



THE DELPHI GROUP

Portal Lifecycle Management: Addressing the Hidden Cost of Portal Ownership

January 2001

WHITE PAPER

| | |
|------------------------------------------------|----|
| The Portal Imperative | 2 |
| The Hidden Challenge of Portal Maintenance | 2 |
| The IT Bottleneck | 3 |
| Managing Portal Components | 4 |
| The Challenge of Disposable Software | 4 |
| Coming to Grips with the Need for Speed | 5 |
| The Iterative Stages of Enterprise Portals | 6 |
| An Integrated Approach to Lifecycle Management | 7 |
| Portal Application Utility Curve | 7 |
| The Leverage of Visual Development | 8 |
| The Leverage of an Open Framework | 9 |
| The Importance of Validation | 10 |
| Comprehensive Access | 11 |
| End Note | 11 |

Further information available from:

Mongoose Technology, Inc.
17225 El Camino Real, Suite 340
Houston, TX 77058
281.461.0099 p
281.461.0505 f
info@mongoosetech.com

<http://www.mongoosetech.com>

Since their introduction just a few years ago, enterprise portals have evolved from a random assortment of static content to the complex integration of online software applications. From this newfound complexity has emerged an unanticipated cost center in the ongoing maintenance of these applications that now typically exceeds the system's initial rollout cost. From this has emerged the need for Portal Lifecycle Management, an important feature requirement typically not addressed by the current generation of portal products and applications.

A large source of this unexpected expense lies in the maintenance of software code comprising the portal application environment. In the absence of specific tools or capabilities designed for managing application lifecycles, making even basic changes to the portal environment often represents a complex re-coding effort requiring significant technical resources.

This problem relates to the fact that many first generation portal development environments evolved from content management origins and were never intended for the diversity of object types that comprise today's enterprise portals. Needed is a development environment designed with an application orientation, rather than a content-centric perspective, which offers the ability to manage portals at component-level across the entire application lifecycle.

One of the most promising solution frameworks to emerge in this area is PortalStudio™ from Mongoose Technology. This document examines Mongoose's PortalStudio within the context of the portal management lifecycle, illustrating the inherent cost introduced by development environments that who have failed to address this important area of portal ownership.

The Portal Imperative

The inspiration for enterprise portals can be found in virtually every organization. It is the proverbial islands of automation represented by the many software applications scattered throughout individual departments, regions, and workgroups. Each is implemented in response to a particular problem or need, yet never quite connected to the periphery of systems and processes it is otherwise directly related to.

It is easy to imagine how this problem compounds over time, eventually leading to a state of information gridlock, where the growing number of disparate applications unable to communicate requires an increasing amount of human intervention (read “time” and “cost”) to manually sort through the morass of information. This is precisely the problem portal software seeks to resolve, by offering a single point of access environment to all enterprise information sources.

Through the examination of 200 organizations currently using or deploying portal software, Delphi Group determined that 88% of the time the motivation behind the initiative was to offer an environment that “provides single point of access to information.” Across the same 200 organizations it was found that “ease of integration with third party applications” was the single greatest factor in selecting portal software, cited 70% of the time.

These figures merely reinforce what is intuitively obvious to anyone inside of an organization using more than software application. That switching from window to window creates a major headache for users, and as such integration issues have become a top priority for most organizations. Information contained in one application will surely be required in the processes and decisions involved with another, and requiring the individual user to manually bridge these gaps leads to frustration, lost productivity, and inevitable mistakes.

The issues of application integration and information aggregation, a throbbing point of pain for most organizations, has led to the “portal imperative” – the emergence of portal software as a universal integration mechanism. For the first time since the introduction of the GUI, portal software presents an order of magnitude jump in the accessibility and ease-of-use of integrated applications. Despite the efficiencies and strengths of the Internet protocols on which the portal is based, this task represents a challenge that has never been adequately addressed by previous information systems.

Portal Maintenance

Once deployed, the issues surrounding the ongoing maintenance of portals presents what is most often an unexpected, under-appreciated, and unnecessarily high cost associated with portal ownership.

That virtually every organization today is examining the notion of portals on one level or another should come as no surprise, when considering the ubiquity of the problems they resolve. As with any fast-moving trend, however, many portal decisions will be made without fully weighing the long-term consequences. These mistakes are an inevitability in the beginning. But rather than an inescapable legacy, they should be a lesson for subsequent generations of portal deployments.

The lesson to be learned from the first generation of portal deployments is “do not neglect portal maintenance.” Unfortunately for many early adopters, they bought into the myth of the “portal-in-a-box” and with it the notion that a portal will exist in perpetuity without ever changing. By failing to address the issue of maintenance in the beginning, these organizations have created an unnecessary burden of cost and time that surfaces when making the inevitable modifications the portal will require.

