

# THE IT ORGANIZATIONAL GAP

---

## THE OVERLOOKED ROLES

PRINTED: WEDNESDAY, DECEMBER 27, 2006

---



---

# THE IT ORGANIZATIONAL GAP

THE OVERLOOKED ROLES

---

## TABLE OF CONTENTS

<b>EXECUTIVE OVERVIEW.....</b>	<b>1</b>
<b>INFRASTRUCTURE DEVELOPER.....</b>	<b>2</b>
<b>Definition.....</b>	<b>2</b>
<b>Examples.....</b>	<b>2</b>
<b>EMBEDDED ARCHITECT.....</b>	<b>6</b>
<b>Definition.....</b>	<b>6</b>
<b>Examples.....</b>	<b>7</b>
<b>CONCLUSION.....</b>	<b>10</b>



---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

### EXECUTIVE OVERVIEW

I have been in the IT business for almost a quarter century. In the handful of companies for which I was an employee, and the several companies for which I have been a contractor, there is a consistent gap in the organizational chart. For understandable and completely justifiable reasons, the IT organizations are focused on delivering business functionality to the enterprise. The roles that are typically missing in the organization are related to the IT software infrastructure. These roles create the opportunity for efficiencies in your organization that will contribute ultimately to the ability to deliver higher-quality software in a more timely manner to your business customers.

This will define those roles and discuss why they need to be added to your IT organization chart.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

### INFRASTRUCTURE DEVELOPER

The term infrastructure as applied to IT typically means the physical components and systems software in the organization. Examples include the network routers and computers as well as the operating systems, such as Microsoft Windows and Unix. Most corporations staff positions to support these elements and those staff members typically have titles such as “Systems Programmer”, “Network Engineer” and “Systems Administrator.”

Certain infrastructure domains have sufficient complexity that they warrant a specialized staff of IT professionals. The most common is the support of the systems software that implement relational databases. Most enterprises have an organization of DBAs that support multiple applications across the enterprise. They may be working on specialized areas in the database domain, such as data warehouses and ETL<sup>1</sup>.

These roles serve the IT operational organization, but there is still an area within the operational realm that is largely under served.

#### DEFINITION

The infrastructure developer is responsible for writing the software that provides operational support for applications. The code assists the operational staff in areas such as problem diagnostics and performance monitoring for capacity planning and reporting of metrics related to service level agreements. While most IT professionals realize the importance of being able to plan for growth and diagnose problems in the production environment, the budget for any specific project can not warrant the expense of building rich operational software within the application.

#### EXAMPLES

One of the key tools to diagnosing problems in a production environment is logging capabilities. The systems software can report on the health of the operating system and major subsystems, but it cannot look *inside* the application to determine its health. Most senior applications developers will log significant events within their application, espe-

---

<sup>1</sup> “ETL” stands for “Extract, transform and load”. It is a specialized area that moves data throughout the various data stores, applying business rules along the way. A common example is moving detailed operational data to a data warehouse for decision support

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

cially error conditions, so that the operational staff and programming staff can diagnose the issue. Most logging implementations will record this information to a disk file.

A problem arises when that application has been deployed on multiple machines in a clustered environment. Clusters are commonly implemented in medium and large enterprises for web-based applications. They present the illusion of a single application running on a single computer. In reality, the request from the web browser is sent to one of several instances of the application on multiple computers. Clusters are implemented primarily for two reasons. First, it allows for the application to scale beyond the capacity of a single computer<sup>2</sup>. Second, clusters provide for greater availability of the application, since a single machine in the cluster can fail, but the application is still available because it can run on other members in the cluster. I have worked in large enterprises that have had sixteen members in a cluster.

The diagnostic problem that arises is that we now have a log for every machine in the cluster. If we are trying to diagnose a problem, we have to look through several logs. One of the steps in determining the source of the problem is to determine its scope. Is the problem isolated to a single application in a single instance of the cluster, the application across the cluster, all applications in an instance of the cluster or all applications in the entire cluster? That means that we have to start aggregating log entries across multiple logs. That is exceptionally time-consuming and inefficient.

What was needed in the example above, which has been a real customer experience for multiple customers of mine, was a logging facility that could record information from multiple computers to a single location. In addition to recording the log entry from the application, it needed to also record the host from which the log entry was generated and the member in the cluster from which the log entry was recorded. Ideally, the logs could be view in real-time, but could also record to a database for later analysis.

Once the operational logging information is recorded in a database, the opportunity exists for analysis and reporting of errors that may not have been reported by the end customer. This capability increases the quality of code since the operational and programming staff have insight into the existence and frequency of problems that would otherwise go unnoticed.

If this log management software sounds like a significant amount of work, that's because it is. It is an IT project in itself. These types of efforts rarely get the focus or funding because most enterprises fund IT projects based on business requirements. Infrastructure software serves *all* of the applications in the enterprise.

