquitous Computing

The underlying architecture of traditional client/server ERP systems have run their course. These systems have become onerously complex and expensive to maintain and integrate, and their inherent rigidity has many organizations looking for alternative architectures. The good news is that we are moving into a new wave of computing, as shown in the following diagram.

The 5 Architectural Waves



The new architectural wave has been given a host of different names: ubiquitous computing, peer-to-peer computing, computing enabled by service-oriented architecture (SOA), and so on. In the world created by this new wave, information extends far beyond the four walls of the enterprise. And indeed, today we are seeing a proliferation of devices — not just desktop and laptop computers slaved to a server, but also fully mobile, wireless devices as well as “appliances” such as production and retail equipment. The challenge is to link all such devices — easily and cost-effectively — to enterprise data and processes. And thus fulfill the promise of ubiquitous computing.

Previous technology waves — from the days of mainframes and dumb terminals to the client/server paradigm — have all been *hierarchical* to some degree. The essence of the new wave’s computing paradigm is loosely coupled, peer-to-peer communication among a virtually unlimited number of devices and systems.

The Emergence of Standards

This new approach to computing is being enabled by the emergence of standards including:

- Web services (applications that communicate with other applications using XML and the Internet).
- Service-oriented architecture (SOA — a set of common standards supporting loosely coupled, highly interoperable web service applications that exist as nodes in a global peer-to-peer network).
- XML (Extensible Markup Language — a flexible method for expressing any data as text; XML facilitates the sharing of data across different systems).
- The Internet itself (an all-pervasive communication medium).

These standards allow us to think about an application as a set of related business objects which communicate with the outside world via XML.

Vendors in the 1990s that tried to “adapt” their legacy systems to the then-new client/server wave were quickly supplanted by vendors creating systems built natively in the client/server model. A similar shift is occurring today. Workday applications are being built natively in the new world of SOA, the key to the new architectural wave.

Middleware Is Not Enough

Much work is being done today in the realm of middleware, as many vendors are focusing on the “plumbing” that is the price of entry to the new world of ubiquitous computing. IBM’s WebSphere, Microsoft’s .NET, SAP’s NetWeaver, Oracle’s Fusion middleware, Cape Clear and other ESB (enterprise service bus) providers are all examples of technologies that let organizations participate in the new computing wave by wrapping existing applications in a “surround” that allows these applications to communicate with other peer-to-peer solutions via web services.

Middleware plumbing is necessary, valuable, and good — it will help all applications talk to each other more seamlessly. But, an old application wrapped in middleware is still an old application. In other words, in the absence of a more revolutionary approach to *application design itself*, such plumbing only adds more “weight,” complexity, and cost to a core of aging legacy applications that were designed as many as 15 years ago.

Middleware doesn’t solve the problem of the cost and complexity of current ERP systems; in fact, it can be argued that if you start with enormous, rigid applications that are inherently difficult to install, and cumbersome and expensive to maintain, and then wrap them with additional layers of technology, you have only made the problem worse.

Building middleware is necessary, but not sufficient. By itself, middleware doesn’t enable organizations to realize the full promise of the next wave. For organizations to harness the full potential of the emerging architectural paradigm, applications must be built that are granular enough to take advantage of the new peer-to-peer world. These applications will, in turn, be nimble and cost-effective enough to deal with business change and provide a competitive advantage.

Limitations of Current Systems

Since the data, relationships, and processes you need to track for your business objects constantly change, your enterprise systems must offer the flexibility to change. Current enterprise systems do not offer this agility. Or, more accurately, it's not that change is impossible — it's just too time-consuming and expensive with current rigid architectures built on complex relational data models, fixed relationships between key business objects, “hard-wired” business processes, and fixed business logic. If you wish to change a single data definition, relationship, or process, you may be faced with forbiddingly complex logic rewrites that touch a thousand other elements in the system.

This is precisely the reason why many enterprise system implementations today include an aging and unchanging “core” surrounded by newer point solutions for the new data, relationships, and processes that an organization wishes to track. Recruiting, enterprise learning, and performance management are all examples of functionality that organizations now see as essential to core HR. Financial systems are facing this same issue augmenting the core financial solutions with internal control, employee expenses, business intelligence, service procurement, activity-based accounting, balanced scorecarding, budgeting, and invoice presentment and payment. In either case, this functionality is typically implemented outside the core and integrated with it, because the core itself is too rigid to be changed.

Workday has taken a radically different approach. Unlike traditional ERP vendors that are wrapping an inflexible core with middleware, Workday is focusing on developing, from scratch, applications built natively for the new wave.