The needless cost of portal maintenance directly results from the lack of adequate component-level tracking capabilities in the majority of today's portal deployment environments. As a result, even basic changes to the portal environment often require expensive technical resources and wasted time spent rediscovering context of the portal's design.

If portals were simply collections of static content, the issue of maintenance would be resolved by basic content management capabilities. But the reality is that portals are on-line applications as complex as any enterprise system. Each layer incorporates a number of specific software components, which should each be managed distinctly from content presented within the portal.

The problem, however, is not that portal solution providers or the organizations using them have denied this complexity. As indicated above, the vast majority of organizations place application integration at the top of their portal priorities. Likewise most portal environment come with a variety of application connectors. The problem is that most portal solutions fail to offer a design environment for tracking and maintaining portal components.

Each of these connectors, the rules and context surrounding them, as well as the interface elements of portal environment, are all pieces of software. Most often they are developed within a proprietary API (application program interface) provided by the portal solution provider. Without tools that manage each piece as individual components linked to the context with which it was created, a capability called *library management services*, the portal grows increasingly unmanageable. With each change or modification the tacit knowledge surrounding its design is kept only in the heads of programmers (if at all).

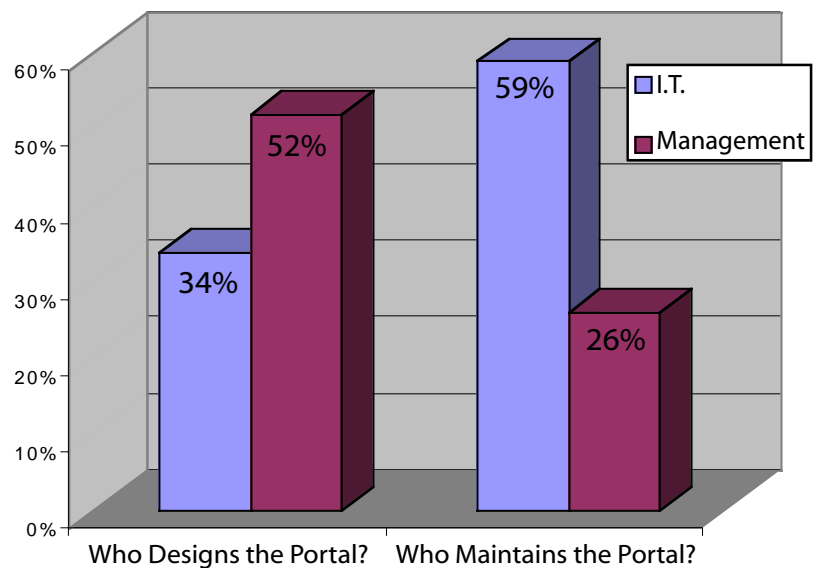
The IT Bottleneck

Deploying a portal within an environment that fails to address the issue of portal maintenance inevitably results in a growing chasm between the business-oriented sponsors of portal initiatives and the degree of technical sophistication required to manage them. Ultimately this leads to needless expense involved with reprogramming the portal with each necessary modification.

Enterprise portals are business applications for business users. This notion is validated by the Delphi Group that found individuals in line-of-business or executive management roles were nearly twice as likely as IT to hold the responsibility for a portal's initial design. In contrast, however, the majority of the time IT is responsible for the ongoing maintenance after implementation (see figure below).

The migration of portal ownership roles results from the technical competency required for the implementation and maintenance of most portal solutions, and very often leads to inconsistency between portal concept and development. It is unlikely, and often inadvisable, that IT can be completely removed from the portal deployment process. The degree to which they

The Shifting Role of Portal Ownership



are typically involved today, however, represents a significant bottleneck in the deploying and updating of portals.

This bottleneck often forces business-oriented portal architects into a compromise between taking advantage of IT resources when they are available, and taking the time necessary for proper portal planning. As a result, there is often little in common between the vision of the portal held by management and that of IT. Without a standard methodology in place to facilitate communication around application requirements, there is very likely a chasm between user expectations and portal capabilities.

This “black box” scenario where IT controls the portal’s development without direct visibility by the intended designers creates a variety of potential problems and inconsistencies. For example, taking the process of information validation away from domain experts and putting it in the hands of technical personnel opens up the potential for erroneously revealing sensitive information to the wrong party of users.

This setup ultimately results in a variety of additional costs needlessly incurred through involving IT resources in what should otherwise be a much simpler process.