---

2 This is termed “horizontal scaling.” The term “vertical scaling” essentially means buying a faster computer.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

We in the IT organization fail to fund projects that are of benefit to *our own* organization<sup>3</sup>, in spite of the realization that an effective IT organization is crucial to the success of almost all modern enterprises.

There are many other examples of cross-project infrastructure software. One area that is getting increased attention is application security. Regulations such as Sarbanes-Oxley and HIPPA as well as more stringent security requirements to interoperate with financial institutions has brought security to the attention of C-level executives. Implementing stove-piped security within each application creates security problems for at least two reasons. First, most applications developers aren't skilled in the subtleties of security. Second, multiple security data stores increase the likelihood that authority will not be added or revoked in a consistent manner. There is also the inefficiencies associated with the maintenance of multiple userids, passwords and permissions across multiple applications.

Ideally, the enterprise would have an implementation of security that is used by all enterprise applications. The implementation would be accomplished in such a way that the enterprise would not be tied to a particular security technology. This would enable an organization to switch to a different security technology with little or no effect to the application code. For example, an enterprise may implement Kerberos authentication, but later find through mergers and acquisitions that they need to move to Windows ActiveDirectory for authentication services.

Many organizations are viewing the implementation of an enterprise portal as the solution to this problem. Most portal implementations have a mechanism to authenticate a userid and password once and then use those credentials for multiple software services residing within the portal. This is a step in the right direction, but it doesn't completely solve the problem. There are other deployment models for applications that aren't well-suited to a portal environment. Most enterprises will run batch processes and implement network-based services that are not web-based services. Since these aren't implemented with web technologies, they are unable to use the authentication from the portal.

Application security is a specialized domain. The customer for application security services usually isn't the business unit that is requesting that an application be developed. Most enterprises have an organization that is responsible for various aspects of network security such as firewalls, virus protection, intrusion detection as well as application security requirements. That same organization may also issue security badges, hire security guards and be responsible for physical security of the enterprise assets and employees. That organization would be the customer that would spell out the requirements for secu-

---

3 The cobbler's children have no shoes.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

rity that the infrastructure developer would develop and support. The customer in a specific business unit is doesn't have the enterprise-wide view of security needs.

A similar case can be made for performance metrics. A well-designed application will produce statistical information that can be used for capacity planning and measuring compliance with service level objectives for response time.

I think you can see that these specialized areas within IT operations typically don't have a well-defined owner or funding. If there isn't an organization within your enterprise that is specifically responsible for infrastructure software, there is a very good chance that the governance of these critical technical areas will be lacking.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

### EMBEDDED ARCHITECT

Many organizations have identified the need for architectural oversight for the applications being developed by the enterprise. The titles may be “Enterprise Architect” or “Application Architect.” The application architecture defines the “shape” of the application and often the technologies used. It will include the process and threading model for the application, the approach for interoperability and legacy integration. It will often include a high level design as well. Most application architectures are agnostic to the implementation language. That is, the architectural requirements could be satisfied by any programming language such as Java or one of the Microsoft .NET languages.

In most organizations, the team lead takes the architectural documentation and creates a detailed design and works with the team members to deliver the code for a specific project in a specific language. At first glance, that seems to cover the need, but this very typical structure is missing a key role and creates a lost opportunity cost.

#### DEFINITION

The embedded architect has skills that are similar to those of the application architect. Specifically, they are aware of advanced object-oriented techniques and design patterns. In addition, the embedded architect is a strong developer in one or more languages and knows how to encapsulate key architectural concepts in code that can leverage those architectural elements *across projects*. That is, they are *embedded* with the application development groups instead of isolated in an architectural group.

In addition to the structural software that would be implemented by the embedded architects, they would be responsible for recognizing objects in the business domain that have common use across applications. They would be responsible for creating these “mini-frameworks” so that more code could be leveraged across projects.

The key difference is that the embedded architect is not responsible for a project specifically, but is responsible for efficiencies through reuse across projects.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

### EXAMPLES

Most interactive<sup>4</sup> applications, whether web-based or rich-client applications, are best implemented using what is called a model-view-controller design pattern<sup>5</sup>. This separates the user interface of the application<sup>6</sup> – the “view” - from the objects that model the business – the “model”. The “controller” implements the workflow internal to the application and mediates the requests between the user interface and the business rules.

If done properly, the controller can be independent of a specific project and therefore can be reused *across* projects. The key is to create applications that are assembled at runtime instead of assembled at link time. In other words, if there is an application framework that can model discrete units of work within any application in general and describe the workflow between these discrete units of work externally to the application, then changes in the workflow due to changing business requirements can be made with less disruption to the existing code base. Less disruption means a quicker time-to-market and less chance of introducing code defects due to changes that occur over the lifetime of the application.