A Revolutionary Approach

To realize the full promise of the new wave, Workday is pioneering new applications using pure object-oriented design that encapsulate data and logic together; such applications communicate on the basis of loosely coupled XML transaction processing, resulting in the maximum possible agility.

Workday has harnessed today's available technologies and standards in a revolutionary way. In the remainder of this paper we will discuss the following aspects of Workday's technology strategy:

- Workday takes a **purely object-oriented approach to modeling logic and data**, as opposed to a client/server relational or an object/relational approach. This frees us from the constraints of working with complex relational data models that rely on SQL processing or object/relational mapping, and enables us to take full advantage of the new paradigm of loosely coupled XML processing in a peer-to-peer SOA world.
- Using an advanced object-oriented toolset, Workday practices **completely definitional development** — that is, development without code. We develop by defining a model of what we want the software to do instead of programming code. This greatly simplifies application development and maintenance, drastically reduces the time and cost associated with changes to applications, and helps avoid the gradual application decay that plagues legacy code-based systems. Our approach is to create definitions and then interpret them at run-time.
- Our **presentation layer (user interface or UI) is completely separate from the business logic** in our system. As UI standards tend to change rapidly, this enables us to update or even completely re-architect our presentation layer without touching the underlying business logic.

Let's discuss each of these points in turn.

Pure Object-Oriented Approach to Modeling Data and Logic

Limitations of Complex Relational Data Models

Most current enterprise systems are built on complex relational data models whose continued use is constrained by certain intrinsic limitations: such data models typically involve thousands of tables; fixed relationships between key business objects result in inflexibility once the database is implemented; and hierarchical keys, foreign keys, and repeated, unchanged data in multiple rows lead to severely redundant data.

The most fundamental limitation of relational data models, however, is that they are separate from the application logic (which is usually some combination of 3GL and 4GL code), but the logic layer and the data layer are very tightly wired together. In the overall system, data is *defined* in the data model, but *referred to* all over the application logic layer. Processes are defined as a series of steps in the logic layer, but each step references all the data and relationship information it needs from the data layer.

The end result is that changes to data or to relationship definitions require complex analysis and change in the logic layer to keep existing functionality working and to implement any new functionality. Also, changes to processes in the logic layer are difficult because of all the data and relationship “context” information that each process step requires.

Encapsulating Data and Logic: Object-Oriented Design for Flexible Data Modeling

At Workday we do not use a relational data model. Our applications are built using object-oriented design only (not object/relational); this enables us to encapsulate the data as well as the processing logic around an object. Our approach is to define applications as objects in what we call an Object Management Server (an in-memory object database); relationships are then maintained between objects.

Encapsulating the data and the logic within an object model — and not being forced to tie them across into a relational model — means that data, relationships, and logic can be changed much more easily and cost-effectively.

In the object/relational approach used by some vendors, the need to map object models to relational models often limits what developers can do with inheritance or how granular classes can be. An object/relational approach also requires the relational model to be kept in sync with changes to the object model. In Workday's pure object-oriented approach, developers are not required to maintain (or even think about) mapping to the underlying database.

Benefits of Workday's object-oriented approach:

- **Data definitions** (attributes) can be added or changed without complex logic rewrites or complex database restructuring. This flexibility greatly expands the scope of what you can do with your Workday system. For example, if you install and implement Workday's core HCM system, you might initially wish to link it to an external recruiting system. A few years down the road, you might decide that you wish to model your recruiting data internally as part of your HCM core. At that point, you can change your Workday data model to include recruiting functionality — without necessitating a massive code compare and upgrade on the code level in addition to a massive database restructuring.
- **Relationships** can be added or changed without complex logic rewrites. For example, in legacy ERP systems, it is very difficult to change rigid organizational hierarchies once the system is implemented; if you wish to manage and track ad-hoc project teams or matrixed and networked organizations, you are typically forced to do it with separate systems outside your ERP. Workday, in contrast, gives you the flexibility to change the way you model the people relationships in your organization, not only pre- but post-implementation; within Workday itself, you can model and track project teams and the multiple reporting lines typical of matrixed or networked organizations, even if such relationships change frequently.
- **Process steps** are not hard-wired; they are loosely coupled to other steps — that is, linked solely by the exchange of XML elements. Consequently, process steps are easily separable, and the ordering of steps can be changed.

Committing to a Peer-to-Peer World

This, then, is what it means to truly commit to a peer-to-peer world: Workday applications are as open talking to other systems as they are talking to themselves.