The Threat to Portal Sponsorship

As has been already established, the on-going maintenance of portal can result in a cost center higher than that of the initial deployment, if not properly addressed from the outset. This is in no small part due to the use of IT resources required for the portal’s deployment and modification.

Left unchecked, the disconnect between the business expertise surrounding portal design and technical skills required to manage it erodes the portal’s value to end-users. It also threatens the portal’s sponsorship within the organization, as the resulting drain on IT

resources creates a white elephant where maintenance costs grow faster than the incremental increase in value.

Delphi Group’s analysis of portal implementations found that the most common financial sponsor of portal initiatives is executive management, outnumbering IT by 5-to-1. As this group of sponsors will likely expect demonstrable ROI from the project, the growing cost center in escalating maintenance issues presents a potential threat here. It is very difficult to determine positive ROI when the cost of ownership continues to grow faster than the rate at which additional value is created.

Another potential and often overlooked roadblock to the portal’s long term success is that of IT itself. Understandably adverse to the notion of shouldering the entire burden of portal maintenance, IT may seek to block the initiative altogether when it is not the project sponsor. In an environment where demand outstrips the ability to respond, IT staff must prioritize which projects to support. It is unlikely that any IT department with an inherited portal maintenance burden will invest its time and resources with eye towards maximum ROI. It is more likely to challenge the project altogether.

The Value of Components

This cascading set of problems relating to portals maintenance illustrates how the addition of some basic capabilities would offer a great deal of benefit to the portal builder. Consider the leverage offered by a library services capability that enable business rules to be applied to portal deployment, or a point-and-click deployment environment that offers non-technical users the ability to manage portal updates.

Just these two functional enhancements would offset many of the problems described above. The basis of this improvement, however, is directly related to the notion of managing the

portal at the component-level.

Managing portals as application components increases the value of the portal by increasing the degree to which its individual capabilities can be reused, and decreases the cost of ownership by reducing the required involvement of IT resources.

Without library management capabilities for portal components, the problems of portal maintenance grow at exponential rates. A portal with every appearance of elegance and sophistication on the outside, may on the inside be a disorganized mess of “spaghetti code” that is cheaper to throwaway and rebuild from scratch than trying to decipher and reuse.

In contrast, a portal deployment environment with a “lifecycle management” perspective enables optimal reuse and manageability of portal components. This approach to portal maintenance involves focusing on the individual lifecycle of each portal component. The concept of lifecycle management, or managing integrated applications as a set of individual components, is nothing new to software development.

Lifecycle management is a fundamental issue for managing any IT resource and will likely be a prominent aspect of the applications integrated inside of a portal. Yet to date it has been absent in most portal deployments, which have instead taken a “content management” perspective that focuses on the information viewed through the portal, rather than portal environment itself.

While the first generation may be little more than glorified intranets, today enterprise portals represent the deployment platform for a complex array of on-line software. This shift has changed resource requirements from that of website management to an environment with the ability to individually maintain application components. Without a strategy for lifecycle management, these

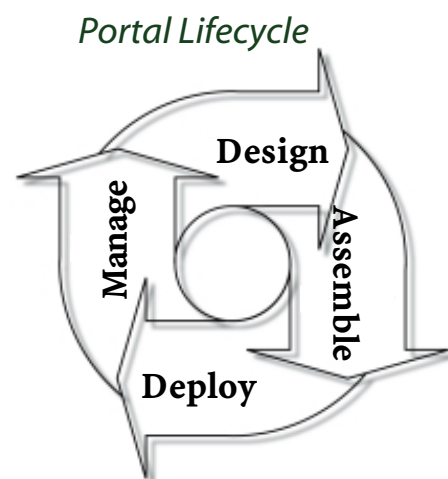
components will quickly become unwieldy and difficult to maintain.

The Iterative Lifecycle of Portal Deployment

Just as any software application, enterprise portals possess a unique, yet definitive lifecycle. In every case, however, they each follow the same pattern of development: they are first born as a concept, validated as a plan, deployed as an application, and ultimately retired.

Many portal initiatives suffer from the self-imposed challenge of taking a “ready, fire, aim” philosophy of planning the portal after its deployed. This is indicative of the same problem described previously, where the portal is developed as a single mass of software. Any change to this style of portal will often require more time to decipher than is otherwise spent redeveloping.

A better approach is to take an iterative approach that follows the same pattern described above. To simplify, we call the four stages of this iterative cycle: *design*, *assemble*, *deploy*, and *manage*. Portal lifecycle management is a matter of following multiple iterations of this cycle.