I know, that last paragraph is a little tough to digest. That's because architecture by its nature is abstract and abstractions are hard to articulate<sup>7</sup>. The key point is that the function of the controller can be implemented in a way that is independent of the specifics of a project. There are open-source projects, such as the popular Struts 1 framework and the emerging Struts 2/WebWork framework that implement this concept for web-based applications. Ideally, an enterprise would implement controller functionality that works for applications that aren't web-based as well.

There is another reason to create a controller framework that is reusable across applications. In the previous section, we described the role of the infrastructure developer that is often overlooked within an organization. While I strongly advocate the inclusion of these infrastructure elements, such as robust logging and performance metrics, within all enterprise applications, the reality is that this can cause additional work for the application developers assigned to the project. They have to integrate these infrastructure programming interface calls with their application code.

---

4 Meaning not batch

5 The abbreviation MVC is used for this design pattern as is the term “model 2”. It originated with the SmallTalk language in the 1980s so it has a long track record of success.

6 For web-based applications that seem to comprise most of today's application development, this is what is displayed in the web browser.

7 Remember the first time you saw algebra and thus saw abstractions added to arithmetic? It's not easy to grasp at first, but it is powerful.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

It turns out that the role of the embedded architect and the role of the infrastructure developer are highly synergistic. The controller is the perfect place to integrate infrastructure code<sup>8</sup> in a way that is transparent to the application code being developed for a specific project. In short, your applications get a large dose of operational richness with zero added work needed by the application developers.

Here's why this works.

There is a large amount of interaction between the user interface and the business model that is mediated by the controller. Loosely speaking, every time you click a button on a web page, your request will pass through the controller and the state of the business object model will be updated in accordance with your request. Therefore, every time we pass through the controller, we can record workflow information about your application to the logging facilities. We can also record performance metrics for use in capacity planning and performance metrics reporting. We also have the chance to check security authorizations at the controller and thus not dispatch any unit-of-work for which the user is not authorized. Since the same controller is used across applications, we get this with zero additional effort by the application developers.

**That's powerful.** You get operational richness and resilience to change for free<sup>9</sup>.

That example is in the gray abstractions of application architecture and can be difficult to grasp. There are other examples closer to the application domain that are more easily understood. Let's take an example of an application that manipulates information about a customer. There is a variety of information about a customer – addresses and phone numbers for example – that multiple applications will display and edit. Many modern user-interface implementations, such as JavaServer Faces (JSF), or the Standard Widget Toolkit (SWT), or Java's Swing, implement a component model for user interface elements. They provide for low-level graphical user-interface components such as push buttons, radio buttons, check boxes, list boxes, etc. Perhaps more important, they provide a framework for the implementation of custom components. This framework is the architectural foundation for creating domain user-interface components, such as a “customer editor” implemented as a JSF component. It would be a significant amount of work to implement a JSF component for a common business object. That effort would not be cost-justified in the context of a single project, but there is a return on investment if that component can be used across applications.

---

8 Sometimes “infrastructure code” is referred to as code for “non-functional requirements”. I never liked that term because they are very functional requirements, it's just operational function instead of business function. If you have heard of IT guys talk about non-functional requirements, much of that is operational support code like security.

9 Perhaps more important, this isn't theoretical. The author has developed code that is in production that meets these objectives.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

Quite honestly, that is a vision. I haven't done that myself nor have I seen it done within other enterprises. I think it makes sense, but I can't say with certainty that it works. I *have* taken an intermediate step that *does* work. I have created generic and abstract “item editors” that I have used across applications to manipulate project-specific business objects. That alone has provided a non-trivial productivity boosts, but I think it still falls short of the potential of that exists in object technology at an enterprise architectural level.

It's easy to get overwhelmed by the complexity in the details within the examples, so it's important to take a step back and look at what we are trying to accomplish and why. There is a new role that exists in the IT domain that didn't use to exist. It exists now because the technical domain of computer science has advanced. The computer science domain of design patterns is a little over ten years old and it took quite a while for the industry to absorb the ramifications. The notion of declarative assembly of components is possible only once advanced object-oriented design techniques and design patterns are understood. The effective implementation of declarative assembly is probably around six years old. These are new technologies that require an rethinking of the organizational structure within IT to effectively implement.

---

# THE IT ORGANIZATIONAL GAP

## THE OVERLOOKED ROLES

---

### CONCLUSION

Advancements in computer science have created software construction and deployment capabilities that were just not possible a decade ago. It is unquestionably possible to continue to run IT organizations as they were run in the 1990's and be effective in doing so. However, to ignore the possibilities that have emerged in this century is to impose an unnecessary **lost opportunity cost** for your enterprise.

In order to realize these opportunities, the IT organization needs to recognize the need for additional roles within the organization. The role of infrastructure developer is to ensure that the cross-project infrastructure and operational needs of the organization are met. The synergistic role of the embedded architects is to ensure that the possibilities for reuse are fully realized within the organization.

Without appropriate staffing and funding, these domains will not be properly served and governed. As a result, your enterprise will miss the opportunity to leverage the capabilities inherent in an understanding of leading computer science techniques.