In addition to lowering the cost of maintenance, Workday's use of object-oriented-only design (and of definitional development, discussed in the following section) makes it possible for us to respond quickly to requirements for fundamental change to our core data models and business processes as the need arises — and the need for change in enterprise applications arises almost continuously as business practices rapidly evolve.

Completely Definitional Development

Definitional development is programming without coding. Workday's toolset enables our application developers to define a class model and use definitional tools to create all aspects of that model (classes, attributes, relationships, and methods). Our Object Management Server uses methods to manipulate and update application data; these methods are also created without code.

There are important benefits to definitional development (as opposed to coding in a 3GL language such as Java or generating code from metadata):

- In the definitional creation of methods, all processing steps get defined to the lowest level. The end result is that we end up with about 19,000 method definitions to perform the processing we need to do in our HCM application and in our tools. That's 19,000 pieces of metadata without a line of code. Compare that to the hundreds of thousands if not millions of lines of code you'd find in any other enterprise application. With fewer moving pieces, we expect our applications to be easier and faster to build, maintain, and upgrade.
- Definitional development provides a more structured approach to programming, eliminating the potential undesirable "side effects" which are possible with the wider degree of freedom developers have when they are programming by coding in Java (or any other 3GL language). The methods in Workday's object model are created from definitional templates — that is, by selecting from lists of valid values — not by typing.
- Workday's object model tracks all references to an object (and all other objects that the object references), making it possible to see exactly where and how something is used — so it's easier to make changes the "right" way rather than having to kludge something because you don't have time to understand all the ramifications of the changes you're making. Since it is difficult to understand every place an object may be referenced or updated in large code lines, changes often are made in one local area where their impact can be isolated, or changes are made conditionally to isolate the impact. Over time, this leads to redundant, more complex, and therefore harder-to-maintain code. We refer to this as *application decay*. Workday believes systems built in a completely definitional fashion can avoid application decay and can therefore be maintained over time at a significantly lower cost.

Kent Beck, the author of *Extreme Programming Explained*,¹ notes that the three fundamental iterative steps of computer programming are "Make it run, then make it right, then make it fast." Complexity and bugs are typically introduced by each step, especially the "make it fast" (optimization) step, whereas the primary business value resides in the "make it right" step. In Workday's definitional development methodology, our Object Management Server takes care of the "make it run" and "make it fast" steps, freeing the developer to concentrate solely on "make it right."

All in all, we believe that "The best code written is the code that is *not* written."

¹ Beck, Kent. *Extreme Programming Explained: Embrace Change*. Boston: Addison-Wesley Professional, 1999. 224 pages.

Presentation (User Interface) Separate from Business Logic

History and experience have shown that user interface (UI) standards change rapidly, resulting in the need to rework the UI for business applications every five years or so. Workday's Object Management Server, on the other hand, contains business logic that we believe should last much longer — on the order of 20 years or more.

We have addressed this issue by making our UI layer completely separate from our data and logic layer. This will enable Workday to make improvements to the UI on an ongoing, as-needed basis, without affecting the data and logic in the underlying system.

Another benefit of a separate UI layer is that all requests that flow into the application for data and logic can be treated exactly the same — regardless of whether they come from the UI or from other systems. This is in keeping with the principles of true peer-to-peer computing.

Summary

The full promise of the ubiquitous computing wave cannot be realized solely by wrapping legacy ERP applications in additional layers of technology that enable them for web services. To fully reap the benefits of the new peer-to-peer world, new applications are needed that are natively peer-to-peer and natively built around the standards of service-oriented architecture.

Using pure object-oriented design, Workday has pioneered new applications that encapsulate data and logic together; such applications communicate on the basis of loosely coupled XML transaction processing, resulting in the maximum possible agility. Workday's revolutionary approach to application development means that applications are easy to change as your business changes, and can be cost-effectively integrated with other systems. In addition, because Workday's user interface layer is separate from the system's data and logic layer, it can be swapped out and replaced by a new UI without affecting the underlying system.

About Workday

Founded in March 2005 by Dave Duffield, Aneel Bhusri and a recognized team of enterprise software veterans, Workday is building the next generation of on-demand, ERP solutions to meet the needs of today's dynamic and global businesses. Workday's Enterprise Business Services were built using the most modern, standards-based technologies, providing an unparalleled level of agility, ease of use, and integration capabilities. The company recently delivered its first suite of products, Workday Human Capital Management (HCM) suite, and will eventually expand its offerings to include Workday Financial Management, Resource Management and Revenue Management suites. For more information about Workday, please visit www.workday.com.