The initial phase of portal deployment is design. In this stage the portal architect determines the logical components of the

portal, mapping application components to business requirements. Of particular importance here is the ability to accurately depict the functional requirements of the portal within the specific context of the portal environment's capabilities, such as connections to third party applications.

These components create a number of compatibility issues and maintenance schedules. For example, a connector may be defined for a particular version of an ERP system. When this version changes, the portal architect must have the resources to determine where the connections are located. Here a design tool connected to library services would offer significant advantage in tracking portal components.

Once individual components are defined and linked in the design stage, the portal architect will assemble them into the logical prototype of the portal. This is a "virtual portal" stage prior to deployment where individual components are tested to verify they will work under regular operation of the portal. Ideally this requires a test bed to simulate the final production environment. It requires control mechanisms within a library management capability to ensure that the components used in the prototype are the same ones used in the deployed solution.

Once the portal prototype has been validated and the target platform is selected, the portal architect will deploy the software across the enterprise. The key objective at this stage is to guarantee that all parts of the design arrive as intended and intact at their final destination. Tools of value at this stage are those that enable both trouble shooting and tracing problems encountered when creating the deployed portal.

The manage phase of a portal represents the on-going maintenance issues which fold back into the previous steps as the portal evolves through multiple iterations of the lifecycle.

During the manage phase the portal development provides the framework to identify and initiate changes to the portal.

Of particular importance to the manage stage, although applicable to all stages of the portal lifecycle, is the ability to recognize the different roles of individual participants in the deployment and maintenance of the portal. For example, the design of the portal interface should be determined by designers and graphic artists, who may have little interest in the technical minutia and be overwhelmed dealing with the portal at a granular level. Similarly, programmers may want to skip the GUI interface of a portal development environment and deal directly with code.

The notion of recognizing participant roles is fundamental to all areas of application development, but is of particular importance in the context of portal lifecycle management. Ensuring the integrity of business processes means they must reflect organizational rules and behavior, not technical limitations. For portals this requires the ability to separate application integration details from the presentation layer (e.g., the graphical interface with which the user interacts).

Over time the individual lifecycle of each individual component will emerge as a distinct entity from the life of the portal itself. Recognition of this is critical to ensuring the portal is controlled by the organization and not visa versa – a notion enabled by a development environment that addresses the iterative approach to portal lifecycle management.

While the process above speaks to an idealized scenario of portal deployment and maintenance, the ease with which it is facilitated is largely a function of the chosen development environment. Although any portal solution will support the concept of

lifecycle management, few offer resources necessary to fully leverage the value of this approach. The best option is an integrated portal development environment specifically designed to support an iterative lifecycle software component management.

An Integrated Approach to Lifecycle Management

One of the few frameworks available to address the issues surrounding portal lifecycle management discussed above is the PortalStudio Interactive Development Environment™ (IDE) from Mongoose Technology. This environment offers portal architects an integrated approach to portal lifecycle management by way of a visual development environment, component library, and various portal maintenance provisions.

Presenting a component-based approach to portal deployment, the PortalStudio IDE allows users to select elements, from canvases and skins to application components and connectors. One aspect that differentiates PortalStudio from other portal environments on the market is the presence of a visual development environment.

This feature facilitates a focus on the design intent of the portal, rather than the technical minutia of raw HTML and XML code (see page 9 for an example of the PortalStudio IDE visual development environment). This allows non-technical users to focus on the business context of the portal, allowing programmers to focus on the creation of individual components.

To appreciate the difference here, first imagine looking at the context visually communicated by an organizational chart, then consider the impossible task of trying to infer a company's reporting structure and relationships by reading an alphabetical listing of employees. This difference illustrates the benefit of visual representation.

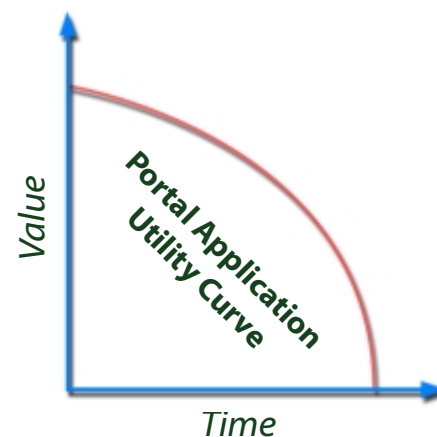
In the case of portals, this difference is translated into increased speed of deployment and reduced costs of ownership. The latter is a complex issue and is discussed throughout this document. The speed with which portal capabilities can be deployed, however, is inseparably linked to portal value and should not be overlooked.

Portal Utility Curve

Within the span of a single career, we have shifted from a world where a week's delay was acceptable response time to an "urgent request," to an era where most organizations take for granted the 24/7 accessibility of information by partners and customers. All ideas, innovations, and improvements follow a utility curve where value is limited by the time it takes to implement them.

Consider your last great idea for innovating within your organization. What was the greatest challenge to mobilizing the organization? Was it the lack of a fax machine or FedEx services or even the time to manually update a Web page? The answer more likely was that the speed of information delivery was incidental. The real delay undoubtedly came from the need to translate the idea into an actionable context, informing involved parties and assembling necessary resources.

These same dynamics suppress the value of any portal application component. The clock is ticking on any idea as soon as it has hatched (or sooner) and its ability to add value starts



only after the moment it is made actionable. Similarly, the value of any portal or application service is only realized once it is accessible to users.

The portal utility curve represents the rapidity by which it can be deployed and adapted to the changing conditions within the organization. As value is only realized once the component is deployed, the value of any portal deployment environment is directly proportional to the speed with which changes can be implemented.

In this regard, PortalStudio's component-level management optimizes portal utility by facilitating the rapid deployment of both the portal environment and subsequent modifications.

Facilitating Rapid Development

Given the time crunch facing all areas of application development, none the least of which portals, it is unrealistic to expect software development practices to spontaneously change overnight. Developers are not likely to drag-out the length of a project simply to make it the end product more manageable at a later time. Instead the most likely way to compel this behavior is to provide a set of tools that makes it easier (and faster, and cheaper) to do things right, such as developing components and the portal environment in anticipation of reuse and maintenance issues.

One way that PortalStudio facilitates this is by enabling both the portal structure, and the individual components that comprise it, to be developed and managed independently. Separating the portal's logical definition (e.g., business rules and relationships connecting portal components) from the actual deployed portal allows development efforts to be focused on isolated components, maximizing the opportunity for reuse. This also minimizes the related cost and effort involved in updating both individual components and the overall structure of the portal.

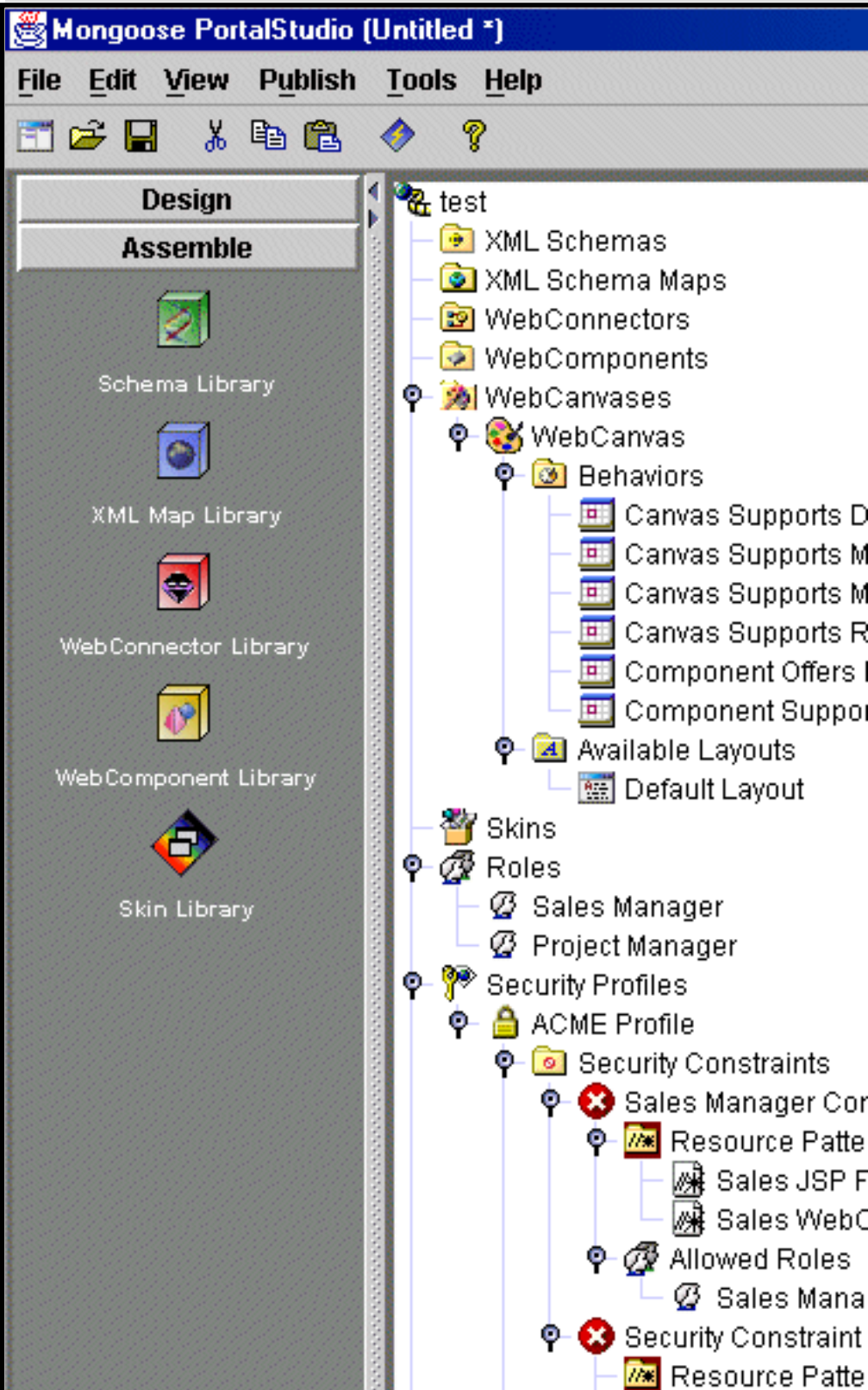
This concept is a familiar one to most areas of software development, but among portal deployment environments it is largely unique to Mongoose's PortalStudio. This logical separation also allows the portal to be developed on one platform, such as Windows 98, and deployed to a multi-server UNIX production environment or an NT server, or any platform supporting the J2EE (Java Enterprise Edition) open source language.

Yet the most obvious benefit of this approach is that over time a library of clearly defined components may be developed that enables both portal design and maintenance to be executed by individuals with little more than a cursory understanding of the development environment. It also continues to enforce business rules and control mechanisms required to enforce the portal's integrity.

Validation and Visualization

The intuitive interface of the PortalStudio visual development environment allows the portal architect to design, deploy, assemble, and manage a portal across multiple iterations of its lifecycle within a single point-and-click framework. By allowing a wider range of individuals with various levels of technical competency to contribute to the portal, PortalStudio enables greater speed of development and maintenance of the portal. Coordinating this collaborative effort, however, requires specific provisions for validation and ensuring the integrity of the portal remains intact.

PortalStudio presents a number of aids for the novice user to minimize the need to create custom code, such as stock templates, canvases, connectors, and components. In addition to those items that are offered by Mongoose, other libraries can also be obtained from third-party sources and developed internally. By allowing a secure method for creating and reusing these components, their value increases dramatically over disposable software.



PortalStudio's IDE Visual Development Environment

Another aspect of Mongoose's PortalStudio visual development environment is the presence of drag-and-drop web component functionality. This feature maximizes the utility curve for portal components by placing the ability to create and assemble components literally at the portal architect's fingertips.

Here again, PortalStudio is distinguished from most other portal environments by its perspective of the portal as a set of components, rather than a conduit for electronic content. This also reinforces the notion of portal structure, by enabling components to be constructed in a hierarchical fashion with application inheritance.

Further differentiating PortalStudio as "component-centric" versus "content-centric" is the presence of a sophisticated component-level library management facility. This feature tracks the development of individual components within the portal and facilitates rapid updates.

For example, a connector component may require modification to synchronize the version with the application it is connecting to. Where most portal environments would require cumbersome API-level coding, the library management services in PortalStudio facilitate the identification and modification of this component in a fairly straightforward process.

Leveraging Standards

Amongst portals, as with other software segments, much attention is given to the issue of support for open standards. As the portal grows and evolves with the organization, new applications and data sources must be accessed. As this requirement increases, so does the premium offered by open standards.

As previously illustrated, portal product differentiation and cost of ownership are directly related to lifecycle management issues. An open environment, such as one that includes J2EE extensions for the portal API or XML as an underlying format for both content and logic definition, will have a significant impact on the cost to deploy and maintain a portal.

Leveraging an open framework in this way significantly reduces cost and effort associated with the integration of third-party products, by leveraging standard plug-ins and connectors as they become available. As a result, the portal owner will be much less likely to be at the mercy of legacy tools nor will there be as great a risk of vendor lock-in.

Ease of Integration

While the user environment a portal offers is one factor in determining its value, the real issue is how well it integrates with third-party applications. Personal niceties like stock quotes and information kiosks may enhance the portal appeal to end-users. But productivity gains will be a function of connecting to mission-critical information sources. This is both the greatest source of value and the greatest cost of portal ownership.

Over time, this complexity will only increase with the organization's need to introduce new tools and information sources. The ability to sustain its value means the portal must add these new resources without requiring a fundamental redefinition of the portal's architecture. This aspect is the dividing line between portal environments that focus on

lifecycle management and those otherwise locked into their initial design.

With its focus on component-level management and leverage of existing standards, PortalStudio offers several advantages over other portal environments in the area of application integration. One is through the use of predefined components, called *WebConnectors*, to access otherwise hard-to-reach third-party environments such as Enterprise Resource Planning and Customer Relationship Management applications. This allows these systems to be included in the portal quickly, without requiring API-level integration.

XML Transformation

PortalStudio also includes an integrated XML server that facilitates the development of custom front-ends for legacy systems. Also provided is an XML-specific mapping function that enables non-technical users to define logical links and business rules specific to data transformation and the aggregation of content from a variety of disparate sources.

These capabilities integrate with the IDE visual environment, allowing users to navigate through a variety of information sources and connector components, rather than requiring code-level hard wiring of application integration.

Similarly, PortalStudio also offers wireless capabilities and connectors to WAP-enabled devices. This is enabled by a context engine for the portal's presentation layer that automatically determines the best output for the current user context based upon user preferences.

A WAP phone will receive a different set of text than a user recognized as having a full browser; or a user will be delivered content in the language identified in his profile, browser, or access source. This capability significantly reduces the development and maintenance efforts required for localization and multi-platform deployment.

Portal Lifecycle Management Checklist

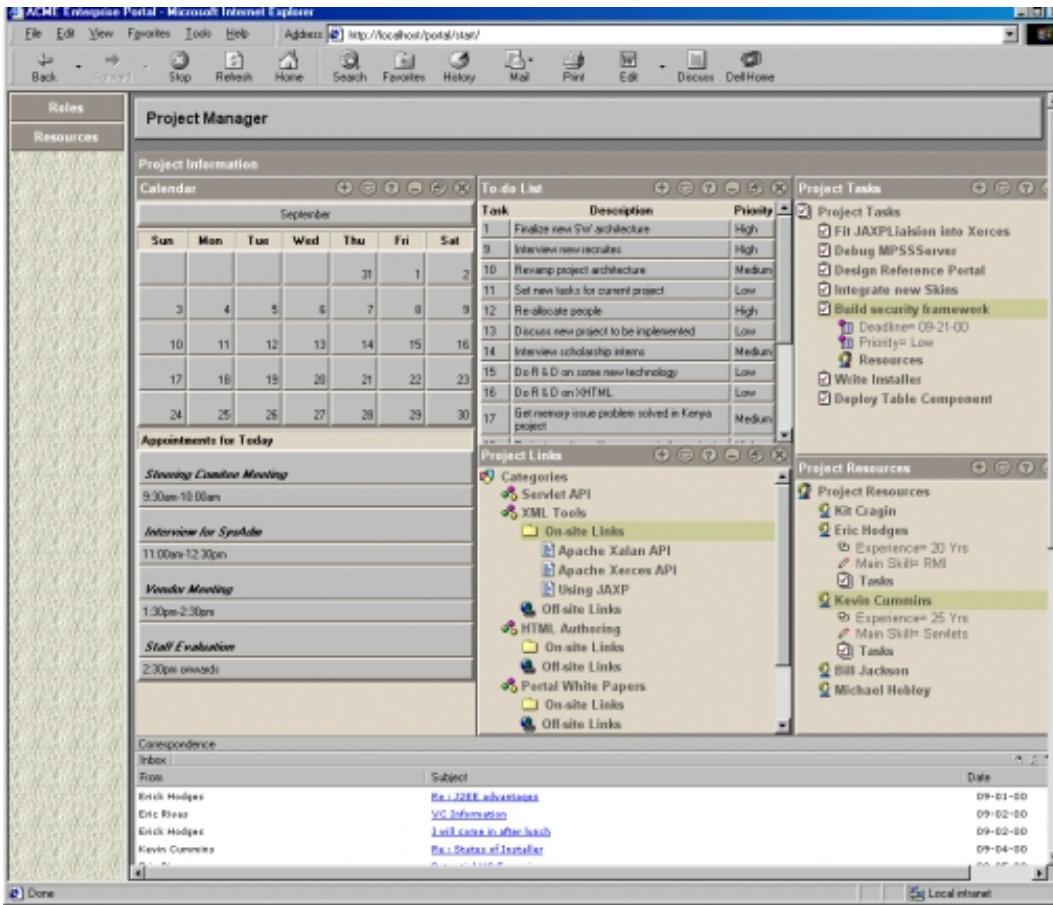
Portal lifecycle management is about making the most of portal software assets. This is not a prescription for radically alternating portal development practices. It is rather a high leverage proposition where basic changes in approach can yield substantial returns. Although the benefits of portal lifecycle management can be realized at any stage in the cycle, the greatest value is realized when initiated at the initial stage portal of deployment.

Realizing the benefits of portal lifecycle management is a function of both practice and technology. While any development environment supports the concept of a portal lifecycle, a core set of capabilities greatly enhance management of software assets for optimal value. Below is a checklist of portal product features that most significantly effect the portal lifecycle.

| | | |
|---------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Component-level Management | Optimal leverage and reusability of software assets requires component level management, including the ability to define the context surrounding each portal element. Maintenance costs are minimized by the ability to identify and isolate each component, allowing the portal to be modified at an elemental level. | Mongoose's PortalStudio allows portal software to be managed as individual components, encapsulating both business and technical context within each element. This allows software components to be defined outside of the portal environment then assembled within the visual development environment. |
| Library Services | Successfully managing software components requires the ability to control access and behavior of each portal element. This must adhere to the rules and roles associated with each participant in management process. The development environment must be aware of how and when each component is used. | PortalStudio provides role-based access to software components. Based on access privileges users may alternatively be able to modify a component, make a copy, or just add it to a portal. This maintains the integrity of software assets without introducing the complexity of a code management system. |
| Iterative Development Process | Managing the portal lifecycle is most successfully facilitated with a development environment that recognizes the iterative stages of the development process. This requires an awareness of each state of the portal lifecycle, as one stage follows the other through a closed-loop of validation, deployment, and modification. | PortalStudio's state-aware environment enables enables both components and the portals they are used in to be managed across overlapping, iterative lifecycles. individual components may be deployed and modified without conflicting with other portal elements, simplifying and accelerating portal maintenance. |
| Visual Development Environment | Visual development environments present a major time advantage by masking the complexity of underlying software components, wrapping them into interchangeable graphical elements. The speed with which visual components can be assembled into portals far outpaces the rate code can be written. Likewise, the rate at which modifications can be made is even faster in this way. | PortalStudio's Integrated Development Environment facilitates the rapid deployment and modification of portals through a point-and-client process of linking graphical elements ranging from software components to predefined business rules, such as user roles and presentation options. It also allows visual validation of the portal's design. |
| Separation of Application and Presentation Logic | A Portal's value and flexibility is optimized by the ability to separate application rules from the presentation environment seen by users. Presenting different information and application options based on roles, location or access medium allows each component in the portal maximum possible use. | PortalStudio allows all components of the portal to be defined separately from how they are deployed. This enables a high degree of personalization, as well as optimal leverage and flexibility of development platforms. A portal deployed for Unix can easily be repurposed for Windows NT or a WAP server, or each platform simultaneously based on individual requirements. |

End Note

Today's notion of the intelligent enterprise was brought to light by the availability of digital technology that provides easy access to large volumes of data. It is therefore ironic that the same organization has grown dumbfounded by a dizzying array of tools used to access the many disparate repositories in which the volumes of data are stored.



Porta I Deployed with PortalStudio

Addressing this problem has been the charter of portal software. Yet the various approaches thus far have overlooked a subtle, but significant problem - the maintenance requirements for these access points have expanded faster than the organization's ability to support them. The problem is not simply one of building conduits to content but rather managing the on-line applications which make content actionable.

Mongoose's PortalStudio presents an application component view, rather than a content management approach, to facilitate the development of data source connectors, application logic, and presentation

layers which define the logical constructs of the portal. Rather than hard wiring these components, the modular approach significantly reduces development time as well as the likelihood of product defects, the latter facilitated through explicit validation provisions.

By placing the control of the portal in the hands of individuals establishing needs, the direction of the portal will be driven by those who will derive the most benefit from its use, significantly reducing the chasm between portal needs and capabilities.

The information contained in this document is intended to provide an overview of a specific product and vendor at the date of publishing. Facts presented have been verified to the best of our ability with the vendor and actual users of the product where indicated, however, Delphi cannot insure the accuracy of this information since products, vendors, and market conditions change rapidly. Delphi Group makes no implied or explicit warranties, endorsements, or recommendations in this report nor should such warranties be inferred from its contents. A complete assessment of your specific application, the method of implementation for a given product or technology, and the current state of that product must be considered in order for a recommendation to be made on any product's suitability for your purpose, needs and requirements. Delphi Group is a leading provider of business and technology advisory services to global 2000 organizations.

Delphi Group
Ten Post Office Square
Boston, MA 02109

p(617) 247-1511
f(617) 247-4957

www.delphigroup.com

